반듯©

오픈소스SW 라이선스 가이드 4.0

문화체육관광부 한국저작권위원회

목차 (contents)

오픈소스 소프트웨어 라이선스 가이드

가이드 제공의 배경

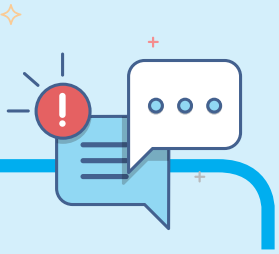
인터넷 서버, 임베디드 분야를 중심으로 꾸준히 성장해 오던 오픈소스 소프트웨어가 스마트폰 시대를 맞아 그야말로 소프트웨어 세계의 중심부를 점령하고 있다. 그와 함께 오픈소스에 관한 법적 리스크도 드러나고 있다. 국내 대기업을 포함한 전 세계 다양한 기업들이 오픈소스 라이선스를 잘못 다루어 소송을 제기당하는가 하면, 상용 소프트웨어 기업들이 특허권 등을 무기로 오픈소스 진영을 공격하고 있다. 그럼에도 불구하고 소프트웨어가 진화하는 전반적 방향을 되돌리기는 어려워 보인다. 소프트웨어 기반이 없는 국내 기업들의 입장에서는 이와 같은 흐름을 잘 따르면서 오픈소스의 법적 리스크를 최소화하는 것이 최선의 방책일 것이다.

기업에서 오픈소스에 관한 법적 리스크를 체크하고 분석하는 데 조금이나마 도움을 주고자 오픈소스 라이선스 가이드를 제공한다. 최근 수년간 국내의 오픈소스 환경은 큰 변화를 겪고 있고, 복잡한 라이선스 이슈들도 많이 발생하고 있다. 이러한 변화를 수용하여 가이드에서는 활발하게 사용되고 있는 오픈소스 라이선스들에 대해 주요 의무사항, 대표적인 프로젝트, 주요 쟁점 및 고려사항 등을 체계적으로 설명하고자 했다. 아울러 기업의 오픈소스 담당자들이 궁금해 하는 사례들을 구체적으로 다룸으로써 현장감을 살리고자 했다. 아무쪼록 이번 가이드가 우리 기업들에게 조금이나마 도움이 될 수 있기를 기대한다.

오픈소스에 관한 법적 리스크는 오픈소스 라이선스 위반, 제3자의 지식재산권 침해, 자사의 지식재산권 관리에 관한 리스크로 구분할 수 있다. 첫 번째, 라이선스 위반에 관한 리스크는 오픈소스에 관한 가장 기본적인 리스크인데, 오픈소스를 사용하면서 관련 라이선스를 준수하지 않는 경우 오픈소스 커뮤니티로부터 소송을 제기당할 수 있다는 점이다. 예를 들면 Busybox와 관련하여 국내 대기업을 포함한 전 세계 다양한 기업들이 소송을 제기당한 사례가 대표적이다. 두 번째, 오픈소스 자체가 제3자의 특허권이나 저작권 등 지식재산권을 침해함으로써 발생할 수 있는 리스크이다. 예를 들어 삼성전자와 LG전자가 안드로이드 플랫폼 기반의 스마트폰을 제조하여 판매하고 있는데, 안드로이드 플랫폼이 마이크로소프트 등의 지식재산권을 포함하고

있을 경우 삼성전자와 LG전자가 부담해야 할 리스크이다. 과거에도 BSD, 리눅스, JBoss와 관련된 법적 분쟁들이 있었고, 최근에는 특히 안드로이드 플랫폼에 관한 분쟁이 생겨나고 있는 추세이다. 세 번째 리스크는 오픈소스를 활용하는 과정에서 자사의 특허권이나 저작권이 영향을 받을 수 있다는 점이다. 예를 들면, 오픈소스를 활용하면서 자사가 수정했거나 새롭게 작성한 코드를 GPL 등의 오픈소스 라이선스로 배포하게 되면, 해당 코드와 관련된 자사의 특허권이나 저작권에 오픈소스 라이선스가 적용된다. 그 결과 자사의 특허권이나 저작권을 경쟁기업들이 무료로 활용할 수 있게 된다.

이상과 같은 법적 리스크들은 기업에서 충분히 관리 가능한 리스크들이다. 따라서 경영자 및 관리자들은 오픈소스에 관한 법적 리스크를 이해하고 자사의 리스크를 분석하고 대비책을 마련할 필요가 있다. 오픈소스를 사용할 때 발생하는 리스크는 법적 측면 이외에도 기술적 측면과 사업적 측면에서도 발생하게 된다. 각각의 리스크들이 모두 중요하기 때문에 세부적으로 살펴보는 것도 중요하지만, 특히 CEO 등 경영자, 관리자의 입장에서는 모든 리스크들을 종합적으로 고려하는 시각을 가지는 것도 중요하다. 관리자의 입장에서는 각각의 리스크를 점검하되, 모두를 만족하게 할 수 없는 경우에는 우선순위를 정할 필요가 있다.



1

소프트웨어 라이선스와 오픈소스

1.1. 소프트웨어 라이선스

오픈소스 소프트웨어에 관한 법적 리스크를 이해하기 위해서는 소프트웨어에 관한 지식재산권과 라이선스에 관한 기본적인 개념을 이해할 필요가 있다. 오픈소스 라이선스는 소프트웨어 라이선스의 일종이기 때문이다. 소프트웨어 라이선스의 일반적인 내용을 이해하면 오픈소스 라이선스의 특징을 더욱 잘 이해할 수 있다. 일반적인 특징을 이해한 후에는 오픈소스 라이선스 각각의 세부적인 쟁점들을 검토할 필요가 있다.

소프트웨어에 대한 저작권 등 독점배타권을 가진 권리자는 다른 사람들이 해당 소프트웨어를 사용하거나 배포하는 것을 허락할 수 있다. 일반적인 상용소프트웨어는 그 대가로 로열티를 요구한다. 국내에서 활용되는 소프트웨어 라이선스계약에는 일반적으로 계약의 목적, 정의조항, 실시권의 설정, 계약 기간, 로열티, 기록의 보관 등에 관한 간단한 내용만이 포함되어 있다. 그런데 미국의 소프트웨어 라이선스 계약서 내용을 보면 매우 상세한 내용을 담고 있다. 소프트웨어 계약은 다른 분야에 비해 상대적으로 역사가 길지 않아서 업계의 표준 관행이나 판결 등 분쟁해결의 기준이 명확하지 않다. 그래서 당사자들은 되도록 상세한 규정을 두고 싶어 하기 때문에 계약서 내용이 상세해지게 된다.

일반적인 상용 소프트웨어 라이선스의 주된 내용은 권리자가 사용자에게 라이선스를 부여하고 사용자는 그 대가를 지급하는 것으로 구성되어 있다. 이렇게 거둬들인 수입으로 권리자는 이익을 남기고, 또 일부는 R&D 자금으로 재투자하여 소프트웨어의 성능을 개선하거나 새로운 소프트웨어를 개발하게 된다. 오늘날 대부분의 소프트웨어 기업들이 이러한 비즈니스 모델을 가지고 있다.

1.2. 오픈소스 라이선스

그렇다면 오픈소스 라이선스들은 일반적인 소프트웨어 라이선스와 어떻게 다른가? 일반 상용소프트웨어와 마찬가지로 오픈소스 소프트웨어에도 저작권 등 지식재산권이 있다. 그래서 권리자의 허락 없이 함부로 사용하면 소송을 당할 수 있다. 그런데 오픈소스의 권리자들은 되도록 많은 사람이 자유롭게 사용할 수 있도록 광범위한 라이선스를 부여하고 있다. 예를 들어 사용자들에게 사용에 대한 권리뿐만 아니라 마음대로 복제 및 배포를 할 수 있도록 하고, 소스코드까지 제공하여 마음대로 수정할 수 있도록 허락한다. 하지만 상용소프트웨어처럼 그에 따르는 로열티를 요구하지는 않으며, 대신 몇 가지 지켜야 할 의무사항을 요구한다. 구체적인 예를 통해 가장 기본적인 몇 가지 의무사항을 살펴보기로 하자.

그림 오픈소스SW 라이선스 모델

저작권, 개발자 및 기여자 정보의 표시

대부분의 오픈소스SW 라이선스는 개발자 또는 기여자에 관한 사항과 저작권에 관한 사항을 제품에 표시하거나 포함하도록 요구하고 있다.

Apache 2.0

3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work.

코드를 수정한 경우 수정한 정보의 표시

이용자가 소스코드를 수정하였을 때에는 수정한 사람, 수정일자 등 수정에 관한 내용들을 포함시키도록 함으로써, 원본과 구별할 수 있도록 한다. 저작인격권의 하나인 동일성유지권에 비유할 수 있다.

Apache 2.0

2. You must cause any modified files to carry prominent notices stating that You changed the files.

라이선스 정보의 제공

많은 오픈소스SW 라이선스들은 이용자들이 오픈소스SW에 관한 권리를 잘 이해할 수 있도록 배포자로 하여금 해당 라이선스의 사본을 함께 첨부할 것을 요구하고 있다.

Apache 2.0

1. You must give any other recipients of the Work or Derivative Works a copy of this License.

동일한 라이선스로 재배포할 것 (카피레프트)

라이선스에 따라 큰 차이를 보이는 부분은 '카피레프트(Copyleft)'[1]에 관한 사항이다. GPL을 대표로 하는 카피레프트 라이선스들은 이용자들이 소프트웨어를 수정한 후 배포하고자 할 때 수정된 소프트웨어도 동일한 라이선스로 배포할 것을 요구한다.

GPL 2.0

You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, … provided that you also meet all of these conditions:

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

소스코드의 제공

카피레프트 조항을 포함하는 라이선스의 경우, 소프트웨어를 배포할 때 소스코드까지 함께 배포하도록 요구한다.

1) 카피레프트(Copyleft)라는 용어는 원래 저작권(Copyright)에 반대한다는 의미로 "Copy-right"에서 "right" 대신 "left"로 바꾸어 사용하기 시작한 것이다. 그런데, FSF가 말하는 "소프트웨어의 자유"를 지키기 위한 구체적인 수단이 GPL이며, 그 중에서도 핵심적인 내용은 파생저작물을 GPL로 재배포할 것을 요구하는 조항이기 때문에, 보통 이 조항을 가리켜 카피레프트 조항이라고 부르고 있다.

GPL 2.0

You may copy and distribute the Program (or a work based on it, under Section 2) in object code … provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code,

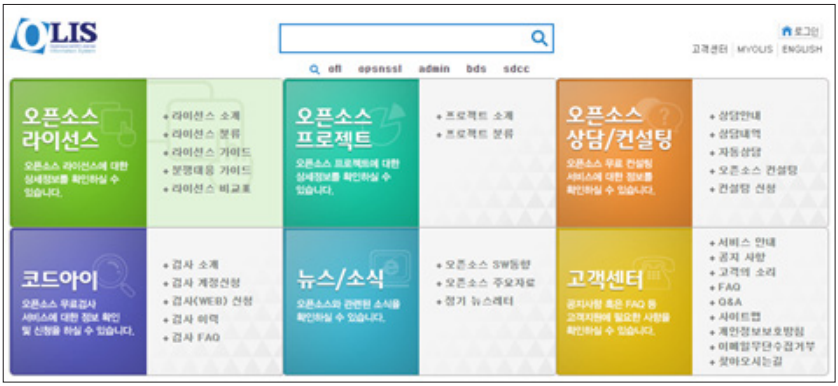
<사례> 국내 기업들의 소스코드 제공

삼성 전자와 LG 전자는 웹사이트를 통해 휴대폰, TV 등 오픈소스를 이용하고 있는 제품들의 소스코드를 제공하고 있다.

<삼성전자의 소스코드 제공 사이트>

<LG 전자의 소스코드 제공 사이트>

오픈소스 라이선스에서 공통으로 다루고 있는 쟁점들을 파악하고 있으면, 새로운 라이선스를 접하더라도 한결 이해가 쉽다. 국내에서는 저작권위원회가 운영하는 OLIS 사이트(http://www.olis.or.kr)에서 70여종의 오픈소스 라이선스를 정리하여 보여주고 있다.



<참고> OSI와 Open Source Definition

OSS 라이선스들의 구체적인 내용은 라이선스마다 차이가 있지만, 모든 OSS 라이선스가 공통적으로 가진 특징이 있다. 왜냐하면 이러한 공통된 특징 혹은 요건을 충족해야만 OSI(Open Source Initiative)[2] 로부터 오픈소스 라이선스로 인증받을 수 있기 때문이다. 몇 가지 특징을 살펴보자.

첫째, 라이선스는 소프트웨어의 자유로운 재배포를 허용해야 하며, 사용자가 소프트웨어를 상업용으로 사용하더라도 로열티 등의 명목으로 대가를 요구할 수 없다. 만약 어떤 라이선스가 '비상업용(non-commercial)'으로만 자유롭게 배포할 수 있도록 허용한다면, OSI의 요건을 충족하지 못하기 때문에 OSS 라이선스가 될 수 없다.

둘째, 해당 프로그램에는 소스코드가 포함되어 있어야 하며, 이용자가 소스코드 형태로 배포하는 것을 허용해야 한다. 예를 들면, 프리웨어(freeware)의 경우 자유로운 재배포가 허용되지만 소스코드가 포함되어 있지 않기 때문에 오픈소스SW에 해당하지 않는다.

OSI로부터 오픈소스 라이선스 인증을 받기 위해서는 이 밖에 8가지의 추가적인 요건을 충족시켜야 한다. 자세한 내용은 OSI의 사이트에서 확인할 수 있다.[3]

<상담 사례> 비상업용으로만 사용하도록 한 경우

Q: 구글의 안드로이드 플랫폼에는 netperf 라는 오픈소스SW가 포함되어 있습니다. netperf license를 찾아보니 다음과 같은 내용이 포함되어 있습니다. 그런데, non-commercial only라고 되어있는데, 저희 회사가 사용하여도 되는지요?

The enclosed software and documentation includes copyrighted works of Hewlett-Packard Co. For as long as you comply with the following limitations, you are herebyauthorized to (i) use, reproduce, and modify the software and documentation, and to (ii) distribute the software and documentation, including modifications, for non-commercial purposes only.

2) http://www.opensource.org/
3) http://www.opensource.org/docs/osd

A: 라이선스 문구에 'non-commercial' 이라는 조건이 포함되어 있으면 회사에서 상업적으로 활용하기는 어렵습니다. 다만, 이러한 조건이 포함되어 있으면 OSI의 인증을 받기가 어려우며, 반대로 OSI의 인증을 받은 라이선스에는 이러한 조건이 포함되어 있지 않습니다. 참고로 현재 HP는 netperf 라이선스 외에 GPL로도 배포하고 있으므로, 상용으로 사용할 때에는 GPL로 배포되는 netperf를 사용할 수 있습니다.[4]

<상담 사례> 표준을 구현한 오픈소스 코드

Q: Audio 또는 Video 코덱 중에 보면 ITU-T, 3GPP, ETSI와 같은 표준을 제정하는 협회에서 Reference Code를 오픈하여 누구나 참조하여 사용할 수 있도록 하고 있습니다. 그런데 이 소스들 중 일부분이 GPL 라이선스화되어 배포되고 있습니다. 이런 경우 Audio, video 코드를 상업적으로 사용하기 위해서는 어떤 절차를 거쳐야 하는지 궁금합니다.

A: 상업적으로 이용하기 위해서는 레퍼런스 코드의 권리자(특허권자를 포함)에게 허락을 받아야 합니다. 아울러 표준기관에서 배포하는 레퍼런스 코드를 직접 이용하는 것이 아니라 GPL로 배포되는 코드를 이용한다면 GPL의 조건도 준수해야 합니다.

예를 들면, FFmpeg의 주요 부분은 LGPL로 배포되고 있기 때문에 표준기관에서 배포하는 레퍼런스 코드가 아니라 LGPL로 배포되는 FFmpeg를 이용하기 위해서는 LGPL 조건을 준수해야 합니다. 한편, MPEG 기술에 대해서는 많은 특허권이 관련되어 있기 때문에 경우에 따라서는 FFmpeg를 사용하기 위해서 MPEG에 관한 특허권자의 허락을 받아야 하는 경우도 있습니다. 예를 들어 MPEG LA는 FFmpeg을 사용하는 사용자에게 추후 특허권을 주장할 수 있으며, 이에 대해 사용자는 Royalty를 지불해야 할 수도 있습니다. 그리고 FFmpeg을 LGPL로 배포하고 있는 저작권자는 해당 SW내 자신들의 특허가 아닌 제3자(MPEG LA)의 관련 특허가 있음(Non Royalty-Free Patents)을 사용자에게 공지하여, 사용시 참고할 수 있도록 하고 있습니다.[5] 물론 자신들의 특허는 LGPL로 배포하므로 Royalty-Free로 하였겠지만, 제3자의 특허는 어떻게 할 수 없으므로 사용자에게 그런 제3자 특허의 위험을 알리는 것입니다.

4) http://www.freshports.org/benchmarks/netperf/.

5) FFmpeg license 참조. FFmpeg is licensed under the GNU Lesser General Public License (LGPL) version 2.1 or later. However, FFmpeg incorporates several optional parts and optimizations that are covered by the GNU General Public License (GPL) version 2 or later. If those parts get used the GPL applies to all of FFmpeg.
Q: Is it perfectly all right to incorporate the whole FFmpeg core into my own commercial product?
A: You might have a problem here. There have been cases where companies have used FFmpeg in their products. These companies found out that once you start trying to make money from patented technologies, the owners of the patents will come after their licensing fees. Notably, MPEG LA is vigilant and diligent about collecting for MPEG-related technologies.

2

주요 오픈소스 라이선스 및 프로젝트 사례

2.1. BSD형 라이선스 및 주요 프로젝트

BSD형 라이선스에는 BSD, MIT, Apache 라이선스 등이 포함되며, 비교적 오랜 역사를 가진 라이선스들이다. 이들 라이선스는 카피레프트(Copyleft) 조항을 포함하지 않으며, 의무사항도 비교적 간단하다.

2.1.1. BSD 라이선스

BSD 라이선스는 버클리 대학에서 만든 라이선스로, 소프트웨어를 재배포할 때 저작권 표시를 할 것과 준수 조건 및 보증부인에 대한 고지사항을 소스코드 또는 문서 및 기타자료에 포함할 것을 요구하고 있다.
4개의 조항으로 구성된 BSD 4-Clause 라이선스 (Original BSD 혹은 Old BSD라고도 불림)에는 "광고 조항"이 포함되어 있다. 이는 파생 저작물의 모든 홍보물에 "This product includes software developed by the <organization>" 문구를 포함해야 하는데, 이는 문제 소지가 있는 조항으로 판단되어 1999년 7월 이를 삭제한 BSD 3-Clause 라이선스 (Modified BSD 혹은 New BSD라고도 불림)가 만들어졌다.
BSD 4-Clause 라이선스의 광고 조항은 별도의 제한 사항으로 볼 수 있어 GPL과 호환되지 않는다. 따라서 BSD 4-Clause 라이선스로 배포된 오픈소스가 GPL 프로그램을 사용하는 구조라면, 저작권자에게 BSD 3-Clause 라이선스로 변경하여 해결하라고 FSF는 안내하고 있다.
이와 같은 이유로 카 인포테인먼트 시스템의 대표적인 오픈소스 플랫폼인 GENIVI 플랫폼에서 사용하지 말도록 금지하고 있는 라이선스에 BSD 4-Clause 라이선스가 포함되어 있다.

BSD 3-Clause 라이선스에서 "제품에 대한 보증이나 홍보에 최초개발자나 기여자의 이름을 사용하지 못한다"는 조항을 삭제한 것이 BSD 2-Clause 라이선스다. 이는 BSD 라이선스 중 가장 간략하고 의무사항이 덜한 라이선스로 볼 수 있다.

<프로젝트 사례> BSD 운영체제

BSD 라이선스로 배포되고 있는 대표적인 프로젝트는 BSD 운영체제이다. 1977년부터 1995년까지 활동했었던 버클리 대학의 컴퓨터시스템연구소(SCRG)가 최초로 배포한 BSD 운영체제는 현재 FreeBSD, OpenBSD, NetBSD, DragonFly BSD 등의 형태로 변형되어 계속해서 오픈소스로 배포되고 있다.

2.1.2. Apache 라이선스

Apache 라이선스는 아파치소프트웨어재단(Apache Software Foundation)에서 만들어 배포한 라이선스이다. 1.0과 1.1 버전은 BSD와 비슷하게 간단한 내용만을 담고 있었지만, 현재 사용되고 있는 2.0 버전은 2004년에 배포된 것으로 비교적 상세한 내용을 담고 있다.

배포 시의 의무사항으로는 저작권, 특허, 상표, 권리귀속(attribution)에 대한 고지사항을 소스코드 또는 "NOTICE" 파일 등에 포함할 것과, 수취인에게 라이선스 사본을 제공하도록 요구하고 있으며, 파일을 수정하여 배포할 경우 수정된 파일에 대해 수정사항을 표시한 안내 문구를 첨부할 것을 요구하고 있다. 하지만 카피레프트 조항을 포함하고 있지 않기 때문에 반드시 동일한 라이선스로 배포할 필요는 없으며, 소스코드 제공의무도 없다는 점에서 기본적으로 BSD 라이선스와 비슷한 것으로 평가할 수 있다.

<상담 사례> Apache 코드와 BSD 코드를 링크한 경우

Q: Apache 라이선스로 배포되는 파일과 BSD 라이선스로 배포되는 파일을 함께 링크한 경우 어떤 라이선스를 따라야 하는지 궁금합니다.

A: BSD, Apache 등 BSD형 라이선스는 원 코드나 수정코드를 재배포할 때 동일한 라이선스로 배포해야 한다는 조항(카피레프트 조항)이 없기 때문에 배포자는 라이선스를 자유롭게 선택할 수 있습니다. 이것은 두 개의 라이선스로 배포되는 코드를 링크할 때도 마찬가지입니다. 따라서 기업에서는 Apache 등 BSD형의 라이선스로 배포되는 다양한 코드를 이용하여 자사의 제품을 만든 후 이를 상용 라이선스로 배포하는 것도 가능합니다. 다만, 이러한 경우에 소스코드를 공개할 필요는 없지만, Apache 라이선스 등이 요구하는 기타의 사항(저작권 및 개발자 표시, 수정내용 고지 등)은 준수해야 합니다.

<프로젝트 사례> 안드로이드 플랫폼

아파치 서버 등 아파치재단의 프로젝트들뿐만 아니라, (2008년 조사결과) SourceForge.net의 5,000개 이상의 프로젝트가 Apache 라이선스로 배포되고 있다. 또한, 2008년 5월 구글은 블로그를 통해 Google Code의 100,000여개 프로젝트 중 25%가 Apache 라이선스를 사용하고 있다고 발표했었다. 6

최근 Apache 라이선스로 배포되는 가장 큰 프로젝트는 안드로이드 플랫폼이다. 커널과 애플리케이션을 제외한 안드로이드 플랫폼의 많은 부분은 Apache 라이선스로 배포되고 있다. 하지만 일부 구성요소들은 CPL, EPL, GPL, LGPL 등으로 배포되고 있으며, 이들은 Apache 라이선스가 아닌 각각의 라이선스 규칙을 따라야 한다는 점을 주의할 필요가 있다.

6) 위키피디아 참조. 2010년 9월 Google Code 프로젝트에서는 OSI 인증라이선스 모두를 받아들이는 것으로 변경했다.

<참고> 안드로이드 애플리케이션의 라이선스 이슈

APACHE2 bionic/linker	CPL dalvik/libcore/junit
APACHE2 dalvik	EPL prebuilt/common/eclipse
APACHE2 dalvik/libcore-disabled/instrument	EPL prebuilt/common/osgi
APACHE2 dalvik/libcore-disabled/sound	EPL prebuilt/darwin-x86/swt
APACHE2 dalvik/libcore/annotation	...
APACHE2 dalvik/libcore/archive	GPL external/blktrace
APACHE2 dalvik/libcore/auth	GPL external/e2fsprogs
APACHE2 dalvik/libcore/awt-kernel	GPL external/e2fsprogs/e2fsck
APACHE2 dalvik/libcore/crypto	...
APACHE2 dalvik/libcore/dalvik	LGPL external/e2fsprogs/lib/blkid
APACHE2 dalvik/libcore/logging	LGPL external/e2fsprogs/lib/e2p
APACHE2 dalvik/libcore/luni	...
APACHE2 dalvik/libcore/luni-kernel	MIT external/e2fsprogs/lib/ss
APACHE2 dalvik/libcore/math	OSL1 external/elfutils
...	W3C dalvik/libcore/xml

미들웨어가 Apache 라이선스로 배포되는 경우 애플리케이션의 라이선스에는 영향을 미치지 않는다. 그 결과 애플리케이션의 개발자들은 자유롭게 라이선스를 결정할 수 있다. 유료의 상용 라이선스를 선택하는 것도 물론 가능하다. 구글이 안드로이드 플랫폼에 대한 라이선스를 GPL이 아닌 Apache 라이선스로 결정한 것은 더욱 많은 개발자들의 참여를 끌어내기 위해서이다. 스마트폰의 경쟁력은 얼마나 많은 애플리케이션을 확보하고 있는가에 달려 있다. 구글의 입장에서는 GPL로 대변되는 오픈소스의 정신을 존중하는 것도 중요했지만, 개발자들에게 라이선스를 선택할 수 있는 자유를 주어 보다 많은 애플리케이션이 개발될 수 있도록 하는 것이 더 중요했다고 볼 수 있다.

그런데 애플리케이션에 따라서는 기존의 오픈소스를 기반으로 개발되는 것도 있다. 이러한 경우 해당하는 오픈소스 라이선스를 지켜야 하는 것은 물론이다. 예를 들어 GPL로 배포되는 오픈소스를 기반으로 안드로이드 애플리케이션을 개발하는 경우 GPL에 따라 소스코드를 제공해 주어야 한다. 앱스토어(안드로이드 마켓)를 통해 직접 소스코드를 제공하는 것은 어려우므로 애플리케이션의 일정 부분에 소스코드를 제공한다는 안내를 하고 다른 웹서버를 통해 소스코드를 제공하는 방법을 고려할 수 있다. 이러한 사항을 지키지 않는 경우 라이선스 위반에 해당하여 소송을 당할 수 있다.

이러한 경우 애플리케이션 개발자가 책임을 지는 것과는 별도로, 앱스토어 운영사도 라이선스 위반에 대한 책임이 있는가? 이 문제는 이용자의 저작권 침해에 대해 네이버 등 서비스제공자도 책임을 져야 하는가의 문제와 유사하다. 서비스유형에 따라 다른 판결이 나오고 있지만, 권리자가 권리침해가 있다는 통지를 해오는 경우 서비스제공자는 즉시 관련 저작물을 삭제하는 것으로 대응하고 있다.

2.2. GPL형 라이선스 및 주요 프로젝트

GPL형 라이선스에는 GPL 2.0, GPL 3.0, LGPL 2.1, LGPL 3.0, AGPL 3.0 등이 포함되며, 대부분 FSF(Free Software Foundation)에서 주도하여 만든 것이다. 비교적 오랜 역사를 가진다는 점에서 BSD와 비슷하지만, 카피레프트 조항과 소스코드 제공의무를 가지고 있다는 점에서 큰 차이가 있다. 카피레프트의 적용 범위 및 소스코드 제공의무의 범위는 GPL, LGPL, AGPL 각각에 차이가 있다.

2.2.1. GPL 2.0

GPL 2.0으로 배포되는 오픈소스는 i) 각 복제본에 적절한 저작권 표시와 보증책임이 없음을 명시하고, ii) GPL 라이선스를 언급하는 고지사항과 보증책임 관련 고지사항을 원본 그대로 유지하고, iii) 소프트웨어를 양도받는 모든 이들에게 소프트웨어와 함께 GPL 라이선스 사본을 제공하고, iv) 파일을 수정한 경우 수정했다는 사실과 날짜를 파일에 명기해야 한다. 그리고 v) 원본저작물과 파생저작물(derivative work)을 GPL 2.0에 의해 배포해야 하며, vi) 원본저작물 및 파생저작물에 대한 소스코드를 제공하거나, 요청 시 제공하겠다는 약정서를 제공해야 한다. 여기서 i) ~ iv)의 의무사항은 Apache 라이선스와 동일하거나 유사한 내용이지만, v) 및 vi)의 의무사항은 BSD형의 라이선스에서는 찾아볼 수 없는 내용이다.

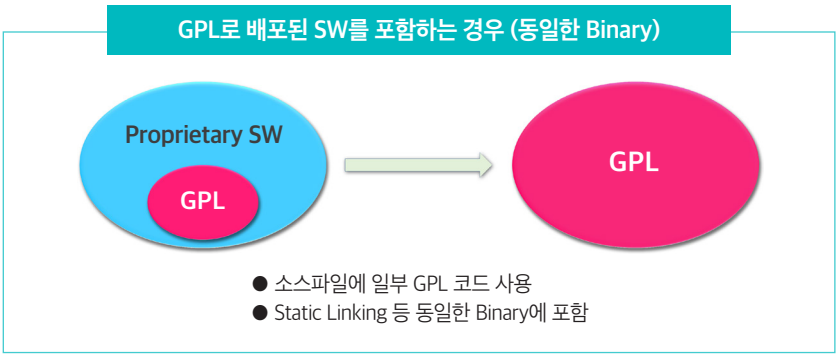
<참고> 파생저작물의 범위에 관한 해석

오픈소스 현장에서는 GPL 2.0에서 언급하고 있는 파생저작물(derivative work)[7]의 범위가 어디까지인지에 관해 논란이 많다. 일반적으로 계약이나 법 조항이 명확하지 않은 경우 당사자의 의도나 거래 관행 등을 종합적으로 고려하여 법원에서 결정하게 된다. GPL도 명확하지 않은 부분이 많으며, 최종적으로는 법원에서 가려지겠지만, GPL을 만든 FSF에서는 GPL의 해석에 도움을 주고자 많은 FAQ를 제공하고 있다.[8]

예를 들면, GPL로 배포된 소프트웨어를 수정하였거나 새로운 소프트웨어에 정적 링크시키는 경우, 즉 두 개의 모듈이 동일한 실행 파일에 포함되어 있을 경우는 해당 실행 파일에 포함된 모든 소스는 GPL이 적용된다.

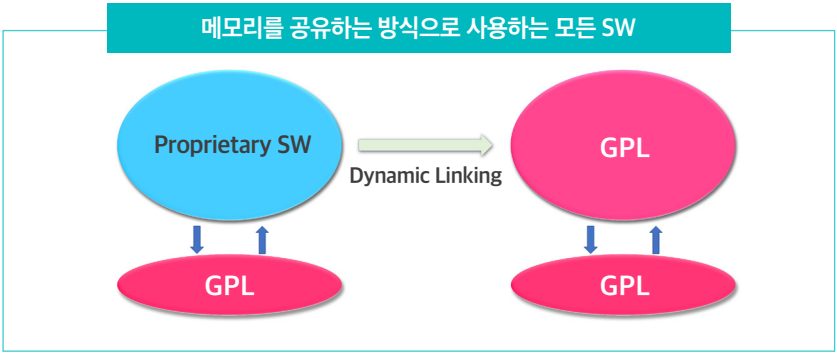
7) GPL 2.0 "... These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it."

8) http://www.fsf.org/licensing/licenses/gpl-faq.html.



※ 바이너리에 GPL 소스코드가 일부라도 포함되어 있으면 바이너리는 GPL이 됨

또한, 동일한 바이너리에 포함되지 않더라도 동적 링크 등의 방식으로 공유주소영역에서 링크되어 실행되도록 설계된 경우, 플러그인이 동적으로 링크되어 함수를 호출하고 데이터구조를 공유하는 경우에도 GPL 소프트웨어와 함께 링크되어 실행되는 소프트웨어에도 GPL이 적용되어 소스코드를 제공해야 한다.



※ 원 저작자가 특별히 기술한 예외사항이 없는 한 모두 GLP화가 됨

반면, 두 개의 프로그램이 파이프(pipes), 소켓(sockets), command-line arguments 형태로 통신하는 경우, 플러그인이 fork나 exec을 이용하는 경우 등은 별도의 저작물로서 GPL이 적용되지 않는다고 답변하고 있다.
그런데, FSF의 이와 같은 해석이 저작권법상의 파생저작물 또는 2차적저작물의 범위를 벗어난다는 주장도 일부 있다. GPL 3.0에서는 이와 같은 비판을 고려하여 “derivative works”라는 용어를 삭제하고 “Work based on the program”으로 통일하고 있다.

<프로젝트 사례> 리눅스 커널

GPL 2.0으로 배포되는 대표적인 프로젝트는 리눅스 커널이다. 리누스 토발즈가 만들어 GPL 2.0으로 배포한 이후 많은 개인 및 기업들의 기여 때문에 진화됐다.

리눅스 커널 소스코드의 루트 디렉토리에 있는 'CREDITS'라는 파일에는 리눅스 프로젝트에 기여한 개인들의 이름, 주소, ID, 이력 등이 나타나 있다. GPL 2.0에 따르면 저자에 관한 이름 등은 함부로 변경하거나 삭제할 수 없다. 따라서 리눅스 커널의 소스코드를 제공할 때는 CREDITS 파일이 빠지지 않도록 신경 써야 한다.

<CREDIT 파일의 내용>

This is at least a partial credits-file of people that have contributed to the Linux project. It is sorted by name and formatted to allow easy grepping and beautification by scripts. The fields are: name (N), email (E), web-address (W), PGP key ID and fingerprint (P), description (D), and snail-mail address (S).
Thanks, Linus

N: Matti Aarnio
E: mea@nic.funet.fi
D: Alpha systems hacking, IPv6 and other network related stuff
D: One of assisting postmasters for vger.kernel.org's lists
S: (ask for current address)
S: Finland

CREDITS와 같은 디렉토리에 있는 COPYING 파일에는 리눅스 커널이 GPL 2.0에 의해 배포되고 있다는 것과 리눅스의 정상적인 시스템 콜에 의해 커널 서비스를 이용하는 'user programs'에는 저작권이 적용되지 않는다는 리누스 토발즈의 글이 포함되어 있다. 'user programs'이 GPL의 적용대상에서 제외된 것처럼, 커널 모듈도 GPL의 적용대상에서 제외되는가에 대해 많은 논란이 있다. 오픈소스 커뮤니티의 일부 개발자 및 기업들은 커널 모듈도 GPL의 적용대상이 되지 않는 것으로 해석하기도 하지만, 적지 않은 수의 기여자, 즉 커널에 대한 권리자들이 이러한 해석에 반대하고 있다. 오히려 저작권법의 관점에서 보면 일부의 권리자들이 소송을 제기하는 것도 가능하므로, 각 기업의 입장에서는 보수적인 판단을 하는 것이 낫다.

<COPYING 파일의 내용>

NOTE! This copyright does *not* cover user programs that use kernel services by normal system calls ··· Also ··· but the instance of code that it refers to (the Linux kernel) is copyrighted by me and others who actually wrote it.

Also note that the only valid version of the GPL as far as the kernel is concerned is _ this_ particular version of the license (ie v2, not v2.2 or v3.x or whatever), unless explicitly otherwise stated.

Linus Torvalds

각 기업에서 GPL의 적용을 받지 않는 커널 모듈이라고 주장하기 위해 최소한 지켜야 하는 기준은 다음과 같다.

첫째, 커널 모듈 라이선스를 GPL로 선언하지 않아야 한다. 즉, MODULE_LICENSE ("GPL")로 선언된 매크로가 존재해서는 안 된다. 이는 커널 모듈에서 GPL로 export 된 커널 함수("EXPORT SYMBOL() 이 아닌 "EXPORT_SYMBOLE_GPL()로 export 된, 이하 GPL-ONLY Symbol)[9]를 사용하지 않았음을 의미한다.

(MODULE_LICENSE("GPL") 매크로가 선언되지 않은 상태에서 GPL-ONLY Symbol을 사용할 경우 모듈 빌드시 error message를 표시하며 빌드되지 않는다.)

둘째, Linux-Only Driver가 아니어야 한다. 이는 리누스 토발즈의 의견에 따라 Linux 외 다른 OS 상에서 동작하는 드라이버를 리눅스 상에 포팅한 것이라면 리눅스(GPL)의 파생저작물로 보지 않을 수 있다. 하지만 리눅스 상에서만 동작하는 드라이버라면 리눅스의 파생저작물이 아니라는 주장의 설득력이 떨어진다.

셋째, 모듈 내 리눅스 커널 소스의 일부가 전혀 사용되지 않았을 경우여야 한다. 리눅스의 표준 커널 모듈 인터페이스만을 사용해 모듈을 제작하였다면 문제없지만, 커널 소스가 모듈 내 일부라도 포함되었다면 리눅스의 파생저작물로 판단할 수밖에 없다.

한편, 대부분의 GPL 오픈소스가 FSF의 권유에 따라 "either version 2 of the License, or any later version"에 의해 배포되는 반면, 리눅스 커널은 2.0에 의해서만 배포된다. 그 결과 기여자 중 일부가 반대한다면 리눅스 커널이 GPL 3.0으로 전환하는 것은 불가능하다고 볼 수 있다.

9) Kernel 개발자들이 GPL-ONLY Symbol의 비율을 높여가고 있기 때문에 GPL-ONLY Symbol의 수는 커널버전이 올라갈수록 증가하고 있다 : 2.5.75(5개) --> 2.6.10(400여개) --> 2.6.18(1000여개) --> 3.0 (500여개) --> 3.8 (7,000여개) --> 3.18 (10,000여개) --> 4.4 (11,000여개) --> 4.8 (12,000여개)

<상담 사례> 리눅스 커널 모듈의 소스코드 제공여부

Q: 리눅스 커널의 경우 모듈로 제공되는 드라이버에 대한 공개 여부가 논란이 있는 것 같은데 어떻게 대응해야 하는가요?

A: GPL 2.0에 대한 FSF의 해석에 따르면, 커널과 동적 링크된 모듈은 파생저작물에 해당하기 때문에 원칙적으로 소스코드를 제공해야 합니다. 그런데, "정상적인 시스템 콜에 의해 커널 서비스를 이용하는 User Program은 GPL 2.0의 적용을 받지 않는다"는 리눅스 커널 라이선스 문구를 인용하면서, 커널 모듈 표준 인터페이스를 사용할 경우에는 예외적으로 GPL이 적용되지 않는다는 견해도 있습니다. 실제로 일부 기업들은 디바이스 드라이버에 대한 소스코드를 제공하지 않고 있습니다.

리눅스 커널 모듈에 GPL이 적용되는지에 대한 문제는 아직 명확한 해석이나 판결이 없습니다. 최소한 앞에서 설명한 세 가지를 만족하지 못한다면 커널 모듈은 GPL의 적용을 받는 것이 명백하고, 만족한다 하더라도 이는 법원의 명확한 판결이 아니므로, 직접적인 소송에서는 다른 판단이 나올 수도 있습니다. 따라서 각 기업에서는 소스코드를 제공하든지, 아니면 법적 리스크를 감수하면서 자사의 영업비밀로 보호하는 전략을 취하는 등의 전략적인 판단을 해야 합니다.

<참고> 소스코드의 제공범위

제공할 소스코드의 범위에 관하여 GPL 2.0은 "all the source code for all modules it contains, any associated interface definition files, the scripts used to control compilation and installation of the executable"이라고 규정하고 있다. 따라서, 소스코드를 컴파일해서 바이너리를 생성하기 위한 build script를 제공할 필요가 있으며, 생성된 바이너리를 제품에 설치하기 위한 install script를 제공해야 한다. (단, 사용자가 생성한 바이너리를 제품에 설치 시 제품이 정상 동작하지 않을 경우는 각 기업의 전략적으로 판단하여 install script를 제공하지 않을 수 있다) 그리고 소스를 컴파일하기 위한 tool chain은 사용자가 인터넷을 통해 쉽게 얻을 방법만 명시한다면 직접 제공하지 않아도 무방하지만, 그렇지 못할 경우는 제공해야 하는 것으로 해석할 수 있다. 하지만, GPL 2.0에서 언급하고 있는 "실행파일을 컴파일하고 설치하기 위한 scripts"가 정확히 어디까지인지는 여전히 불명확하다. 예를 들면, 소스를 컴파일해서 실행 파일을 만들기 위해서 tool chain도 제공해주어야 하는지, 실행 파일을 하드웨어에 설치하기 위해 설치 방법까지 알려주어야 하는지 등은 여전히 명확하지 않는 것으로 볼 수 있다. 이용자들이 자유롭게 수정하여 이용할 수 있도록 한다는 GPL의 취지를 고려한다면 모두 포함해야 한다는 것이 맞고, FSF도 대체로 그렇게 해석하고 있다. 하지만 GPL이 명확하게 규정하고 있지 않다는 점, 현실적으로 보안등 다양한 이유로 기업들이 제공하기가 쉽지 않다는 점 등을 고려하면 다른 해석도 가능하다. 최종적으로는 법원이 판단해야겠지만, 그 전까지는 각 기업이 전략적으로 판단할 수밖에 없다.

<상담 사례> GPL 바이너리를 수정하지 않고 재배포 시 의무사항

Q: GPL로 공개된 오픈소스 패키지의 바이너리를 수정하지 않고 제품에 탑재하여 배포하였습니다. 이 경우에도 소스 코드를 공개해야 하나요?

A: FSF FAQ에 따르면 오픈소스를 바이너리 형태로 다운받아서 수정하지 않고 그대로 제품에 사용하여 배포하는 경우에도 GPL 요구 사항에 따라 소스 코드를 함께 배포해야 합니다.[10]

또한 바이너리를 다운로드 받을 수 있는 website link만 제공하거나, standard version 대비 수정한 부분의 소스 코드만을 제공하는 것은 GPL 요구사항을 만족시킬 수 없습니다. [11]

<상담 사례> GPL 바이너리를 수정하지 않고 재배포 시 의무사항

Q: GPL인 헤더파일을 include하여 자사 소프트웨어에 사용할 경우, 자사 소스 코드도 공개해야 하나요?

A: GPL 헤더파일 내 소스 코드를 복사하여 가져오거나, include하여 사용하는 소프트웨어도 GPL의 파생저작물이 되어 소스 공개 등 GPL 의무 사항 준수가 필요합니다. 그러나 FSF를 이끄는 리차드 스톨만은 리눅스 커널 메일링 리스트[12]를 통해 GPL 헤더파일 부분이 단지 structure definitions, typedefs, enumeration constants, 단순 macros 등 이라면 이를 사용하는 소프트웨어를 GPL의 파생저작물로 보지 않아도 된다고 언급하였습니다.

또한 참고로 Linux의 User Space 프로그램이 일반적인 시스템콜을 이용하기 위한 목적으로 Linux Kernel 헤더 파일을 include하는 경우에는 이 User Space 프로그램을 Linux Kernel의 파생저작물로 간주하지 않습니다.[13]

<상담 사례> 리눅스 커널 모듈을 BSD OS에 사용하는 경우

Q: 제 연구실에서 리눅스 커널 모듈의 형태로 개발한 소프트웨어를 미국에 있는 협력 대학에서 BSD 라이선스의 OS에 활용하려고 합니다. 미국 대학에서 개발된 소프트웨어는 현재 저희의 소프트웨어 없이도 동작 가능한 별도의 독립 소프트웨어 패키지인데, 저희의 소프트웨어를 사용하여 성능 향상을 꾀하고 있습니다. 협력 대학에서 (저희 소프트웨어에는 수정을 가하지 않은) 통합된 소프트웨어 패키지를 BSD 라이선스로 배포하고자 합니다. 저희가 만든 리눅스 커널 모듈을 GPL이 아닌 BSD 라이선스로 배포될 때 GPL을 위반하는 것은 아닌가요?

A: 경우를 나누어 보겠습니다. 첫째, 연구실에서 해당 모듈 전체를 자체 개발한 경우에는, 리눅스 커널과 함께 배포하는 경우라면 리눅스 커널의 라이선스인 GPL의 영향을 받겠지만, 리눅스 커널을 배포하지 않고 모듈만을 배포하는 경우에는 리눅스 커널의 라이선스와 상관없이 커널 모듈을 별도의 라이선스로 배포하는 것이 가능합니다.

10) http://www.gnu.org/licenses/gpl-faq.en.html#UnchangedJustBinary
11) https://www.gnu.org/licenses/gpl-faq.en.html#DistributingSourceIsInconvenient
12) http://www.gnu.org/licenses/gpl-faq.en.html#UnchangedJustBinary
13) https://www.gnu.org/licenses/gpl-faq.en.html#DistributingSourceIsInconvenient

둘째, 연구실에서 개발한 커널 모듈이 기존의 커널모듈을 수정하여 작성한 경우라면, 기존의 커널 모듈이 어떠한 라이선스에 의해 배포되었는지에 따라 다릅니다. 만약 GPL에 의해 배포된 것이라면 수정된 커널 모듈도 GPL에 의해 배포해야 합니다.

결론적으로, 커널 모듈 전체를 별도로 작성한 경우라면 미국 대학 측에 GPL이 아닌 BSD 라이선스로 허락해 줄 수 있습니다. 그런데, BSD로 허락해 줄 경우, 다른 사람들도 BSD로 사용 가능하기 때문에, 차후 수익모델 등의 창출이 어려움이 있습니다. 이 경우, 커널 모듈을 GPL로 배포하되, BSD와 결합 가능하다는 예외조항을 설정하는 형태로 할 수 있습니다.

<참고> 소스코드를 제공해야 하는 대상

GPL 오픈소스를 사용하는 소프트웨어를 바이너리 형태로 배포할 경우 소스코드도 함께 제공해야 한다면, 과연 누구에게 소스코드를 제공해야 하는가를 분명하게 인지할 필요가 있다. 우선 SI 업체에서와 같이 해당 소프트웨어의 바이너리를 제공받는 측에 소스코드를 함께 제공한다면 그 순간 소스코드 제공에 대한 의무사항은 모두 만족하게 된 것이며, 향후 제3자의 소스코드 제공 요청에 대해서 응답해주어야 하는 의무는 없다. 그러나, 임베디드 시스템과 같이 불특정 다수에게 바이너리가 포함된 제품을 배포하는 상황에서 해당 제품의 저장 공간의 제약 등으로 소스코드를 함께 배포할 수 없는 경우에는 소스코드를 제공하겠다는 약정서(Written Offer)를 제품과 함께 제공하는 방법을 채택할 수 있다. 다만, 이 경우에는 출시된 제품에 대해 최소 3년간 소스코드를 요청하는 누구에게든지 완전한 소스코드를 제공해야 하는 의무가 있다. 그러므로 각 기업에서는 소스코드를 제공하는 방법에 대해서도 신중한 선택을 해야 한다.

The software included in this product contains copyrighted software that is licensed under the GPL. A copy of that license is included in this document on page X. You may obtain the complete Corresponding Source code from us for a period of three years after our last shipment of this product, which will be no earlier than 2011-08-01, by sending a money order or check for $5 to:
GPL Compliance Division
Our Company
Any Town, US 99999

Please write "source for product Y " in the memo line of your payment.
You may also find a copy of the source at http://www.example.com/sources/Y/.
This offer is valid to anyone in receipt of this information.

<A Practical Guide to GPL Compliance에서 제시하는 Written Offer 예제, SFLC>

<참고> 오픈소스 라이선스의 양립성(Compatibility)

둘 이상의 오픈소스의 소스코드를 사용하여 새로운 프로그램을 개발할 경우, 각 오픈소스의 라이선스 요구사항이 서로 상충하는가의 문제를 양립성(Compatibility) 문제라고 한다.보통 각 라이선스의 요구사항을 모두 충족시킬 수 없는 경우 이러한 문제가 발생하며, 라이선스의 요구사항을 충족시키지 못하는 경우는 라이선스 위반에 해당하기 때문에 둘 이상의 오픈소스를 결합할 때에는 각 라이선스가 서로 양립 가능한지 미리 조사해야 한다. 예를 들면, 각각 GPL과 MPL로 배포되는 오픈소스의 소스코드를 함께 결합해 사용하는 경우,MPL은 MPL로 배포된 오픈소스에 해당하는 소스코드로부터 파생된 부분을 동일한 MPL로 배포하기를 요구하고,GPL은 GPL로 배포된 오픈소스에 해당하는 소스코드로부터 파생된 부분뿐만 아니라, 그와 링크된 전체의 소스코드를 GPL로 배포하기를 요구하기 때문에 이 두 라이선스의 요구를 동시에 충족시키는 것은 불가능하다.

특히 GPL과 관련하여 양립성 문제가 많이 발생하는 이유는, GPL은 사용자들에게 GPL에서 규정하고 있는 것 이외의 제한사항을 추가하지 못하도록 엄격하게 통제하고 있기 때문이다.[14] 예를 들면, 많은 오픈소스 라이선스가 특허 관련 조항, 준거법(choice of law) 등과 관련하여 GPL과 다른 내용을 포함하고 있는데, GPL의 입장에서는 이런 내용이 모두 이용자들에게 추가적인 제한을 가하는 조항들로 해석된다. 반대로, GPL과 LGPL은 코드 공개 범위는 다르지만, 다른 내용은 같기 때문에 라이선스를 변경하거나(예를 들어 LGPL을 GPL로 Relicensing) 라이선스를 변경하지 않더라도 공개범위가 더 큰 라이선스인 GPL로 배포한다면 양립 가능한 것으로 볼 수 있다. GPL을 포함하여 FSF가 만든 라이선스 간의 양립성에 대해서는 GNU 사이트에서 확인할 수 있으며,[15] 아울러 FSF는 GPL과 양립 가능한 오픈소스 라이선스의 목록도 제공하고 있다.[16]

14) GPL 2.0. "[any distributed work] that ... contains or is derived from the [GPL-licensed] Program ... be licensed as a whole ... under the terms of [the GPL].", and that the distributor not "impose any further restrictions in the recipients' exercise of the rights granted"
15) http://www.gnu.org/licenses/gpl-faq.html#AllCompatibility
16) http://www.gnu.org/licenses/license-list.html

<GNU GPL 계열 라이선스 간 양립성>

		배포하고 싶은 프로젝트:					
		GPLv2 only	GPLv2 혹은 이후 버전	GPLv3 혹은 이후 버전	LGPLv2.1 only	LGPLv2.1 혹은 이후 버전	LGPLv3 혹은 이후 버전
Code 형태로 사용:	GPLv2 only	가능	가능	불가	GPLv2로 변환할 경우 가능	GPLv2로 변환할 경우 가능	불가
	GPLv2 혹은 이후 버전	가능	가능	가능	GPL로 변환할 경우 가능	GPL로 변환할 경우 가능	GPLv3로 변환할 경우 가능
	GPLv3	불가	GPLv3로 변환할 경우 가능	가능	GPLv3로 변환할 경우 가능	GPLv3로 변환할 경우 가능	GPLv3로 변환할 경우 가능
	LGPLv2.1 only	GPLv2로 변환할 경우 가능	GPL로 변환할 경우 가능	GPLv3로 변환할 경우 가능	가능	가능	GPLv3로 변환할 경우 가능
	LGPLv2.1 혹은 이후 버전	GPLv2로 변환할 경우 가능	GPL로 변환할 경우 가능	GPLv3로 변환할 경우 가능	가능	가능	가능
	LGPLv3	불가	GPLv3로 변환하거나 업그레이드할 경우 가능	GPLv3로 변환할 경우 가능	GPLv3로 변환할 경우 가능	LGPLv3로 업그레이드할 경우 가능	가능
Use a Library 형태로 사용:	GPLv2 only	가능	가능	불가	GPLv2로 변환할 경우 가능	GPLv2로 변환할 경우 가능	불가
	GPLv2 혹은 이후 버전	가능	가능	가능	GPL로 변환할 경우 가능	GPL로 변환할 경우 가능	GPLv3로 변환할 경우 가능
	GPLv3	불가	GPLv3로 업그레이드할 경우 가능	가능	GPLv3로 변환할 경우 가능	GPLv3로 변환할 경우 가능	GPLv3로 변환할 경우 가능
	LGPLv2.1 only	가능	가능	가능	가능	가능	가능
	LGPLv2.1 혹은 이후 버전	가능	가능	가능	가능	가능	가능
	LGPLv3	불가	가능	가능	가능	가능	가능

하지만, 대부분의 라이선스는 다른 라이선스와의 양립성 정보를 일일이 제공하고 있지는 않기 때문에,각 라이선스의 양립 여부를 판단하기 위해서는, 라이선스나 FAQ를 참조하거나 라이선스 관계자나 권리자에게 연락을 취하여야 한다. 예를 들면, EPL의 경우에는 제3자의 소스코드와 결합하는 문제에 대해서, Eclipse 재단의 Legal Advisor(license@eclipse.org)에게 연락을 취하도록 권장하고 있다.

<참고> 라이선스 충돌 해결방안

라이선스 충돌	해결방안
Proprietary 라이선스와 오픈소스 라이선스간의 충돌	-파생저작물의 범위가 중복되지 않도록 분리설계 -라이선스 충돌이 발생하지 않는 다른 라이선스 SW로의 대체 (예: 오픈소스 SW의 상용 버전 존재시 상용버전으로 대체) -저작권을 가지고 있는 자사 Proprietary SW의 라이선스 변경 (예: Proprietary 라이선스 --> 오픈소스 라이선스) -라이선스 충돌이 발생하지 않도록 오픈소스 SW의 자체개발
오픈소스 라이선스간의 충돌	- 파생저작물의 범위가 중복되지 않도록 분리설계 - 라이선스 충돌이 발생하지 않는 다른 라이선스 SW로의 대체

<상담 사례> GPL과 EPL의 양립성

Q: GPL과 EPL간에 라이선스 충돌발생 이유는 코드공개 범위가 상이하기 때문인지요? 아니면, 동일한 코드에 대해 GPL은 GPL로 배포해야 하고, EPL은 EPL로 배포해야 하기 때문에 충돌이 발생하는 것인지요? 만약 후자의 경우라면 코드공개 의무가 있는 오픈소스 라이선스는 GPL SW와 파생 저작물이 겹칠 경우 모두 GPL과 충돌이 발생한다고 봐도 될지요?

A: 단순히 코드공개 범위가 상이하다고 해서 충돌이 발생하는 것은 아닙니다. GPL과 LGPL은 코드 공개범위는 다르지만, 다른 내용들은 동일하기 때문에 양립가능한 것만 보아도 알 수 있지요. EPL은 특허관련조항, 준거법(choice of law) 등에 대해서도 규정하고 있는데, GPL의 입장에서는 이런 내용들이 모두 라이선시들에게 추가적인 제한을 가하는 조항들로 해석될 수 있기 때문에 충돌하는 것입니다.

GPL 2.0	EPL 1.0
licensor는 licensee의 권리 행사를 제한할 수 있는 어떠한 사항도 임의대로 추가할 수 없음(제6조). Licensor가 특허권 보유시 특허권 행사할 수 없음을 묵시적으로 규정하고 있으나, Licensee가 특허권 보유시 규정이 없음.(제7조)	Licensor가 특허권 보유시, Royalty free로 특허권을 행사할 수 없음(제2조). Licensee가 본 Program에 대한 특허권 보유시, 해당 특허권으로 특허권을 행사하면 행사한 날로부터 License가 종료됨(제7조). EPL은 뉴욕주의 법률과 미국 지식재산권법의 적용을 받음.

- 충돌사유
  - GPL은 licensee의 권리를 제한하는 것을 못하도록 하였으나, EPL은 2조 및 7조의 제한처럼 GPL에 없는 제한이 포함되어 있으므로 양립불가
  - 특정 프로그램(SW)이 GPL 2.0과 EPL 1.0으로 배포되는 경우, 본 프로그램을 활용하는 자가 특허권 보유시, GPL 2.0 특허권을 행사를 할 수 있으나, EPL 1.0으로는 행사가 불가함으로 여기에서 충돌이 발생

한편, 양립성 문제는 각각의 라이선스에 대한 내용뿐만 아니라 소프트웨어의 결합방식에 따라서도 다르게 발생합니다. 예를 들면, GPL의 경우에는 링크를 통해 결합한 코드도 파생저작물로 보고 모두 GPL 라이선스로 배포할 것을 요구하기 때문에 충돌문제가 발생하지만, EPL 등 약한 카피레프트(weak copyleft) 조항을 가진 라이선스들은 대부분 오브젝트 또는 실행파일들을 commercial 라이선스를 포함한 다른 라이선스로 배포할 수 있도록 허용하고 있습니다. 소스코드만 EPL 등으로 배포하면 됩니다.

<상담 사례> 약한(Weak) 카피레프트 라이선스들간의 양립성

Q: LGPL, EPL, Apache, BSD로 배포되고 있는 OSS library들을 수정을 하지 않는 조건에서 하나의 응용프로그램에서 링크시켜(import 시켜) 써도 라이선스 양립성 위반에 해당하지 않습니까?

A: LGPL, EPL 등의 라이선스들은 카피레프트 조항을 가지고 있고 라이선스의 내용도 다르기 때문에 각각의 라이선스로 배포되는 코드들을 결합하는 경우 양립성 문제가 발생할 수도 있습니다. 그런데, GPL과는 달리 약한 카피레프트 조항을 가지고 있는 라이선스들은, 링크를 통해 연결하는 별도의 코드(모듈)에 대해서는 영향을 미치지 않습니다. 따라서 질문과 같이 각각의 라이선스로 배포되는 라이브러리들을 수정도 하지 않고 단순히 링크만 시켜 사용하는 경우에는 양립성 문제가 없는 것으로 보여집니다. 다만, 개별적인 라이선스 준수 의무, 즉 라이선스 사용 여부 명시, 라이선스 사본 첨부 및 저작권 고지사항 포함, 소스코드의 제공(LGPL, EPL의 경우) 등의 사항들은 지켜야 함은 물론입니다.

<상담 사례> GPL인 OOO 오픈소스 패키지 내 GPL과 양립할 수 없는 License의 파일이 존재

Q: GPL 2.0인 OOO 오픈소스 패키지 내에 APSL 1.2의 소스 코드가 포함된 것을 확인하였습니다. APSL 1.2는 GPL 2.0과 양립할 수 없는 것으로 아는데, 이 오픈소스 패키지를 사용해도 되는지 궁금합니다.

A: GPL과 양립할 수 없는 라이선스의 소프트웨어를 GPL 프로그램에 포함해서 배포하는 것은 두 라이선스의 의무 사항을 동시에 충족할 수 없는 문제를 유발합니다. 먼저, APSL 1.2의 소스코드가 GPL인 OOO 오픈소스와는 별도로 테스트 등을 위해 추가된 파일은 아닌지 확인하시기 바랍니다. 이 경우에는 GPL 프로그램을 배포할 때 테스트용 파일은 제외할 수 있으므로 라이선스 양립성 문제를 회피할 수 있습니다. 하지만, 실제로 OOO 오픈소스의 소스 코드를 컴파일하여 프로그램을 생성할 때, APSL 1.2의 소스 코드도 포함되는 구조라면, OOO 오픈소스의 저작권자에게 연락하여 라이선스 양립성 문제를 제기하고, APSL 1.2의 소스 코드를 GPL과 호환될

수 있는 라이선스로의 변경을 요청하여 양립성 문제를 해결하는 것이 OOO 오픈소스를 라이선스 문제없이 사용할 수 있는 방법일 것입니다.

2.2.2. GPL 3.0

GPL 3.0은 GPL 2.0이 배포되고 난 이후 오픈소스 환경을 둘러싼 다양한 변화들을 수용하여 만든 라이선스이다. GPL 3.0의 의무사항으로는 i) 각 복제본에 저작권 고지와 보증책임이 없음을 명시할 것, ii) GPL 3.0의 조건 및 제7조의 조건에 관한 내용을 있는 그대로 유지할 것, iii) 소프트웨어를 양도받는 모든 이들에게 소프트웨어와 함께 GPL 라이선스 사본을 제공할 것, iv) 소프트웨어를 수정했을 경우 수정사실 및 일시를 명시할 것, v) 원본저작물과 그에 기반한 저작물(Work based on the program)을 GPL 3.0에 의해 배포할 것, vi) 원본저작물 및 그에 기반한 저작물(Work based on the program)에 대한 소스코드를 제공하거나, 요청 시 제공하겠다는 약정서를 제공할 것, vii) 사용자 제품(user product)에 대한 인증키 등 설치정보(installation information)를 제공할 것, viii) 차별적인 특허라이선스 계약을 체결하지 말 것 등의 내용이 포함되어 있다.
i) ~ vi) 의 내용은 GPL 2.0의 내용과 같거나 내용을 더욱 명확히 하는 것이었지만, vii) 및 viii)의 내용은 GPL 3.0에 처음으로 포함된 내용으로 많은 논란을 불러일으켰다.

<참고> Tivoization과 설치정보의 제공

미국의 Tivo 라는 회사는 케이블방송 또는 위성방송 등의 방송프로그램을 하드디스크에 녹화하거나, 경우에 따라서는 개인 PC나 DVD 등에 저장할 수 있도록 하는 DVR(digital video recorder)제품을 시장에 출시하여 성공하였다. 한편 해당 제품에는 리눅스 커널 등 GPL 소프트웨어가 포함되어 있었기 때문에 Tivo는 GPL 2.0의 규정에 따라 관련 소스코드를 이용자들에게 제공하였다. 그런데 이용자들이 해당 소스코드를 수정하여 다시 Tivo의 제품에 포팅하였으나 정상적으로 작동하지 않았다. 이는 Tivo사가 이용자들이 해당 제품을 수정하여 사용하는 것을 허용하지 않았기 때문이다. 이와 같은 Tivo의 정책에 대해 스톨만은 Tivo가 GPL을 악용하고 있다고 주장하였다. GPL의 근본 목적은 이용자들이 해당 소프트웨어를 자유롭게 수정하여 사용할 수 있도록 하는 것임에도 불구하고 Tivo는 이를 막고 있다는 것이다. 스톨만은 이와 같은 GPL의 남용행위를 Tivoization이라고 이름 짓고, 이를 방지하기 위해 GPL 3.0에 다음과 같은 규정을 마련하였다.

GPL 코드를 특정한 사용자 제품(User Product)에 포함하거나 혹은 그와 함께 배포하는 경우에는 해당 소스에 설치 정보(Installation Information)를 함께 제공해야 한다. 이때 '사용자 제품(User Product)'이란 통상적인 소비자 제품이나 집안에서 쓰일 목적으로 설계되었거나 판매되는 제품을 말하며, '설치 정보(Installation Information)'란 소프트웨어를 수정하여 해당 제품에

설치하고 실행하는 데 필요한 방법(methods), 절차(procedures), 인증키(authorization keys) 혹은 여타 정보 모두를 의미한다. 다만 소프트웨어가 ROM에 설치된 경우처럼, 해당 제품의 제조업체나 여타 제3자도 수정된 코드를 제품에 설치할 수 없는 경우에는 설치정보를 제공하지 않아도 된다. 또한, 사용자가 설치하거나 수정한 저작물 및 해당 제품에 대한 지원서비스나 보증, 업데이트를 계속해서 제공할 필요는 없다. 그리고 사용자의 수정으로 인해 네트워크의 작동에 실질적이고 부정적인 영향을 끼치거나, 네트워크를 통한 통신 규칙 및 프로토콜을 위반하는 경우에는 네트워크에 대한 접속을 거부할 수 있다.

<상담 사례> Secure Boot가 적용된 제품의 인스톨 정보

Q: Secure Boot 기능 적용제품의 GPL 의무사항 준수 방법에 대해 문의 드립니다. GPL의 의무사항을 준수하려면 Secure Boot 기능이 적용된 Embedded Linux 탑재제품의 경우 Linux에 대한 소스코드 이외 Build Script와 Install Script까지 제공해야 하는 걸까요? 실제 제공을 하더라도 Install script로는 install이 되지 않는데도 말입니다. 아니면 소스코드와 Build Script까지만 제공하면 되는 것일까요?

A: Secure Boot는 Intel社에서 개발한 특허 기술로, 보안이나 안정성을 위해 특정 제품의 플래시 메모리에 포함된 펌웨어를 인가되지 않은 것으로 변경하는 것을 방지하는 기술입니다.Secure Boot 장치는 '저장장치' 및 '비밀키로 암호화된 실행코드'와 결합되어 있고,호스트 프로세서는 Secure Boot에 포함된 것과 동일한 비밀키로 실행코드를 생성하여, Secure Boot 장치 내의 실행코드와 대응될 경우에만 이를 실행하는 방식으로 펌웨어의 변경을 제한합니다.
TiVo社 또한 Secure Boot와 동일한 목적으로 펌웨어의 변경을 제한하였던 사례 중 하나로 알려져 있는데,TiVo社는 자사에서 판매하는 DVR(Digital Video Recorder) 장치가 동작하기 위한 펌웨어를 리눅스 커널에 기반을 두어 개발하였고,리눅스 커널이 GPLv2를 따르는 오픈소스SW이기 때문에, TiVo社는 GPLv2의 요구사항에 따라 제품 판매시 소스코드를 제공하였지만,TiVo社의 DVR 장치는 전자서명 기술을 통해 사용자가 수정한 펌웨어를 동작시킬 수 없게 제한해두었기 때문에, 사용자들은 제공받은 소스코드를 수정하여 제작한 펌웨어를 동일한 DVR 장치에 사용할 수 없었습니다.
GNU 프로젝트를 지원하고 있는 FSF 측에서는 TiVo社가 비록 GPLv2의 요구사항은 모두 준수하고 있지만, Free Software Definition에서 소개하고 있는 자유 중 하나인,"원 버전을 대신하여 수정된 버전을 사용할 수 있는 자유"를 제대로 보장하고 있지 못하고 있음을 지적하였습니다. 아울러 GPLv3에서는 "Tivoization"이나 "Secure Boot"에 대항하여 수정된 펌웨어를 사용할 수 있는 자유를 보장하기 위한 조항을 새로이 추가하였습니다.
반면에, Linux Kernel의 주 개발자인 Linus Torvalds는 전자서명 기술을 이용한 TiVo社의 펌웨어 제한에 대해 보안측면에서 유익하며, 수정한 펌웨어는 다른 HW를 통해 구동시킬 수 있기 때문에 큰 문제가 되지 않는다고 주장하였습니다.Linux Kernel 또한 계속해서 GPLv2를 유지하고 있습니다.
위와 같은 내용을 종합해 볼 때, GPLv3로 배포된 오픈소스SW를 사용하는 경우에는,Secure Boot가 적용되어 있더라도, GPLv3의 Tivoization 관련 요구사항을 만족하기 위해서는 소스코드와 설치정보(Installation Information) 까지 함께 제공하여,수정된 펌웨어를 사용할 수 있는 자유를 가능한 보장해야 하는 것으로 판단됩니다.
한편, GPL v2에도 "compilation과 installation을 위한 script"가 분명히 언급이 되어 있기 때문에

install 정보 미제공에 대해서는 문제가 될 수 있다고 판단됩니다. TiVo의 사례를 보고 GPL v3가 나온 건 알고 있었지만, 그렇다고 GPL v2 에서는 install 정보제공의무가 없다고 단정하기는 어렵습니다. TiVo의 이슈가 이후 더 이상 진행되지 않은 것뿐이지 이슈가 여전히 잠재되어 있다고 봅니다.

<상담 사례> OSS로 배포하는 LMS의 라이선스 선택

Q: 회사에서 LMS(학습관리시스템) 오픈소스SW 패키지 ABC를 개발하였고, 최근 커뮤니티를 통해 소스를 공개했습니다. ABC 패키지는 GPL 3.0 라이선스로 배포합니다. ABC는 제로보드 같은 모듈형 시스템이기 때문에 모듈추가가 자유롭습니다. (단, 모듈은 core 와 메모리를 공유함, 예를 들면 토론을 위한 모듈을 추가할 수 있음) ABC 패키지는 GPL 3.0 라이선스를 적용시키고, 외부 개발자들이 추가로 개발하는 모듈은 저작자가 개별적으로 라이선스를 지정하여 상업용으로 판매하도록 하는 것이 가능한가요? 그리고, ABC 패키지를 이용해 커스터마이징 사업을 하는 업체에서 모듈을 추가로 개발하고 고객사에 비용을 받으면서도 추가한 모듈에 대한 소스코드를 공개하지 않을 수 있나요?

A: 개발회사가 ABC 패키지에 대한 모든 권리를 가지고 있다면 어떠한 형태로의 라이선스 적용이 가능합니다. 따라서, ABC를 GPL 3.0에 의해서 배포하면서 추가 개발한 모듈에 대해서는 별도의 라이선스 적용이 가능하다는 예외를 설정할 수 있습니다. 그누보드[69]의 core는 GPL 2.0로 배포하고 스킨이나 플로그인은 저작자마다 개별적인 라이선스를 적용할 수 있도록 한 것과 비슷합니다. 또는 처음부터 ABC 패키지를 LGPL로 배포하여 ABC 자체를 수정한 것은 소스코드를 공개하도록 하고, 별도의 라이브러리형태로 추가된 모듈에 대해서는 개별적인 라이선스 적용이 가능하도록 할 수 있습니다. 제로보드[70] core의 소스코드가 LGPL v2로 배포되고 있는 것과 비슷합니다.

2.2.3. LGPL

LGPL은 주로 라이브러리에 사용하기 위해 FSF가 GPL과는 별도로 만든 라이선스이다. 라이브러리에 GPL 라이선스를 적용하게 되면 응용프로그램까지 GPL로 배포해야 하므로 사용자는 해당 라이브러리의 사용을 꺼리게 된다. FSF는 GPL의 내용을 약간 수정하여 라이브러리 자체를 수정한 경우에는 카피레프트 조항을 적용하지만, 해당 라이브러리를 이용한 응용프로그램은 카피레프트 조항을 적용하지 않고 소스코드 제공의무도 없는 형태로 LGPL 라이선스를 만들었다.

LGPL의 의무사항은 i) 각 복제본에 적절한 저작권 안내와 보증책임이 없음을 명시할 것, ii) LGPL 라이선스를 언급하는 안내 사항과 보증책임 관련 고지사항을 원본 그대로 유지할 것, iii) 소프트웨어를 양도받는 모든 이들에게 소프트웨어와

17) http://www.sir.co.kr/main/gnuboard4/license.php.
18) http://xe.xpressengine.net.

함께 LGPL 라이선스 사본을 제공할 것, iv) 라이브러리 형태로의 수정을 허용하며, 만약 수정한 경우 수정사실과 날짜를 파일에 명기할 것, iv) 원본저작물과 파생저작물을 LGPL 또는 GPL에 의해 배포할 것, v) 원본저작물 및 파생저작물에 대한 소스코드를 제공하거나, 요청 시 제공하겠다는 약정서를 제공할 것, vi) 응용프로그램을 배포할 경우, LGPL 라이브러리를 사용하고 있다는 사실을 명시할 것, vii) 사용자가 라이브러리를 수정해도 응용프로그램을 사용할 수 있도록 (예를 들어 응용프로그램의 오브젝트 코드를 제공하거나, 해당 라이브러리의 형태를 공유 라이브러리 방식 등을 이용하여) 허용할 것 등이다. i) ~ v)의 내용은 GPL 2.0과 동일하거나 유사하지만, vi) ~ vii)은 LGPL에 특유한 내용이다.

<프로젝트 사례> GNU C Library

LGPL로 배포되는 대표적인 프로젝트는 GNU C Library 또는 glibc이다. glibc는 GNU 프로젝트에서 배포되는 C 표준 라이브러리이다. 1980년대부터 FSF에서 만들어 배포하기 시작했는데, 1990년대 초반 리눅스 커널 개발자들이 glibc에서 분기(fork)된 별도의 Linux libc 프로젝트를 진행하였다. 그런데, FSF에서 1997년 POSIX 표준에 가깝고 다양한 기능을 추가한 glibc 2.0을 배포하자 리눅스 커널 개발자들은 별도의 프로젝트를 포기하고 FSF의 glibc에 합류하였다. 다만, glibc의 크기, 속도 등에 만족하지 못하는 오픈소스 개발자들을 중심으로 지금도 별도의 프로젝트들이 진행되고 있다. 예를 들면, 안드로이드 플랫폼에서 사용되는 Bionic(BSD)이 대표적이며, 이밖에 dietlibc(GPL), eglibc(LGPL), uClibc(LGPL), Newlib, Klibc(GPL) 등이 있다.

<상담 사례> OSS로 배포하는 LMS의 라이선스 선택

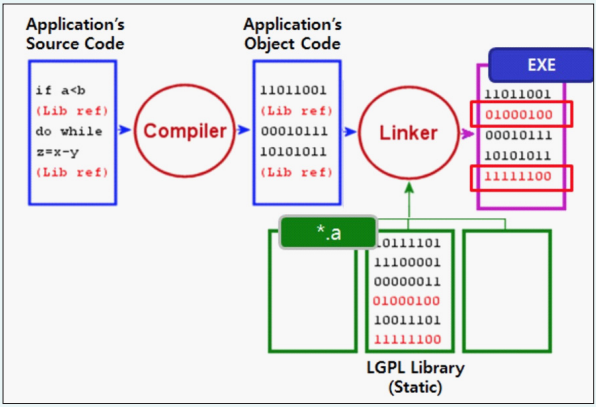
Q: 프로그램을 개발하면서 LGPL 라이브러리를 링크하여 사용할 때 소스 코드 공개에 대한 의무 사항이 궁금합니다.

A: LGPL은 라이브러리를 정적 링크할 때와 동적 링크할 때 다소 다른 의무 사항을 요구합니다.

1. LGPL 라이브러리를 정적 링크 시 요구 사항

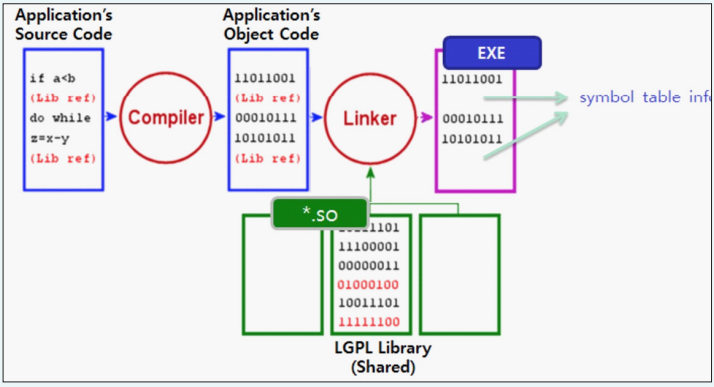
LGPL 라이브러리를 정적 링크하는 애플리케이션을 배포하는 경우에는 (즉, LGPL 라이브러리가 정적 라이브러리 (예: *.a 파일)) LGPL 라이브러리의 소스 코드를 공개해야 할 뿐만 아니라, 애플리케이션의 오브젝트 코드도 공개해야 합니다. 이는 사용자에게 LGPL 라이브러리를 수정하고, 수정된 라이브러리를 포함한 애플리케이션을 생성하기 위해 Relink 할 수 있는 방법을 제공하기 위한 LGPL의 요구사항을 충족시키기 위해서입니다.

19) Newlib은 여러 가지 오픈소스 라이브러리로부터 가져왔기 때문에 라이선스도 각각의 라이브러리에 따른다. http://www.sourceware.org/newlib/.



2. LGPL 라이브러리를 동적 링크 시 요구 사항

LGPL 라이브러리를 동적 링크하는 애플리케이션을 배포하는 경우에는 (즉, LGPL 라이브러리가 동적 라이브러리 (예: *.so 파일)) LGPL 라이브러리의 소스 코드를 공개해야 합니다. 이외, 애플리케이션의 소스코드나 오브젝트 코드는 공개할 필요 없습니다. 이는 사용자가 수정한 LGPL 라이브러리는 애플리케이션과 Relink 하지 않고도 애플리케이션 동작 시 반영되기 때문입니다.



단, LGPL 라이브러리가 애플리케이션에 포함되어 함께 배포되는 것이 아니라, 사용자의 컴퓨터에 이미 존재하고 있는 LGPL 라이브러리를 배포한 애플리케이션이 동적 링크하여 사용하는 구조라면, LGPL 라이브러리의 소스 코드조차도 공개할 필요 없습니다.

20) https://www.gnu.org/licenses/gpl-faq.en.html#LGPLStaticVsDynamic

<상담 사례> 자체개발한 라이브러리의 라이선스 선택

Q: 제가 만든 라이브러리를 사용할 경우, 제 라이브러리를 사용했다는 것을 알리고 싶고, 일반적으로 소스 공개의 의무는 없지만, 만약 저의 소스를 고칠 경우 제 소스의 경우에만 변경사항을 공개하길 원합니다. GPL처럼 전체소스의 공개를 원하지 않습니다. 상업용 프로그램 개발에도 이용되는 방향도 권장하는 바입니다. 또한 가급적 다른 라이선스들과도 양립가능하기를 원합니다. 그래서 관심 있게 보고 있는 것이라면 LGPL 과 MPL 이긴 한데요. LGPL, MPL 등 여러 라이선스로 허락해서 사용자들이 원하는 라이선스를 선택해서 사용할 수 있도록 하는 것이 가능한가요? 그리고, 동적링크는 소스공개 의무가 없는 것으로 알고 있는데, LGPL 라이브러리를 정적링크를 이용하여 사용할 때에는 링크된 응용프로그램의 소스코드까지 같이 제공해야 하나요?

A : 프로그램의 권리자는 자신의 프로그램을 어떠한 라이선스를 통해서도 배포할 수 있습니다. 또한 복수의 라이선스를 선택하여 배포할 수도 있습니다. 모질라 프로젝트의 경우 MPL, GPL, LGPL의 세가지 라이선스로 배포되고 있습니다.

LGPL 라이브러리를 정적으로 이용하는 경우에도 응용프로그램의 소스코드 제공 의무는 없습니다. 다만, 이용자들이 라이브러리를 수정하여 다시 응용프로그램을 이용할 수 있도록 애플리케이션의 오브젝트 코드를 제공할 의무가 있습니다.

<상담 사례> LGPL인 Java 라이브러리(Jar) 사용

Q: LGPL인 Jar 파일을 import하여 사용하려고 합니다. 이때 소스 공개 의무 사항이 어떻게 되는지 궁금합니다.

A: LGPL은 GPL과는 달리, 라이브러리를 링크하여 사용하는 애플리케이션에 대해서는 소스 코드 공개를 요구하지 않습니다. Java의 JAR 파일을 import 하는 것은 동적 링크를 생성하는 것과 유사하게 동작한다고 볼 수 있으므로 LGPL인 JAR 파일을 import 하여 사용하는 애플리케이션의 소스 코드는 공개할 필요가 없습니다. 단, LGPL인 JAR 파일 자체에 대한 소스 코드는 공개해야 합니다.[21]

21) http://www.gnu.org/licenses/lgpl-java.html

<상담 사례> 오픈소스를 이용한 웹서비스

Q: 홈페이지를 만들어 웹서비스를 제공하려 하고 있습니다. 1차적으로 LGPL이 걸린 오픈소스 SW를 이용하여 결과를 얻어내고, 이 결과를 이용, 분석하여 2차적인 결과를 만들어 내게 됩니다. 이 2차적인 결과를 상업적인 용도로 사용하려 합니다. 이렇게 사용하는 것이 문제가 없나요? 더불어, LGPL이 GPL일 경우에도 같이 설명해 주셨으면 좋겠습니다.

A: 오픈소스SW 라이선스는, 오픈소스SW를 이용해 상업적인 이득을 취하는 것을 막지 않으며, 별도로 명시하는 경우를 제외하고는, 일반적으로 오픈소스SW를 이용하여 산출된 결과물에 대해서도 제약을 가하지 않습니다. LGPL로 배포된 오픈소스SW 또한, 상업적인 이용에 제한을 두고 있지 않으며, 산출된 결과물이 라이선스의 영향을 받지는 않습니다.GPL 또한, 상업적 이용에 제한을 두지 않고 있으며, 그 결과물에 대한 정책도 동일합니다. GPL FAQ에서는 오픈소스SW의 단순이용 결과물에 대해서는 SW의 이용자에게 저작권이 발생하게 되며, 다만, 오픈소스SW의 일부가 결과물에 포함되는 경우에만 라이선스의 영향을 받게 된다고 명시하고 있습니다.[22]

<상담 사례> GPL과 LGPL의 양립성

Q: GPL과 LGPL이 양립가능한(compatible) 이유가 LGPL은 GPL로 재라이선스(re-license) 가능하기 때문에(LGPL 3조) 양립가능한 것인지요? 아니면, GPL로 재라이선스하지 않더라도 양립할 수 있는 것인지요? 아울러 GPL 적용시 기존 LGPL 표시를 GPL로 변경하여야 한다고 되어 있는데요, LGPL을 GPL로 재라이선스 할 시 LGPL SW의 수백~수천 개의 코드파일 상단에 LGPL 표시를 GPL로 바꾸는 작업을 해야 하는 것인지요? 아니면 꼭 그렇게 하지 않더라도 다른 방식으로 표기가 가능한 것인지요?

A: LGPL도 GPL처럼 "추가적인 제한(further restriction)"을 금지하는 규정을 가지고 있습니다. LGPL의 입장에서 보면 GPL의 카피레프트조항이 추가적인 제한에 해당하기 때문에 양립하지 않을 수 있습니다. 하지만 LGPL은 제3조를 통해 GPL로 배포할 수 있도록 허용함으로써 GPL과의 양립성을 갖게 됩니다. LGPL 3조는 사용자가 LGPL 코드를 수정하여 재배포할 때 LGPL이 아니라 GPL로 배포할 수 있으며, 만약 GPL로 배포할 거라면 모든 표시를 GPL로 바꾸라는 의미입니다. LGPL 코드를 GPL 코드와 결합하여 배포하는 경우에는, GPL의 해석상 LGPL 코드도 GPL로 배포해야 하기 때문에 GPL로 재라이선스해야 하는 것으로 해석됩니다. 하지만, LGPL코드를 전혀 수정하지 않았거나, 약간 수정했더라도 수정한 사람이 굳이 GPL로 배포할 것을 원하는 것이 아니라면, 굳이 GPL로 재라이선스할 필요는 없을 것 같습니다. GPL의 해석에는 맞지 않지만, 현실적으로 여기에 이의를 제기하거나 비난할 사람은 없어 보입니다. 그렇다면 굳이 번거롭게 표시를 모두 바꿀 필요는 없을 것 같습니다.

2.2.4. Affero GPL

BSD, Apache, GPL, LGPL, MPL, EPL 등 대다수의 오픈소스 라이선스들은 해당 소프트웨어를 복제하여 '배포(distribute)'할 때 지켜야 하는 다양한 요구사항들을

22) http://www.gnu.org/licenses/gpl-faq.html#CanIUseGPLToolsForNF 참조.

규정하고 있다. 이를 반대로 해석하면 해당 소프트웨어를 배포하지 않고 기업이 해당 오픈소스를 내부적으로만 사용하거나 네트워크 서버 형태로 이용하고 있는 경우에는 라이선스에 따른 의무사항들이 거의 없다는 점이다.

오픈소스 라이선스의 이러한 한계에 대해 비판적인 견해를 가지고 있던 Affero 프로젝트[23]는 기존의 GPL 라이선스를 변경하여 네트워크 서버에 의해 서비스를 제공하는 경우에도 카피레프트 조항과 소스코드 제공의무가 적용되도록 하였다. Affero GPL의 의무사항은 GPL과 기본적으로 동일하다. 다만, 그 적용의 범위가 단순한 '배포'를 넘어서 네트워크 서버 형태로 소프트웨어를 이용하는 경우에도 적용된다는 점에 차이가 있다.

Affero GPL 1.0

2.d) If the Program as you received it is intended to interact with users through a computer network and if, in the version you received, any user interacting with the Program was given the opportunity to request transmission to that user of the Program's complete source code, you must not remove that facility from your modified version of the Program or work based on the Program, and must offer an equivalent opportunity for all users interacting with your Program through a computer network to request immediate transmission by HTTP of the complete source code of your modified version or other derivative work.

<참고> CPAL, APSL

Affero GPL과 같이 네트워크 서버에도 라이선스 의무사항을 준수할 것을 요구하는 대표적인 오픈소스 라이선스는 CPAL과 APSL이다.

Common Public Attribution License Version 1.0

15. ADDITIONAL TERM: NETWORK USE.

The term "External Deployment" means the use, distribution, or communication of the Original Code or Modifications in any way such that the Original Code or Modifications may be used by anyone other than You, whether those works are distributed or communicated to those persons or made available as an application intended for use over a network. As an express condition for the grants of license hereunder, You must treat any External Deployment by You of the Original Code or Modifications as a distribution under section 3.1 and make Source Code available under Section 3.2.

23) http://www.affero.org/

Apple Public Software License

1.4 "Externally Deploy" means: (a) to sublicense, distribute or otherwise make Covered Code available, directly or indirectly, to anyone other than You; and/or (b) to use Covered Code, alone or as part of a Larger Work, in any way to provide a service, including but not limited to delivery of content, through electronic communication with a client other than You.

<참고> GPL 3.0과 AGPL

GPL의 개정 과정에서도 네트워크 서비스의 경우에도 GPL을 적용할 것인가에 관한 논의가 있었지만, 최종적으로 GPL 2.0과 마찬가지로 GPL 3.0도 배포(Convey)의 경우에만 적용하기로 했다. 다만 Affero GPL의 취지를 존중하여, 특별한 조항을 두게 되었다. GPL은 원칙적으로 이용자들에게 GPL이 규정하고 있는 내용 이외에 다른 추가적인 제한(further restriction)을 부가하지 못하도록 하고 있으며, 그 결과 이와 같은 제한을 가진 다른 오픈소스 라이선스들과는 양립 불가능(incompatible)하다. Affero GPL 또한 GPL보다 엄격한 제한을 하고 있으므로 원칙적으로 GPL과 양립 불가능하다. 그러나 GPL 3.0 제13조에서는 이와 같은 제한사항에도 불구하고 GPL 3.0과 Affero GPL 3.0은 양립 가능하다고 규정하였으며, Affero GPL도 3.0 버전을 배포하면서 비슷한 조항을 삽입하였다.

GPL 3.0

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

<상담 사례> SaaS 형태로 오픈소스를 이용하는 경우

Q: 우리 회사에서 IT 자원은 회사의 IDC에 설치하고, 고객에게는 네트워크를 통해 서비스를 제공하는 형태인 IaaS(Infrastructure as a Service), PaaS(Platform aaS), SaaS(Service aaS)를 기획하고 있습니다. 만약 이러한 서비스를 제공하면서 소프트웨어를 도입, 수정해서 적용하는 경우 소스코드 공개의 의무가 발생하는 건가요?

A: GPL을 비롯한 대부분의 오픈소스 라이선스의 의무사항은 소프트웨어를 배포할 때 발생합니다. 따라서 네트워크 서비스를 통해 오픈소스를 이용하는 경우에는 소스코드 제공 등의 의무가 발생하지 않습니다. 그런데 일부 오픈소스 라이선스는 네트워크 서비스 형태로 오픈소스를 이용하는 경우에도 소스코드 제공 등의 의무가 있다고 규정하고 있습니다. Affero GPL, Common Public Attribution License 등이 대표적입니다. 따라서 이용하고자 하는 오픈소스의 라이선스를 구체적으로 확인할 필요가 있습니다.

<상담 사례> AGPL 라이브러리를 이용한 소프트웨어

Q: Linux에서 파일을 처리하는 AGPL 라이브러리를 가져다 수정하여 다른 OS에 포팅한 후, 수정된 라이브러리를 사용하여 네트워크상으로 서비스를 제공하는 소프트웨어를 개발하였습니다. 저희가 수정한 AGPL 라이브러리의 소스코드는 공개할 수 있습니다. 그런데, Network로 서비스하는 소프트웨어는 보안 등 여러 가지 문제로 인해 공개하기가 어렵습니다. 라이브러리를 사용하여 네트워크상으로 서비스를 제공하는 소프트웨어의 소스를 공개해야 하나요?

A: 원칙적으로 AGPL 프로그램을 수정하거나 파생저작물을 만들어 이용하는 경우, 수정 버전 및 파생저작물에 대한 소스코드를 컴퓨터 네트워크를 통해 원격으로 대화하는 모든 프로그램 사용자들에게 제공해야 합니다. 따라서, AGPL 라이브러리를 수정하여 소프트웨어개발에 사용하셨다면 수정된 라이브러리 및 해당 라이브러리를 이용하는 서비스 프로그램의 소스코드를 제공해야 할 것으로 보입니다.

2.2.5. GPL Exceptions

여러 오픈소스 프로젝트들은 해당 프로젝트는 GPL로 배포되길 바라면서 이를 활용하여 동작하는 프로그램들은 GPL의 copyleft 조항에서 자유로울 수 있도록, GPL 2.0의 2조를 다소 완화한 조건으로 배포할 수 있도록 허락해주는 exception을 추가하기도 한다. 이는 추가적인 제한 사항을 주는 것이 아니므로 GPL과도 호환된다.

Bison Exception

Bison은 GNU 파서 생성기로, 정의된 문법을 처리하고 해석하여 C 코드로 만들어주는 도구다. Bison의 주요 결과물(Bison parser implementation file)은 상당 부분 Bison의 소스 코드를 그대로 복사하여 skeleton 코드를 포함하게 된다. 일반적인 GPL이라면 이 skeleton 코드에도 GPL이 적용되어야 하고, 이에 따라 bison 결과물인 C 파일과 이를 사용하는 프로그램 모두 GPL이 적용되어, 사용자들이 bison 사용을 부담스러워할 수 있다.
이에 Bison에 예외 조항을 추가하여 Bison이 생성하는 소스 코드에 대해서는 GPL

적용이 되지 않도록 예외를 추가하였다.
소스 코드 내 GPL 라이선스 설명 아래 다음과 같은 문구가 있으면 Bison exception이 적용되는 파일로 볼 수 있다.

> As a special exception, you may create a larger work that contains part or all of the Bison parser skeleton and distribute that work under terms of your choice, so long as that work isn't itself a parser generator using the skeleton or a modified version thereof as a parser skeleton. Alternatively, if you modify or redistribute the parser skeleton itself, you may (at your option) remove this special exception, which will cause the skeleton and the resulting Bison output files to be licensed under the GNU General Public License without this special exception.
>
> This special exception was added by the Free Software Foundation in version 2.2 of Bison.

Classpath exception

GNU Classpath[24] 프로젝트는 자바 언어의 가상머신 및 컴파일러에서 사용되는 핵심 클래스 라이브러리를 자유 소프트웨어로 대체하기 위한 프로젝트이다. 이를 널리 사용할 수 있도록, GNU Classpath 등 Classpath exception이 적용된 수정하지 않은 core class library를 link하여 생성한 program은 GPL의 영향을 받지 않는다. 예를 들면, OpenJDK 프로젝트에서 가상머신 자체는 GPL 2.0으로 배포하고, Class library와 가상머신 내의 Public API로 노출되는 부분은 Classpath exception으로 배포하고 있다.

Autoconf exception

GNU Autoconf[25]는 M4 매크로의 확장 패키지로, 소스코드 패키지들의 환경을 설정하는 쉘 스크립트를 자동으로 생성한다. Autoconf는 GPL로 배포되며, configure.ac의 시스템 정보를 입력받아, Autoconf의 일부분과 결합, configure 스크립트를 생성한다.
이에 해당 configure 내에 autoconf의 일부가 포함될 수밖에 없음에도 해당 configure는 GPL의 파생 저작물이 되지 않도록 예외를 적용해주는 것이 autoconf exception이다.

24) https://www.gnu.org/software/classpath/
25) https://www.gnu.org/software/autoconf/autoconf.html

GCC Runtime Library Exception (GCC RLE)

GCC[26]는 GNU 대표적인 컴파일러로, 이 Exception의 목적은 non-GPL 소스코드를 GCC로 컴파일할 때, GCC Runtime Library (header file 포함)가 Program에 포함되는 것을 허용하기 위한 것이다. 이 Exception에 의하면, "정당한 Compile 과정 (Eligible Compilation Process)" 중 Independent Module과 GCC Runtime Library가 결합하여 형성된 Target Code라면, 이는 GPL에 상관없이 다른 라이선스로 배포될 수 있다.

GCC Runtime Library는 libgcc, libstdc++, libfortran, libgomp, libdecnumber 등이 있다. 만약 GCC Runtime Library가 정당한 컴파일 과정에 의해서가 아닌, 다른 방법으로 Program에 포함된 경우에는 GPL 의무 사항을 모두 준수해야 한다.

2.3. MPL형 라이선스 및 주요 프로젝트

MPL형 라이선스는 주로 기업들이 주도하는 오픈소스 프로젝트에서 사용하는 라이선스로 MPL, CDDL, EPL 등이 포함된다. BSD형과 GPL형의 라이선스와는 달리, 처음부터 법률가들이 참여하여 만들었기 때문에 소프트웨어 라이선스의 관점에서는 더욱 정교하다고 볼 수 있지만, 프로그래머들에게는 그만큼 복잡하고 이해하기 어려운 라이선스들이다. 카피레프트 조항을 포함하고 있다는 점에서 GPL형과 비슷하지만, 적용 범위와 소스코드 제공범위는 GPL보다는 LGPL에 가까운 것으로 볼 수 있다.

2.3.1. MPL

MPL은 1998년 넷스케이프사가 자사의 브라우저를 오픈소스로 배포하면서 만든 라이선스이다. MPL로 배포되는 오픈소스를 이용하기 위해서는 i) 원 코드에 포함된 저작권 표시, 개발자 및 권리자에 관한 사항들을 그대로 표시할 것과, ii) 배포 시 MPL 라이선스 사본을 첨부할 것, iii) 수정했을 경우에는 최초개발자의 코드로부터 파생되었다는 사실, 수정사항 및 날짜 등을 포함한 파일(Exhibit A)을 소스코드의 각 파일에 포함할 것, iv) 원본 및 수정코드를 MPL에 의해 배포할 것과, v) 수정코드에 대한 소스코드를 전자배포방식 등을 통해 제공할 것, vi) MPL 코드를 사용할 때 제3자의 지식재산권에 의한 라이선스가 필요하다는 사실을 알고 있는 경우 “LEGAL"

26) https://gcc.gnu.org/

파일에 관련 내용을 포함할 것 등이다.

2.3.2. MPL 2.0은 MPL 1.1에 비해 더욱 짧고 이해하기 쉽게 만든 라이선스이다. 특히 Apache License 2.0과 양립할 수 있게 되었으며 특허 관련 조항들이 정비되었다. 또한 MPL 2.0에서 Secondary License로 정의한 라이선스인 GPL 2.0, LGPL 2.1, AGPL 3.0과 각각 이후 버전들)과 양립할 수 있게 된 것이 큰 변화이다.

<참고> MPL의 소스코드 제공범위

MPL에도 카피레프트 조항이 포함되어 있다는 점에서 GPL과 유사하다. 즉 수정한 코드에 대한 소스코드를 제공해야 한다. 그런데 MPL은 "수정(modification)"의 의미를 "원 코드 또는 이전 수정코드의 내용이나 구조에 추가되거나 삭제되는 것을 말하며, 코드가 복수의 파일로 배포될 경우에는 i) 원 코드나 이전 수정코드를 포함하는 파일의 내용에 추가하거나 삭제된 코드, ii) 원코드나 이전 코드의 일부분을 포함하는 새로운 파일"을 말하는 것으로 정의하고 있다. 따라서 MPL 코드가 포함되지 않은 새로운 파일을 만든 경우에는 MPL 코드와의 결합방식에 상관없이 소스코드를 제공할 필요가 없는 것으로 해석할 수 있다.

<프로젝트 사례> Firefox 웹브라우저

MPL 라이선스로 진행되는 대표적인 프로젝트는 Firefox 등 Mozilla가 진행하는 프로젝트들이다. 그런데, MPL 라이선스는 GPL 형의 라이선스와 양립성이 없으므로 GNU 프로젝트와 모질라 프로젝트를 함께 이용하는 것이 불편하다. 이러한 문제점을 해결하기 위해 Mozilla는 Firefox를 포함한 대부분의 프로젝트에 대해 MPL뿐만 아니라 GPL, LGPL 라이선스를 적용하여 배포하는 트리플(triple) 라이선스 정책을 채택하고 있다. 한편 "Mozilla Firefox"는 상표로 보호되고 있으며 일정한 조건을 만족하는 경우에만 상표로 사용할 수 있다.
그 결과 Debian 프로젝트에서는 Firefox를 수정한 웹 브라우저를 "Iceweasel"이라는 이름으로 사용하고 있다.

2.3.2. CDDL

CDDL은 썬(sun)이 자사의 유닉스 운영체제인 솔라리스를 오픈소스로 배포하면서 만든 라이선스이다. MPL을 참조하여 만들었기 때문에 MPL의 내용과 비슷하다. i) 저작권 등 권리관련 사항, 라이선스 관련 사항 등의 고지사항을 제거하거나 변경할 수 없으며, ii) 배포 시 CDDL 라이선스 사본을 첨부해야 하며, iii) 수정한 경우 수정코드의 기여자임을 밝혀야 한다. iv) 원본 및 수정코드를 CDDL에 의해 배포해야 하며, v) 수정코드에 대한 소스코드를 합리적인 방식으로 제공해야 한다.

<프로젝트 사례> 오픈 솔라리스

CDDL로 배포되는 대표적인 사례는 썬(Sun)의 솔라리스 운영체제이다. 리눅스, BSD 등 오픈소스 경쟁제품들에 대응하여 썬의 핵심 제품 중의 하나인 솔라리스를 오픈소스화한 것이다. 그런데 2009년 오라클이 썬을 인수하면서 썬이 추진해 왔었던 오픈소스 정책에 많은 변화가 나타나고 있다. 자바(Java)의 지식재산권에 기반을 둔 구글을 상대로 소송을 제기한 사례도 이러한 변화를 보여준다. 향후 오픈 솔라리스의 전망도 불투명한 상황이다. 현재 오라클은 오픈 솔라리스의 배포를 중단하고 있다.

<상담 사례> CDDL 오픈소스를 이용한 제품의 소스코드

Q: CDDL로 배포되는 오픈소스를 이용하여 만든 B2B 제품을 타 업체에 납품하고 있습니다. CDDL의 경우, 소스코드에 대해 수정을 하였을 경우, 고지사항을 작성하여 두거나 자사 사이트를 통해서 소스코드를 공개하면 되는 것으로 알고 있습니다. 만약, 소스코드를 수정하지 않고, 있는 그대로의 라이브러리를 사용하였을 경우에도, 고지사항을 작성하거나 소스코드를 공개해야 하는지 궁금합니다.

A: CDDL로 배포되는 오픈소스를 오브젝트 형태로 재배포하는 경우에는 프로그램에 대한 소스코드를 기여자로부터 제공받을 수 있다는 사실을 밝히고 수취인이 이 소스코드를 소프트웨어 교환에 통상적으로 사용되는 매체를 통해 적절한 방법으로 구할 방법을 알려주어야 합니다. 따라서 CDDL 코드를 수정하지 않고 사용하는 경우라면, 해당 소스코드를 제공받을 수 있는 사이트를 적절히 안내하면 직접 소스코드를 제공하지 않아도 되는 것으로 해석됩니다. 다만, 소스코드 제공 의무를 제외한 기타의 준수사항들(보증의 부인 및 책임의 제한 등)은 지켜야 합니다.

2.3.3. CPL, EPL

CPL과 EPL은 IBM이 이클립스(Eclipse) 등 오픈소스 프로젝트를 진행하면서 만든 라이선스이다. CPL과 EPL은 특허보복조항에 관한 사항에서만 차이가 있고 다른 내용은 모두 동일하다. IBM은 현재 EPL만 사용하고 있다. EPL로 배포되는 소프트웨어를 배포할 때 지켜야 할 의무사항으로는 i) 각 코드의 저작권 고지사항을 제거하거나 변경하지 말 것과, ii) EPL 라이선스 사본을 포함할 것과, iii) 각 기여물의 창작자를 식별할 수 있도록 신분을 밝힐 것과, iv) 오브젝트코드로 배포하는 경우 EPL 조건을 준수하고, 보증부인 및 책임배제에 관한 내용과 소스코드의 확보방법을 알려줄 것, v) 소스코드로 배포하는 경우 EPL 라이선스를 적용할 것과, vi) 상업적 배포의 경우 기여자에게 책임이 발생하지 않도록 조치할 것 등의 내용을 포함하고 있다.

27) http://www.juzzle.org.

<상담 사례> CPL로 배포되는 Juzzle을 사용하는 경우

Q: Juzzle[27]을 사용하고 있습니다. CPL 1.0으로 배포되는 것으로 알고 있습니다. 상업적이든 비상업적이든 관계없이 사용할 수 있나요? 소스 코드를 수정해서 사용하는 경우 어떠한 의무사항이 있나요? 또는 소스코드를 수정하지 않고 프로그램을 이용만 하는 경우 지켜야 할 사항은 무엇인가요?

A: OSI에서 인증한 오픈소스 라이선스들은 모두 상업적 이용에 제한을 하지 않고 있으며, CPL도 마찬가지입니다. 수정여부와 상관없이 CPL 코드를 이용하는 경우 저작권 고지사항을 제거하거나 변경하지 말 것, CPL 라이선스 사본을 첨부할 것, 오브젝트코드로 배포하는 경우 소스코드의 확보방법을 고지할 것, 소스코드로 배포하는 경우 CPL 라이선스를 적용할 것 등이 있습니다. 덧붙여 소스코드를 수정하여 배포하는 경우에는 기여자로서 창작자를 식별할 수 있도록 신분을 밝힐 것 등이 포함됩니다.

<참고> 소스코드의 제공범위

EPL도 카피레프트 내용을 포함하고 있다. 재배포시에 EPL이 적용되는 프로그램은 "기여(Contribution)"의 개념에 의해 결정된다. EPL은 "기여(물)"(이)란, a) 최초 기여자의 경우, 본 계약서에 따라 배포된 최초의 코드 및 문서를 말하며, b) 이후의 각 기여자의 경우에는 i) 프로그램에 대한 변경사항 및 ii) 프로그램에 대한 추가사항을 말한다"고 규정하고 있으며, 이 때 "프로그램에 대한 추가사항 중에서, (i) 다른 라이선스 계약서에 따라 프로그램과 관련하여 배포된 소프트웨어의 별도 모듈인 경우, 그리고 (ii) 프로그램의 파생 저작물이 아닌 경우는 기여에 포함되지 않는다"고 규정하고 있다. 다시 말해서 EPL의 경우에는 "모듈(module)"을 기준으로 소스코드의 제공범위를 결정한다.

<상담 사례> CPL, EPL의 소스코드 공개범위

Q: CPL, EPL의 경우 공개 범위가 어떻게 되는 건가요? 모듈 단위로 공개를 한다고 하는데, 모듈의 의미가 애매해서 정확한 가이드가 있었으면 합니다. 또, CPL과 EPL간의 차이가 뭔지도 상세한 설명 부탁드립니다.

A: MODULE 단위 소스코드 공개 요구사항은 원 저작물의 소스코드를 수정하여 새로운 SW에 포함하였을 경우에 해당되며,원 저작물을 사용함에 있어 수정한 부분에 대하여, 원 저작물의 소스코드에서부터 존재하던 파일 중 수정된 부분을 포함하는 모듈을 모두 공개하여야 합니다. 이에 따라, 일반적으로 원 저작물의 모듈을 새로운 SW에 적용하기 위해 인터페이스의 수정 등을 거치는 경우가 해당하며, 파생 저작물의 저작자가 새로이 생성한 부분 중 기존 모듈에 해당되지 않는 부분에 대해서는 공개하지 않아도 됩니다. 이러한 '모듈'의 범위에 대해서는 상황에 따라 다르지만, 일반적인 모듈의 정의에 따라 특정 기능을 수행하는 단위를 기준으로 하여 배포하여야 하는 것으로 판단됩니다.

CPL과 EPL에서는 다음과 같은 내용을 포함하여 별도의 모듈인 경우 소스코드의 제공범위에 포함하지 않는 것으로 보고 있습니다.

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

한편, CPL과 EPL은 특허보복조항의 적용범위에 관해서만 차이가 있고 나머지 내용은 동일합니다. 즉, EPL은 CPL 제7조 중 아래의 내용만 삭제하고 이름을 바꾼 것입니다. CPL의 경우 라이선시가 기여자를 상대로 (모든) 소프트웨어 관련 특허침해소송을 제기하는 경우 라이선스가 종료되는 것으로 규정하고 있었는데, 특히 다수의 특허 포트폴리오를 가지고 있는 기업의 입장에서는 부담스런 내용이었기 때문에 이 부분을 삭제하고, EPL로 배포되고 있는 해당 소프트웨어와 관련된 특허침해소송의 경우로만 제한한 것입니다.

CPL 1.0 제7조

"If Recipient institutes patent litigation against a Contributor with respect to a patent applicable to software (including a cross-claim or counterclaim in a lawsuit), then any patent licenses granted by that Contributor to such Recipient under this Agreement shall terminate as of the date such litigation is filed."

<프로젝트 사례> 이클립스(Eclipse) 프로젝트

EPL로 배포되는 대표적인 프로젝트는 이클립스(Eclipse)이다. 이클립스는 통합개발환경(IDE)을 포함한 대표적인 소프트웨어 개발 환경이다. 주로 자바로 작성되어 자바 애플리케이션 개발 도구로 사용되지만, 플러그인 시스템을 통해 확장하는 것이 가능하므로 C, C++, COBOL, Python, Perl, PHP 등으로 개발할 때도 이용할 수 있다.

2.4. 폰트 라이선스

오픈소스 소프트웨어 확산과 함께 오픈소스 폰트도 많이 만들어지고 있다. 2014년 어도비와 구글이 손을 잡고 무료 폰트를 공개하기 시작했으며, 현재 구글은 http://fonts.google.com 을 통해 오픈소스 폰트를 공개하고 있다.

2.4.1. GPL Font Exception

폰트의 저작권법 특성상, GPL 폰트를 내장한 문서는 GPL의 파생저작물로 간주 될 수 있다. 하지만, 이는 대부분 폰트 저작권자의 의도는 아닐 것이다. 이와 같은 이유로

FSF (Free Software Foundation)의 David "Novalis" Turner는 2005년 4월, font exception을 만들었다. 즉 GPL Font Exception이 적용된 폰트라면 폰트 자체는 GPL에 의해 저작권 및 라이선스 고지와 해당 폰트 공개 의무사항을 준수해야 하지만, 폰트가 적용된 문서에는 GPL이 적용되지 않는 것이다.

GPL Font Exception

As a special exception, if you create a document which uses this font, and embed this font or unaltered portions of this font into the document, this font does not by itself cause the resulting document to be covered by the GNU General Public License. This exception does not however invalidate any other reasons why the document might be covered by the GNU General Public License. If you modify this font, you may extend this exception to your version of the font, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.[28]

2.4.2. SIL Open Font License (OFL)

SIL Open Font License는 SIL International에서 자신들의 unicode 폰트를 배포할 대 사용하는 라이선스로 저작권 및 라이선스 고지를 하면 자유롭게 사용 가능하다. 폰트 자체를 단독으로 판매 금지하는 조항이 있어 GPL과 호환되지 않으며, 다른 프로그램과 함께 배포하거나 판매하는 것은 허용된다.

2.4.3. Ubuntu Font License

3.1.1. Ubuntu Font License(UFL)는 Ubuntu에서 만든 Ubuntu Font Family를 배포할 때 사용하는 라이선스이다. OFL의 영향을 받아 만든 라이선스이지만, 해당 폰트를 배포할 때 다시 Ubuntu Font License로 배포해야 하는 copyleft 성격을 지니고 있다는 점에서 큰 차이를 갖는다. 배포 시 의무 사항으로는 저작권 및 라이선스 고지를 해야 하며, 수정 여부에 관계없이 폰트 소프트웨어의 모든 부분은 다시 UFL로 배포되어야 한다. Font를 배포하는 것을 목적으로 하는 것이 아닌 문서에 해당 UFL 하의 Font가 적용되어 사용된 경우는 라이센스의 적용 대상이 아니므로, 이는 GPL Font Exception과 유사하다고 볼 수 있다.
추가적인 의무 사항으로, 원본은 Font 명칭도 그대로 사용해야 하며, 상당한 변형이 있는 수정 버전은 원본 이름 및 유사한 명칭이 아닌 전혀 새로운 명칭을 사용해야 하며, 변형이 크지 않은 수정 버전은 원본 명칭을 유지하면서 “derivative x”를 추가하는 형식으로 원본의 이름과 구분하게 해야 한다.

28) http://www.gnu.org/licenses/gpl-faq.en.html#FontException

3

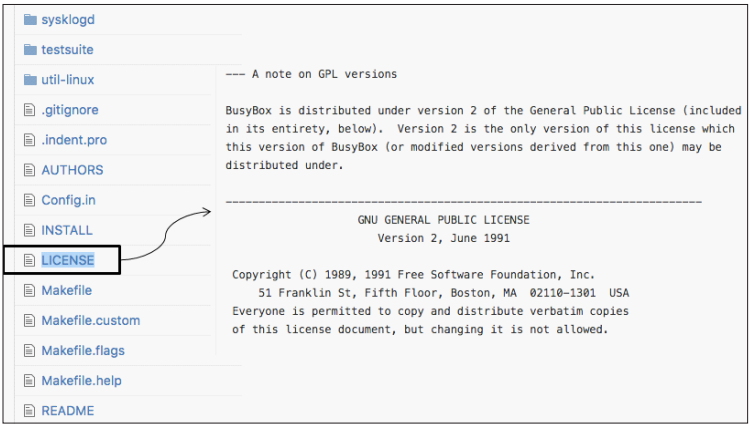
오픈소스 라이선스의 확인과 표시

3.1. 라이선스의 확인

오픈소스 라이선스 의무 사항을 올바르게 준수하기 위해서는 배포하는 소프트웨어 내에 어떠한 오픈소스가 사용되었는지를 먼저 파악해야 한다. 소프트웨어를 개발하면서 사용한 오픈소스를 파악하는 방법은 직접 확인하는 방법과 Tool을 이용하는 방법으로 구분할 수 있다.

3.1.1. 직접 확인하는 방법

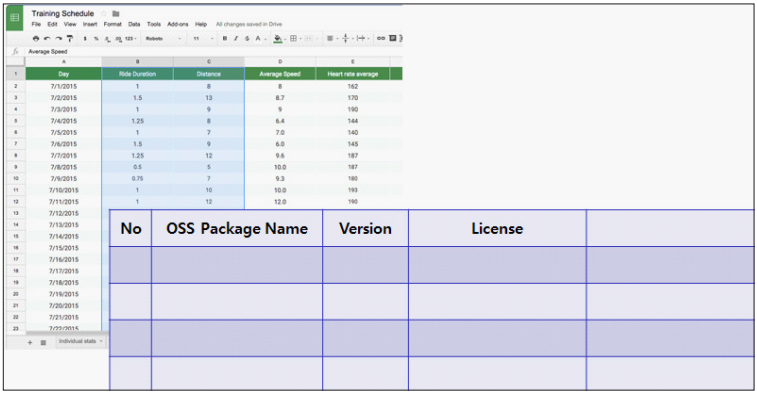
4.1.2. 주요 오픈소스의 패키지 내에는 라이선스 정보를 나타내는 파일이 포함되어 있다. 최상위 디렉토리 내에 LICENSE, COPYING 등의 이름으로 존재하며, 오픈소스 사용자는 이 파일을 통해 패키지가 어떤 오픈소스 라이선스로 배포되고 있는지를 확인할 수 있다.[29]



또한, 오픈소스는 소스 파일 내 라이선스 문구가 명시되어 있으므로 오픈소스 사용자는 각 파일 내 포함된 문구를 통해 라이선스를 확인할 수 있다.

29) TLDRLegal (https://tldrlegal.com/)에서 제공하는 라이선스 검색 기능을 이용하면 텍스트 검색을 통해 라이선스 정보를 확인할 수 있다.

이처럼 소프트웨어 개발자들이 사용하려는 오픈소스의 라이선스를 직접 확인하고, 이를 개발팀 내에서 공유할 수 있는 문서에 기재해 나가는 형태로 오픈소스 확인 프로세스를 갖춘다면, 개발이 완료되는 시점에 별도의 자동화 도구를 이용하지 않고도 개발 소프트웨어 내에 포함된 오픈소스 및 각 라이선스 정보를 확보할 수 있게 된다.



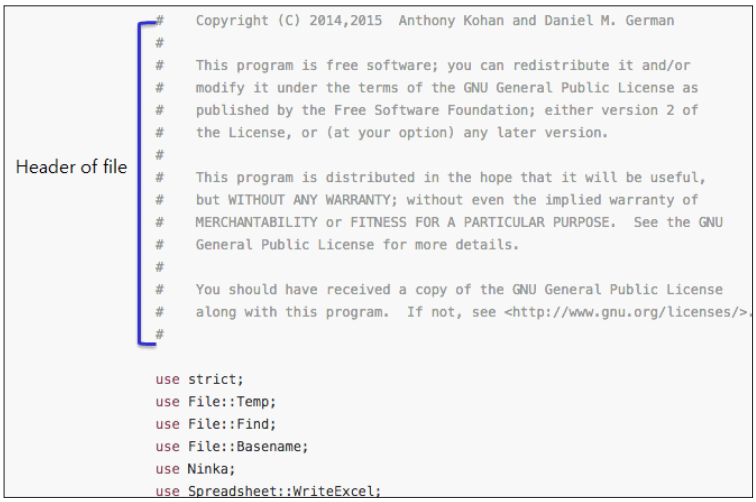
이처럼 직접 확인하는 방법은 소프트웨어 개발팀의 규모가 일정 수준 이하이고, 오픈소스 라이선스 교육이 충분히 선행되었을 경우에 적합하다.

3.1.2. 도구를 이용하는 방법

소프트웨어 내 오픈소스 포함 여부를 확인하기 위한 도구는 다음 세 가지 형태로 분류할 수 있다. 소스 코드 내의 문자열을 검색하는 도구와 실제 소스 코드 내용을 스캔하여 오픈소스 코드와 비교하는 도구, 그리고 소스 코드 없이 바이너리를 스캔하여 오픈소스를 확인하는 도구이다.

3.1.2.1. 문자열 검색 도구

문자열 검색 도구는 소스 파일 상단의 라이선스 문구를 검색하여 자동으로 라이선스를 판단하는 기능을 수행한다.



문자열 검색 기능을 수행하는 도구들은 다음과 같으며 모두 오픈소스로 공개되어 있어서 기업이 무료로 사용할 수 있다.

1. FOSSology[30] : GPL 2.0
2. Ninka[31] : GPL 3.0
3. Scancode[32] : Apache 2.0
3. TripleCheck[33] : AGPL 3.0

하지만, 문자열 검색 도구는 소스 파일 내 라이선스 고지 문구가 삭제된 경우, 오픈소스 사용 여부를 발견할 수 없다는 약점이 존재한다. 따라서 문자열 검색 도구를 기업이 효과적으로 사용하기 위해서는 개발자들이 소스 파일 내 라이선스 고지 문구를 수정하거나 삭제하지 않도록 충분한 교육이 선행되어야 한다.

30) http://fossology.org
31) http://ninka.turingmachine.org
32) https://github.com/nexB/scancode-toolkit/
33) http://triplecheck.net/download.html

3.1.2.2. 코드 스캔 도구

코드 스캔 도구는 오픈소스의 소스코드와 정확하게 일치하거나 유사한 패턴을 보이는 코드 조각이 존재하는지를 확인하는 기능을 수행한다. 따라서 소스 코드 내 라이선스 문구가 삭제된 경우에도 FOSSology 등의 문자열 검색 도구와는 달리 코드 스캔 도구는 오픈소스 검출이 가능하다.

코드 스캔 기능을 수행하는 도구들은 다음과 같으며, 모두 라이선스를 구매해야 사용할 수 있는 상용 도구이다.

Synopsys BlackDuck[34]
Flexera Code Insight[35]
RogueWave OpenLogic[36]
FOSSID[37]
Whitesource[38]
LSWARE FOSSGUARD[39]

코드 스캔 도구를 제공하는 기업들은 Github 등 오픈소스 호스팅 사이트에서 주기적으로 오픈소스를 취합하여 오픈소스 데이터베이스를 구축하고, 이를 활용하여 사용자의 소스 코드와 일치하는 오픈소스 정보를 제공한다. 하지만 오픈소스는 특성상 시간이 지나면서 여러 개발자에 의해 수정, 배포가 반복되면서 다수의 새로운 오픈소스로 배포될 수도 있는데, 코드 스캔 도구는 오랜 시간 동안 이런 형태로 생성된 오픈소스를 모두 데이터베이스에 포함하게 되고, 그 결과로 분석 대상 코드에 대해 다수의 오픈소스와 일치하는 분석 결과를 보여주게 된다. 일반적으로 코드 스캔 도구가 제공하는 기능은 이렇게 분석 대상 코드와 일치하는 데이터베이스 내의 오픈소스를 찾아주는 역할만 수행하고, 실제 정확한 출처는 사용자가 직접 판단해야 하는 역할을 요구한다. 오픈소스 라이선스에 익숙하지 않은 사용자는 이러한 판단을 하기가 쉽지 않으며, 특히 해당 소프트웨어를 개발한 사람이 아니면 잘못된 판단을 할 가능성이 높다. 따라서 코드 스캔 도구에서 검출이 된 코드 조각에 대해서는 도구 사용자가 아닌 해당 코드의 개발자에게 코드 출처를 확인하여 라이선스를 판단하는 것이 효과적인 코드 스캔 도구의 사용 방법이다.

34) https://www.blackducksoftware.com
35) http://www.palamida.com/
36) http://www.roguewave.com/products-services/open-source-support
37) http://fossid.com
38) https://www.whitesourcesoftware.com/
39) http://www.lsware.co.kr/solution/sol_foss.html

한국저작권위원회에서는 오픈소스 사용 여부를 확인해 볼 수 있도록 코드 아이를 개발하여 국내 영세업체를 대상으로 무상 서비스를 제공하고 있다.[40] 코드 아이를 사용하여 자신의 소스 코드 파일이나 폴더를 선택해 검사를 요청하면, 서버에 저장된 데이터베이스와 비교해 검사 보고서를 제공한다. 소스 코드는 암호화된 상태로 SSL(Secure Socket Layer) 상에서 전달되기 때문에 유출될 위험은 없다.

<상담 사례> 소스 파일 내 라이선스 문구와 COPYING 파일의 라이선스 정보가 다를 경우

Q: 사용하려는 OOO 오픈소스 패키지의 최상위 디렉토리 내에 존재하는 COPYING 파일은 GPL 3.0인 반면, 소스 파일 내 라이선스 문구는 GPL 2.0입니다. 어떤 라이선스로 판단해야 할까요?

A: 오픈소스는 통상 최상위 디렉토리 내에 "LICENSE", "NOTICE", "README", "COPYING" 등의 파일을 포함해서 라이선스 정보를 고지합니다.

이와는 별도로 COPYING 파일의 경우, GPL이나 LGPL 등 라이선스의 "사본 동봉 의무"에 따라 포함되는 경우도 있으므로, 해당 라이선스가 전체 소스 파일에 적용되는 것은 아닐 수 있음에 주의가 필요합니다. COPYING 파일 내 라이선스 문구와 소스 파일 상단의 라이선스 문구가 서로 다르다면, 각 소스 파일에 대해서는 먼저 소스 파일 내 라이선스 문구를 기반으로 라이선스를 판단해야 합니다.

만약, 소스 파일 내의 라이선스 문구와 최상위 디렉토리 내의 라이선스 정보 파일 내 라이선스가 서로 전혀 다를 경우에는 저작권자에게 연락하여 해당 오픈소스를 어떤 라이선스로 공개한 것인지 확인을 요청하는 것이 좋습니다. 실제 일부 오픈소스 저작권자의 경우, 실수로 잘못된 라이선스 정보 파일이 포함된 것을 뒤늦게 확인하여 수정 후 재배포하는 경우가 있습니다.

40) https://www.olis.or.kr/ossw/codeEye/introduction.do

3.1.2.3. 바이너리 스캔 도구

3rd Party로부터 소프트웨어를 받을 때는 대부분 소스 코드 없이 바이너리 형태로 입수한다. 그런데 지금까지 언급한 도구들은 모두 바이너리 형태의 소프트웨어에 대해서는 오픈소스 확인 기능을 제공하지 못한다. 하지만 3rd Party로부터의 소프트웨어 입수가 증가하는 소프트웨어 개발 환경을 고려하면 바이너리 형태의 소프트웨어에 대한 오픈소스 및 라이선스의 확인이 필요한 것은 자명하다.

바이너리 형태의 소프트웨어를 스캔하여 오픈소스 사용 여부를 확인할 수 있는 도구는 다음과 같다.

BAT[41] : 오픈소스 기반 (Apache 2.0, 단, 데이터베이스는 상용)
Synopsys Blackduck Binary Analysis[42] : 상용 도구

바이너리 스캔 도구로는 BAT가 오픈소스로 공개되어 있어서 동작 원리 파악이 가능하다. BAT는 분석 대상 바이너리에 대해 hexdump, strings 등의 명령어를 이용하면 String 정보를 추출하고, 이를 오픈소스 데이터베이스 내 String 정보와 비교하여 오픈소스 사용 여부를 판단한다. 이러한 방식은 간단하고, 아키텍처나 컴파일러와 관계없이 사용 가능한 장점이 있지만, 바이너리 분석 특성상 소스 코드를 직접 분석하는 도구에 비해 정확도가 떨어진다.

3.1.2.4. 디펜던시 체크 도구

최근 yocto 프로젝트[43]처럼 빌드 시스템의 발전으로 빌드시 여러 오픈소스 웹사이트에서 소스 코드를 자동으로 다운로드받아 바이너리를 생성하는 경우가 많다. 어떤 모듈을 설치하기 위하여 그 모듈에 디펜던시가 있는 다른 모듈들을 자동으로 설치해준다는 점에서 특히 유용하다. 예를 들면 네트워크 서버 개발시 많이 사용되는 node.js 플랫폼 도 하나의 모듈을 설치하는 경우에 디펜던시가 있는 다른 모듈이 많이 설치된다. 이렇게 각 디펜던시에 해당하는 모듈의 오픈소스 정보를 파악하기에 시간이 많이 소요되기 때문에 이를 자동화하는 도구를 이용하는 것이 효율적이다. 대부분의 디펜던시 체크 도구는 실제로 소스코드를 분석하는 것은 아니고 빌드 시스템에 따라 빌드시 필요한 정보를 기록하는 특정 파일들의 정보를 분석하여 오픈소스 정보를 찾아주는 방법을 이용하고 있다.

41) http://www.binaryanalysis.org
42) https://www.synopsys.com/software-integrity/security-testing/software-composition-analysis.html
43) https://www.yoctoproject.org

대부분의 디펜던시 체크 도구는 오픈소스로 배포되고 있어, 특정 빌드 시스템에 해당하는 디펜던시 체크 도구를 검색하여 찾아보는 것이 꼭 필요하다. 또한 검색하여 찾은 디펜던시 체크 도구는 오픈소스이기 때문에 해당 소스를 직접 수정하여 보완, 적용할 수 있다. 몇 가지 디펜던시 체크 도구 사례를 찾아보면 다음과 같다.

node.js 관련 디펜던시 체크 도구

- https://www.npmjs.com/package/license-checker
- https://www.npmjs.com/package/license-compatibility-checker
- https://www.npmjs.com/package/license-check

maven 관련 디펜던시 체크 도구

- https://github.com/mojohaus/license-maven-plugin
- https://github.com/carlomorelli/licensescan-maven-plugin

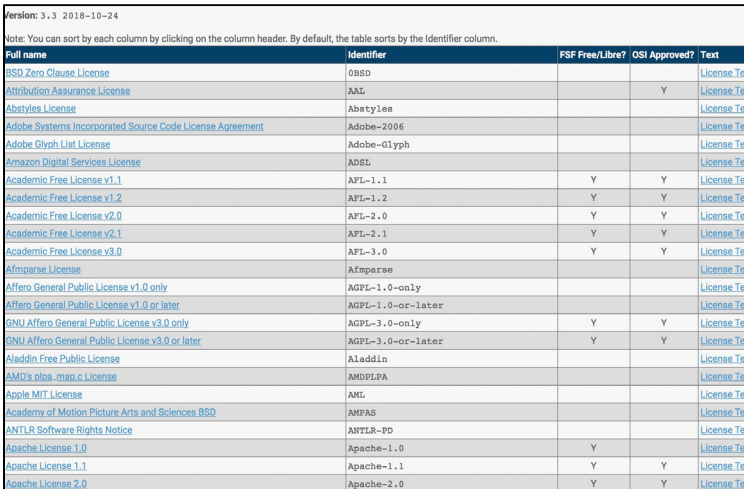
3.2. 라이선스 정보의 표시

대부분의 오픈소스 라이선스들이 요구하고 있는 공통적인 사항은 각각의 오픈소스 소프트웨어에 대한 저작권자 정보와 라이선스 정보를 표시하도록 하고 있다. 하지만 공개되어 있는 소스코드를 보면 저작권 표시와 정확한 라이선스가 명시되어 있지 않는 경우가 있다. REUSE 지침을 개발자들에게 전파하여 소스 코드마다 저작권 및 라이선스 정보를 표기하는 것은 오픈소스 라이선스를 잘 인지하여 준수하게 함으로써 저작권 침해를 줄일 수 있다. 또한 REUSE 도구를 활용하면 REUSE 지침 준수 여부를 쉽게 체크할 수 있어 자동화도 가능하며, 관련 도구를 제공하는 것은 개발자들에게 해당 지침을 사용하도록 권장하기가 쉽다.

3.2.1. SPDX

SPDX는 리눅스 재단의 SPDX 워킹그룹에서 진행하는 오픈소스 내의 저작권 및 라이선스 정보를 교환하기 위한 표준이다. SPDX는 소프트웨어 라이선스 정보를 확인하기 위한 중복 작업을 줄이면서도 오픈소스 라이선스 준수를 더 쉽게 하는 것을 목적으로 한다. SPDX는 소프트웨어에 대한 오픈소스 관련 정보를 명확히 식별할 수 있도록 정보를 제공하기 위하여 오픈소스 소프트웨어에 대하여 패키지 단위, 파일 단위로 라이선스 및 저작권 정보를 확인할 수 있게 한다.

SPDX 라이선스 식별자는 SPDX 문서에서 사용하는 여러 라이선스에 대해 쉽고 효율적으로 식별하는 것을 목적으로 한다. 2018년 10월 24일 기준으로 3.3버전 354개 라이선스가 등록되어 관리되고 있다.[44]



Version: 3.3 2018-10-24

Note: You can sort by each column by clicking on the column header. By default, the table sorts by the Identifier column.

Full name	Identifier	FSF Free/Libre?	OSI Approved?	Text
BSD Zero Clause License	0BSD			License Te
Attribution Assurance License	AAL		Y	License Te
Abstyles License	Abstyles			License Te
Adobe Systems Incorporated Source Code License Agreement	Adobe-2006			License Te
Adobe Glyph List License	Adobe-Glyph			License Te
Amazon Digital Services License	ADSL			License Te
Academic Free License v1.1	AFL-1.1	Y	Y	License Te
Academic Free License v1.2	AFL-1.2	Y	Y	License Te
Academic Free License v2.0	AFL-2.0	Y	Y	License Te
Academic Free License v2.1	AFL-2.1	Y	Y	License Te
Academic Free License v3.0	AFL-3.0	Y	Y	License Te
Afmparse License	Afmparse			License Te
Affero General Public License v1.0 only	AGPL-1.0-only			License Te
Affero General Public License v1.0 or later	AGPL-1.0-or-later			License Te
GNU Affero General Public License v3.0 only	AGPL-3.0-only	Y	Y	License Te
GNU Affero General Public License v3.0 or later	AGPL-3.0-or-later	Y	Y	License Te
Aladdin Free Public License	Aladdin			License Te
AMD's plpa_map.c License	AMDPLPA			License Te
Apple MIT License	AML			License Te
Academy of Motion Picture Arts and Sciences BSD	AMPAS			License Te
ANTLR Software Rights Notice	ANTLR-PD			License Te
Apache License 1.0	Apache-1.0	Y		License Te
Apache License 1.1	Apache-1.1	Y	Y	License Te
Apache License 2.0	Apache-2.0	Y	Y	License Te

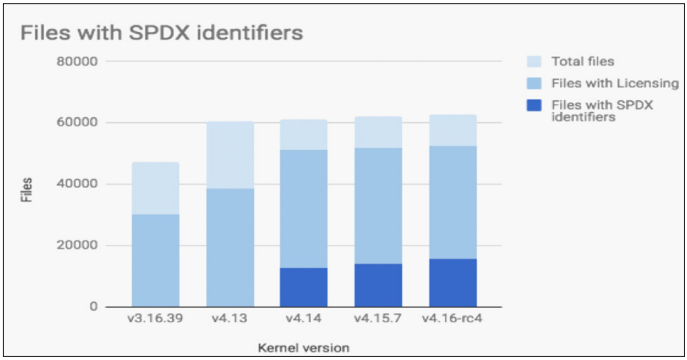
SPDX 라이선스 식별자를 이용하는 것은 라이선스 명칭을 하나로 통일하여 사용함으로써 라이선스를 쉽게 식별할 수 있다는 장점이 있다. 소스 코드를 작성할 때 100여 줄의 긴 라이선스 원문 대신 한 줄의 SPDX 라이선스 식별자로 기록하는 것은 효율적이다. 예를 들어 SPDX에서 Apache License 2.0을 다음과 같이 정리하고 있다.[45]

> 라이선스 명칭 : Apache License 2.0
> 식별자 : Apache-2.0
> 라이선스 웹페이지 :
> http://www.apache.org/licenses/LICENSE-2.0
> http://www.opensource.org/licenses/Apache-2.0
> 정보 : 2004년 1월 릴리즈
> 라이선스 원문 : (생략)
> OSI 인증 : O

최근 리눅스 커널 프로젝트에서 SPDX 라이선스 식별자를 활용하여 소스 코드를 정리한 사례가 있다.[46] 2014년 리눅스 커널 3.16 버전에서 940개 이상의 다른

44) https://spdx.org/licenses/
45) https://spdx.org/licenses/Apache-2.0.html
46) https://lwn.net/Articles/739183/

라이선스 파일이 검출되었고, 2017년 리눅스 4.13 버전에서 1,020개 이상의 다른 라이선스 파일이 검출되었다. SPDX 워킹그룹과 리눅스 커널 개발자들이 협업하여 리눅스 4.14 버전에서 여러 다른 라이선스를 정리하고 SPDX 라이선스 식별자를 추가하는 작업을 진행한 결과 10,000여 개 이상의 파일에 SPDX 라이선스 식별자가 추가되었다.[47]



이후 새로 추가되는 파일에 SPDX 라이선스 식별자가 있는지를 점검할 수 있는 checkpatch.pl라는 스크립트 파일을 제공하고, 리눅스 커널 메인테이너들에게 SPDX 라이선스 식별자를 사용하도록 전파하고 있다. 또한 리눅스 커널 외에도 uboot[48], linaro[49] 등의 주요 오픈소스 프로젝트에서 SPDX 라이선스 식별자를 추가하는 작업을 진행하고 있다.

3.2.2. REUSE를 통한 저작권 및 라이선스 정보의 표시

REUSE는 자유 소프트웨어 재단 유럽 단체에서 진행하는 오픈소스 내 저작권 및 라이선스 표기 가이드를 제공하는 프로젝트이다.[50] REUSE 이전에 소스코드 내에 저작권 및 라이선스 표기에 대한 표준화를 추진했던 사례는 없다. 리눅스 재단에서 오픈소스 컴플라이언스를 전파할 때 FOSSology나 Scancode와 같은 오픈소스 분석 도구 외에 SPDX 및 REUSE를 함께 전파하고 있어 영향력이 더 커질 수 있다. REUSE 프로젝트는 SPDX 라이선스 식별자를 사용하며 이와 같은 표기법으로 작성하였는지 체크해주는 툴을 제공하고 있어, 가이드대로 작성하였는지 여부를 체크하기가 편리하다.

47) Kate Stewart, Philippe Ombredanne & Greg KroahEHartman, "Automating Compliance : Linux Case Study for Solving the Problem at the "Source" " https://spdx.org/news/announcement/2018/04/automating-compliance
48) https://github.com/u-boot/u-boot
49) https://www.linaro.org
50) https://reuse.software

2017년 말에 배포된 소프트웨어 REUSE 실무지침서 2.0버전의 주요 내용은 다음과 같다.[51]

① 사용한 각 라이선스의 정확한 텍스트를 제공할 것

모든 라이선스 텍스트를 프로젝트에 포함시켜야하는데, 이 경우 SPDX 라이선스 식별자와 리포지토리를 활용하는 것이 바람직하다. 라이선스 텍스트를 LICENSE, LICENCE, COPYING, COPYRIGHT의 이름으로 최상위 디렉토리에 포함시킨다. 여러 오픈소스가 사용된 경우 최상위 디렉토리에 licenses 디렉토리를 설정한다.{LICENSE*, LICENCE*, COPYING*, COPYRIGHT*, LICENSES/**}
SPDX 라이선스 식별자를 파일 이름으로 설정하는 것이 바람직하다. 라이선스 텍스트를 가리키는 파일 이름을 변경할 수 없는 경우, 파일 내에 Valid-License-Identifier 태그와 License Text를 삽입한다. SPDX 라이선스 식별자가 없는 경우, LicenseRef-<unique_code> 형태의 식별자를 생성하여 사용한다.

```
Valid-License-Identifier: MIT
License-Text:

MIT License

Copyright (c) <year> <copyright holders>

Permission is hereby granted, free of charge, to any person obtaining a copy of this
software and associated documentation files (the "Software"), etc...
```

라이선스에 대한 예외를 포함시킬 경우 SPDX-Exception-Identifier와 Exception-Text 태그를 활용한다.

```
SPDX-Exception-Identifier: Classpath-exception-2.0
Exception-Text:

 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.
```

51) Version 2.0 of the REUSE practices, released 2017-12-14. https://reuse.software/practices/2.0/

② 각각의 파일에 저작권 정보와 라이선스 정보를 포함할 것

모든 파일의 상단 주석 부분에 저작권자 및 라이선스 정보를 포함할 필요가 있다. 복수의 저작권자가 있는 경우 연도 순서 등으로 표시한다.

```
/*
 * Copyright (c) 2017 Alice Commit <alice@example.com>
 * Copyright (c) 2009-2016 Bob Denver <bob@example.com>
 * Copyright (c) 2007 Company Example <charlie@example.com>
 *
 * SPDX-License-Identifier: GPL-2.0
 */
```

파일의 헤더 부분에 저작권자 및 라이선스 정보를 포함하기 어려운 상황인 경우 다음의 3가지 방법 중 하나를 선택한다.

1. filename.license이라는 텍스트 파일을 제공
2. DEP-5/copyright[52] 포맷으로 제공
3. SPDX 포맷으로 제공

Version control system(VCS)을 통해 저작권 정보를 기록할 수도 있지만, VCS는 통상적으로 저작권(copyright) 정보가 아니라 저작자(authorship) 정보를 기록하므로 주의할 필요가 있다.

```
/*
 * This file is part of project X. It's copyrighted by the contributors
 * recorded in the version control history of the file, available from
 * its original location http://git.example.com/X/filename.c
 *
 * SPDX-License-Identifier: GPL-3.0
 */
```

③ 포함된 소프트웨어에 대한 인벤토리를 제공할 것

LICENSE.spdx라는 하나의 파일로 리포지토리 안 최상위 디렉토리에 포함되어야 한다.

52) https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/

3.2.3. REUSE TOOL

REUSE TOOL은 개발자가 REUSE 권장 사항을 따르도록 하기에 유용하며, 컴플라이언스를 위한 지침서 역할을 하고 오픈소스 정보 생성을 위한 컴파일러 역할을 한다.[53] Python 언어로 작성된 도구로 GPL-3.0 라이선스로 배포되는 오픈소스이다. REUSE 권장 사항과 다르게 라이선스 정보가 없는 파일 목록을 다음과 같이 출력해준다.

```
~/Projects/curl$ reuse lint
.gitattributes
README
docs/libcurl/CMakeLists.txt
lib/.gitattributes
[...]
```

그리고 오픈소스 정보를 다음과 같이 SPDX 형식으로 출력해준다.

```
~/Projects/curll$ reuse compile
SPDXVersion: SPDX-2.1
DataLicense: CC0-1.0
SPDXID: SPDXRef-DOCUMENT
DocumentName: curl
DocumentNamespace: http://spdx.org/spdxdocs/spdx-v2.1-c8c7047c-855c-45a6-bed0-c23900498a79
Creator: Person: Anonymous ()
Creator: Organization: Anonymous ()
Creator: Tool: reuse-0.0.4
Created: 2017-11-15T11:42:28Z
CreatorComment: <text>This document was created automatically using available reuse informatio
[...]
```

이러한 도구를 활용하여 각 소스 코드 내에 저작권 및 라이선스 정보를 정확하게 명시했는지 쉽게 확인할 수 있다. 오픈소스 코드는 기존 독점 소프트웨어에 비해 파생 저작물이 많이 만들어지므로, REUSE 도구를 통해 검증한 소스 코드를 배포하면, 결국 저작권 및 라이선스 정보를 계속 재사용할 수 있다.

53) https://git.fsfe.org/reuse/reuse

4

오픈소스 거버넌스의 구축

4.1 현황 진단

기업에서 오픈소스 거버넌스 구축을 위한 첫 번째 단계는 기업의 오픈소스 소프트웨어 관련 활동에 대한 전반을 진단하는 것으로, 해당 기업의 사업 분야 및 오픈소스 활용 전략 등을 파악하는 것을 포함하고 있다. 사용하는 오픈소스 소프트웨어가 회사 내부용인지 혹은 고객에게 배포되고 있는지 뿐만 아니라, 외부에서 유입되는 소프트웨어 내에 오픈소스 사용 여부도 점검할 필요가 있으며, 그 과정에서 공급사 및 파트너사의 협력 관계 또한 중요할 수 있다.

4.1.1. 사업 분야

회사의 사업 분야에 따라 오픈소스 거버넌스 구축 체계 및 정책이 달라질 수 있다. 예를 들어 어플리케이션과 같이 소프트웨어 자체만을 배포하는 경우와 하드웨어 내에 소프트웨어를 임베디드하여 만든 제품을 배포하는 경우를 생각해보자. 임베디드 제품을 판매하는 회사의 경우 GPL-3.0이나 LGPL-3.0과 같은 라이선스를 사용하는 것은 큰 부담이 될 수 있다. 왜냐하면 GPL-3.0과 LGPL-3.0은 하드웨어에 다시 소프트웨어를 설치할 수 있는 정보를 제공해야 하는 의무 사항이 있는데, 이는 제품 보안과 상충하는 부분이 있기 때문에 고려해야할 사항이 많고, 또 설치 정보를 제공하기 위해 추가 개발이 필요한 경우도 있기 때문이다. 따라서 임베디드 제품을 판매하는 기업들은 위와 같은 이유로 설치 정보 제공의 의무가 있는 오픈소스 라이선스 사용을 금지하는 것을 정책으로 삼기도 한다.

이와 비슷한 관점으로 네트워크 서비스만 제공하는 기업의 경우는 다른 회사에 비해 더욱더 간단하게 프로세스 구축을 할 수도 있다. 왜냐하면 대부분의 오픈소스 라이선스는 “배포”의 개념 안에 네트워크 서비스인 경우를 제외하고 있기 때문에, 네트워크 서비스를 배포로 보는 AGPL-3.0과 같은 라이선스만 유의해서 프로세스를 구축하면 된다. 예를 들어 살펴보자. 웹서비스를 위해 Apache-2.0으로 배포되는 아파치 웹서버와 GPL-2.0으로 배포되는 MYSQL, PHP License로 배포되는 PHP 등의 패키지가 많이 사용되는데, 만약 위와 같은 서버를 구축하여 다른 회사에 배포하는 회사라면, 상대측에 본 서버를 전달하면서, 각각의 라이선스 의무 사항을 준수해야하는 반면, 해당 웹서비스를 직접 구축하여 서비스한다면, 배포에 포함되지 않기 때문에 의무 사항이 발생하지 않게 된다. 따라서 웹서비스만을 하는 회사라면, 네트워크 서비스에도 의무 사항이 있는 오픈소스 라이선스만 유의해서 확인하는 절차를 거칠 수 있다.

또한 회사의 사업 분야가 B2B인지 B2C인지에 따라서도 프로세스 구축에 다른 점이 있을 수 있다. 만약에 B2B와 같이 일반 고객이 아닌 다른 기업을 대상으로

소프트웨어를 배포하는 기업이라면, 오픈소스 컴플라이언스에 대한 상대 기업의 요구 사항을 정확히 파악해야 한다. 최근 자동차 업계를 예로 들면, 자동차 업체에 소프트웨어를 납품하기 위해서는 그 기업의 오픈소스 컴플라이언스 정책에 동의하는 계약 등을 별도로 맺고, 또 오픈소스 적용 계획이나 방법 등에 대한 별도의 문서를 제출하도록 요청하는 기업들이 늘어나고 있다.
그리고 B2C 기업은 고객의 수가 많기 때문에, 소스 제공의 방법으로 웹사이트 구축을 더 선호한다. 왜냐하면 모든 고객에게 직접 소프트웨어를 동봉하여 제공하는 것이 웹사이트 운영보다 더 많은 비용이 든다고 생각하기 때문이다. 반면 B2B 기업은 고객의 수가 더 제한적이기 때문에, 소스 제공을 위하여 웹사이트 구축을 굳이 하지 않아도 될 수 있다.

4.1.2. 오픈소스 소프트웨어 사용 범위

기업 내에서 오픈소스 소프트웨어를 어떤 범위 내에서 사용하는지를 파악하는 것이 필요하다. 예를 들어 오픈소스 소프트웨어를 회사 내부에서만 사용하는 것인지 아닌지를 확인하는 것이다. 대부분의 오픈소스 라이선스 의무 사항은 소프트웨어를 "배포"하는 시점에 부과되기 때문에, 사내에서만 사용하는 소프트웨어인 경우는 의무 사항이 발생하지 않는다. 사내에서만 사용하는 대표적인 예는 사내에서 사용하는 근태 소프트웨어나 개발 과정에서만 사용하고 실제 외부로 배포하는 제품 내에 탑재되지 않는 테스트 프로그램 등이다. 이 경우는 오픈소스 사용 현황만 파악하고 그 이후의 컴플라이언스 절차가 불필요할 수 있다.

4.1.3. 외부 소프트웨어 유입 여부

최근 소프트웨어 복잡도가 높아지면서, 자체적으로 모든 소프트웨어를 직접 개발하는 기업보다 외부에서 소프트웨어를 구매하거나 협업에 의해서 개발하는 경우가 많아졌다. 이처럼 외부 업체에서 소프트웨어가 유입되는지 여부를 먼저 확인하여 만약 외부에서 유입되는 경우가 있다면, 이를 고려하여 프로세스를 구축해야 한다. 왜냐하면, 외부에서 유입된 소프트웨어라 하더라도 해당 소프트웨어 내에 오픈소스 소프트웨어가 포함되어 있다면 오픈소스 라이선스 준수 의무는 일차적으로 해당 소프트웨어를 배포하는 기업에 있기 때문이다.

4.1.4. 소프트웨어 개발 범위 및 규모

소프트웨어 개발 규모나 소프트웨어 개발자 규모에 따라 컴플라이언스를 위한 도구나 자동화 도구를 적용하는 범위가 달라질 수 있다. 또한 직접 개발을 하지 않고,

모두 외부에서 개발한 소프트웨어만 배포하는 경우에도 오픈소스 거버넌스 체계와 이를 위해 도입해야할 도구가 달라질 수 있다. 따라서 이와 같은 현황을 파악하는 것도 기업 진단에 필요한 부분 중의 하나이다.

4.2. 조직 구성

오픈소스 거버넌스 체계를 구축하는 데 있어 조직을 어떻게 구성하고 조직별로 어떤 역할을 정하여 조직 간의 협업이 잘 이루어지게 하는 것은 기업 전체적으로 오픈소스 거버넌스 실행력을 결정하게 하는 중요한 요소이다.

4.2.1. 개발 조직

제품을 개발하면서, 직접 오픈소스 소프트웨어를 찾고 오픈소스 소프트웨어를 제품 내에 포함하여 개발하는 조직으로, 실제 오픈소스 소프트웨어의 다운로드 경로나 개발한 소프트웨어 내의 일부 코드들이 오픈소스 소프트웨어인지 아닌지를 가장 정확히 확인할 수 있는 조직이다. 또한 오픈소스 담당 조직으로부터 라이선스별 의무 사항 및 공개 범위 등에 대한 안내를 받은 후에 실제 소프트웨어 개발 설계 등을 변경하거나 보완하는 것도 직접 수행할 수 있다. 따라서 오픈소스 라이선스별 의무 사항을 잘 이해해야 하고, 또한 오픈소스 라이선스 식별에 대해 가장 중요한 역할을 수행할 수 있는 조직이기도 하다.

4.2.2. 구매 조직

일반적으로 소프트웨어 개발을 직접 하지 않는 조직은 오픈소스 컴플라이언스와 무관하다고 생각할 수 있으나, 구매 조직은 컴플라이언스 체계에 중요한 역할을 갖고 있다. 왜냐하면 외부에서 유입되는 소프트웨어와 관련된 구매의 경우, 구매 조직에서 제품 내 오픈소스 소프트웨어 사용 여부 및 사용한 오픈소스 라이선스 등을 확인하고, 또한 계약서 내에 오픈소스 사용에 있어 기업에 불리하게 진술된 조항이 없는지 등을 점검하거나 법무 조직에 이를 의뢰해야 하기 때문이다.

4.2.3. 법무 조직

법무 조직은 외부로부터 유입되는 소프트웨어 계약을 맺을 때 관련 계약서를 직접 검토하여, 자사 오픈소스 컴플라이언스 정책에 어긋나는 사항은 없는지, 해당 계약서 내용으로 자사 컴플라이언스 준수가 가능한지 아닌지 등을 확인해야 한다.
또한 기업에서 처음 도입하는 오픈소스 라이선스가 발견될 때마다 오픈소스 담당

조직과 함께 검토하여 해당 라이선스가 어떤 의무 사항을 갖는지 그리고 해당 라이선스가 기업의 정책에 맞는지 여부를 확인하고 또 필요에 따라 각 라이선스에 대한 대응 방안들을 함께 결정해야 한다.

4.2.4. 교육 조직

교육 조직은 오픈소스와 관련된 체계적인 교육을 수립하여 교육 대상에 따라 차별적인 교육을 개발하고 운영해야 한다. 이와 같은 교육 체계는 내부에서 직접 진행될 수도 있고, 외부 개발된 온라인 교육 자료나 외부에서 진행되는 교육 등을 수강하게 하는 방법이 될 수도 있다. 중요한 것은 교육 체계를 통하여 기업 내에 오픈소스 관련 지식이 널리 전파될 수 있도록 해야 하는 것이다.

4.2.5. 오픈소스 거버넌스 담당 조직

담당 조직은 오픈소스 거버넌스가 실제로 운영되게 하는 주요한 임무를 수행한다. 법무 조직과 협업하여 오픈소스 정책을 수립하고 오픈소스 거버넌스 체계를 만들어 실제 제대로 운영되는지를 모니터링한다. 또한 전사적으로 오픈소스 라이선스에 대한 인식이 잘 전파될 수 있도록 여러 프로그램을 운영하며, 지속적인 라이선스 컨설팅을 진행해야 한다.

4.3. 전략 수립

오픈소스 소프트웨어를 사용하여 제품을 개발하고 판매하는 기업의 입장에서 기업의 R&D 전략에 맞게 오픈소스 소프트웨어의 활용 방안, 오픈소스 라이선스, 비즈니스 모델 등을 고려한 기업의 오픈소스 거버넌스 전략을 마련할 필요가 있다. 즉 오픈소스 소프트웨어를 활용하여 기업이 달성하고자 하는 목표는 무엇인가를 고려해야 한다. 오픈소스 소프트웨어가 무료이기 때문에 단순한 비용 절감을 위해서인지, 기술의 변화로 인해 표준을 따르기 위한 어쩔 수 없는 선택인지, 빠른 기술 변화로 인하여 생태계를 따라가려는 방안인지 등을 생각해야 한다. 이 밖에 기존 제품의 오픈소스 컴플라이언스를 따르기 위한 전략, 이를 바탕으로 한 기업 이미지, 그리고 오픈소스 소프트웨어 유지 관리 및 개발 비용의 분담 등을 목표로 할 수도 있다.
또한 오픈소스 컴플라이언스를 위해서, 오픈소스 사용에 대한 위험 요소 또한 파악하고 준비 체계를 잘 구축하여 사고를 예방해야 한다. 이를 위해서는 자연스럽게

인력 투입과 또 이에 따른 비용을 필요로 하므로, 최소의 비용으로 최대 예방 효과가 있는 전략이 필요할 수도 있다. 즉, 오픈소스 컴플라이언스를 100% 만족하는 것은 굉장히 큰 비용을 투자할 뿐 아니라 어려우므로, 컴플라이언스 위험과 비용의 상관관계에서 그 합의점을 찾는 것도 전략적인 부분이라 할 수 있다.

4.4. 정책 설정

기업의 오픈소스 전략을 수립한 후에 오픈소스 소프트웨어와 관련된 기업의 정책을 구체적으로 결정해야 한다. 이와 같은 경우 기업의 목표를 명확히 확인하고, 오픈소스 소프트웨어 정책을 기업의 목표와 조화시킬 필요가 있다. 본 정책을 수립할 때, 오픈소스 소프트웨어 유입 방법, 사용 방법, 라이선스 준수 방법 등을 정의하고, 내부에서 해당 정책에 따른 절차를 준수할 수 있게 해야 한다. 즉, 오픈소스 가이드, 업무 프로세스, 라이선스 검증 방법 등 모든 활동이 오픈소스 정책을 기본으로 해야 한다.

오픈소스 정책에 포함될 수 있는 항목들에 대한 예시는 다음과 같다.

정 책 항 목	설 명
오픈소스 사용	사용 가능한 오픈소스 범위 및 승인 절차
오픈소스 교육	오픈소스 사용을 위한 교육 개발 및 운영 방안
오픈소스 라이선스 검증	오픈소스 라이선스 검증 방법 및 조치 방안
소스 코드 공개	오픈소스 라이선스 준수를 위한 소스 공개 방법 및 절차
오픈소스 기여	오픈소스 프로젝트를 생성하거나 외부 오픈소스 프로젝트에 기여하기 위한 방안
외부 업체 관리	외부 업체로부터 공급받는 오픈소스 소프트웨어 관리 방안

리눅스 재단에서 배포하고 있는 “Generic FOSS Policy”를 참고하면, 다음과 같은 정책을 샘플로 소개하고 있으니, 참고 자료로 활용할 수 있다.

● 컴플라이언스 프로세스 내에 포함해야 할 사항
● 배포하는 소프트웨어 내에 외부 소프트웨어를 포함하는 경우
● 서버 소프트웨어의 특별 규정
● 오픈소스 프로젝트에 기여할 때 규정

"Generic FOSS Policy"의 컴플라이언스 프로세스 내에 포함해야 할 사항에대한 정책은 다음을 참고할 수 있다.

회사 산출물에 포함된 모든 오픈소스 소프트웨어 식별

- 식별된 오픈소스 패키지를 사용하기 위해 오픈소스 담당 조직에 사용 허가를 받기 위한 요청서를 제출
- 오픈소스 사용 요청에 대한 검토 진행 (아키텍처 종속성 분석, 식별된 오픈소스 소프트웨어에 대한 출처 분석, 라이선스 식별 및 분석, 지식재산권에 대한 잠재적 영향 분석 등)
- 오픈소스 사용 승인 결정
- 식별한 오픈소스에 대하여 라이선스 의무 사항 확인
- 오픈소스 라이선스에 대한 의무 사항 충족

또 다른 예로 오픈소스 사용에 대한 정책은 다음과 같이 고려할 수 있다. 오픈소스 사용 정책은 길거나 복잡하면 좋지 않으며, 간단한 정책일수록 더욱 효과적이다. "Open Source Compliance in the Enterprise"에서 권장하는 오픈소스 사용 정책에 포함되어야 할 사항은 다음과 같다.

- 개발자들은 제품 내에 오픈소스를 포함할 때 승인을 받는 절차를 포함한다.
- 외부 업체로부터 받은 소프트웨어는 검사를 통하여 오픈소스가 포함되었는지 여부를 꼭 확인하고, 제품 출시 전에 해당 라이선스 의무 사항을 충족할 수 있도록 확인해야 한다.
- 모든 소프트웨어는 외부 업체로부터 받은 소프트웨어 외에도 독점 소프트웨어, 오픈소스 소프트웨어를 모두 검사하고 검토해야 한다.
- 제품은 고객에게 전달되기 이전에 오픈소스 라이선스 의무 사항을 모두 충족해야 한다.
- 오픈소스 사용이 동일하다고 하더라도 한 제품에 대한 승인이 다른 개발 제품의 승인을 의미 하는 것은 아니다.
- 변경된 모든 요소에 대해서 승인 절차를 다시 거쳐야 한다.

다음 표는 GENIVI[1]에서 제공하는 오픈소스 정책 문서를 요약해 놓은 사항이다. 사용 허여되는 오픈소스 라이선스, 사용 불가한 오픈소스 라이선스, 예외적으로 허용되는 오픈소스 라이선스에 대해서 각각 신호등으로 나타내어 관리하는 것을 알 수 있다. GENIVI의 정책을 참고하여 각 기업에서도 사용 허가 여부에 따라 라이선스를 분류해놓으면 개발자 입장에서 제품 내에 해당 라이선스 사용 여부를 쉽게 파악할 수 있기 때문에 효율적일 수 있다.

분류	설명	예
Green-light License	GENIVI의 리뷰를 거쳐 제한없이 사용될 수 있는 것으로 인정받은 라이선스	MPL-2.0, Apache-2.0, MIT, BSD-3-Clause
Red-light License	사용 금지된 라이선스	GPL-3.0, LGPL-3.0, BSD-4-Clause
Orange-light License	어떤 경우에 허용되는 라이선스로 사전 리뷰를 거쳐 사용할 수 있는 라이선스	GPL-2.0, LGPL-2.1, Public Domain

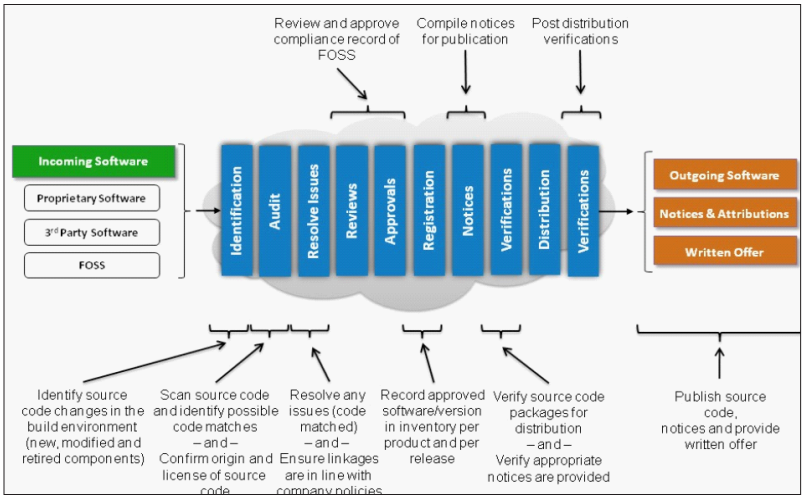
1) GENIVI 는 차량 인포테인먼트(IVI: In-Vehicle Infotainment) 오픈소스 개발과 커넥티드 카를 위한 기술을 위한 비영리 협의체이다. (https://www.genivi.org/)

4.5. 프로세스 구축

오픈소스 거버넌스 프로세스는 크게 몇 가지로 나누어 볼 수 있다. 여러 기업에서 주로 처음 구축하는 프로세스가 오픈소스 컴플라이언스 프로세스이다. 그 후 오픈소스 사용의 성숙도가 높아짐에 따라 필요성이 대두되는 오픈소스 기여 프로세스, 그리고 이와 더불어 사내외 오픈소스 관련 요청 및 문의에 대해 프로세스를 구축하면 도움이 된다.

4.5.1. 컴플라이언스 프로세스

오픈소스 컴플라이언스 프로세스는 리눅스 재단에서 제안하고 있는 프로세스가 가장 널리 전파되어 있으며, 여러 기업이 이를 기반으로 각 기업의 프로세스를 구축하고 있는 것으로 보인다. 하기 프로세스처럼 단계별 확인할 사항 및 최종 결과물이 제대로 나올 수 있다면, 각 기업의 특색에 맞게 최적화하여 프로세스를 생략 또는 수정하는 것도 괜찮아 보인다.

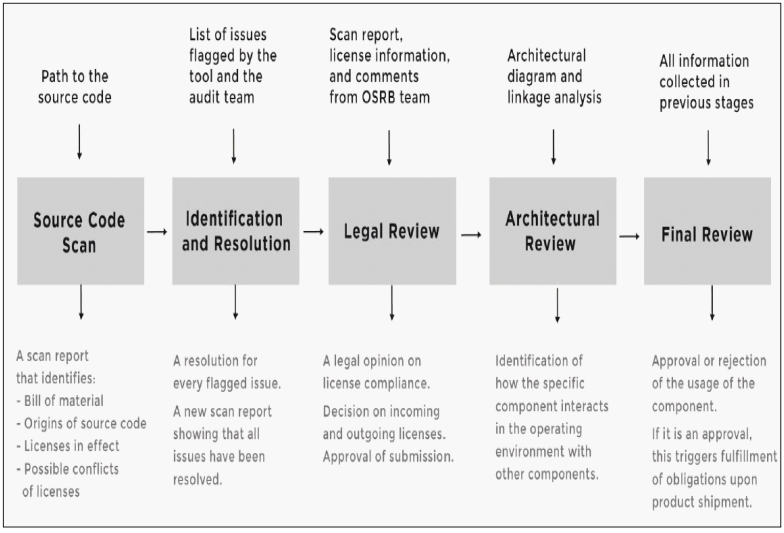


[출처 : Open Source Compliance in the Enterprise]

각각의 단계를 거치면서 나오는 산출물은 다음 표와 같이 정리할 수 있다. 각 기업에 맞게 각각의 단계를 통합하거나 생략할 수 있지만, 각 산출물은 모두 포함할 수 있도록 하는 것이 좋다.

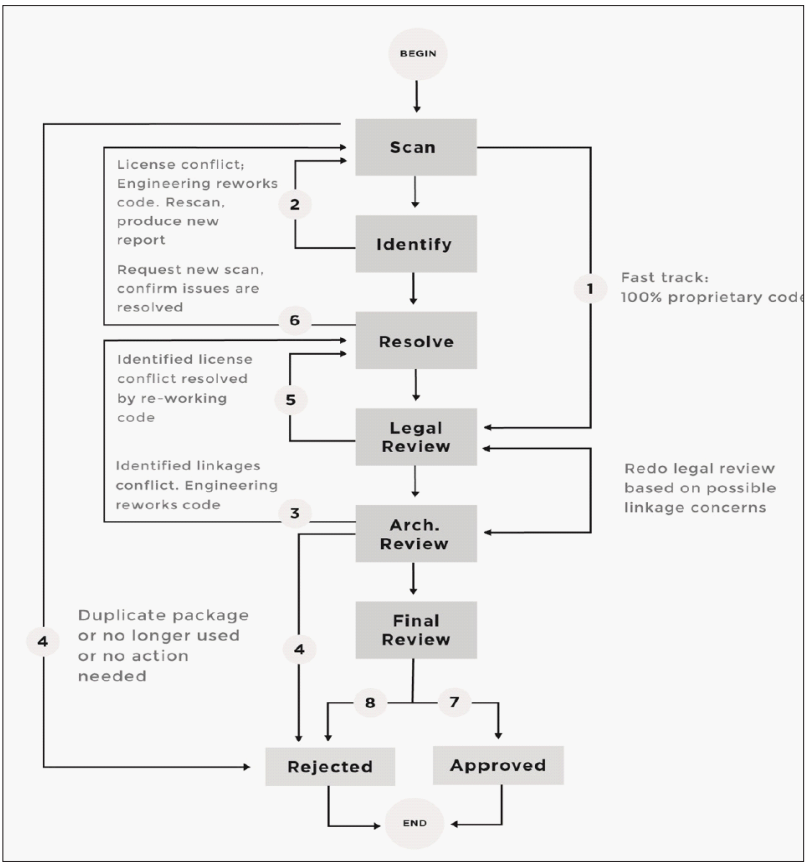
단 계	산 출 물
Identification	리뷰를 위한 오픈소스 식별
Audit	소스 코드 출처와 라이선스 식별한 결과
Resolve Issues	회사 정책에 맞게 이슈 해결
Reviews	Audit Report 결과 확인 및 해결된 이슈 표시
Approvals	승인된 오픈소스 버전 및 라이선스 의무 사항 명시
Registration	승인된 오픈소스 내역 등록
Notices	적절한 오픈소스 고지문 준비
Verification	배포를 위한 소스 패키지 확인 및 적절한 고지문이 제공되는지 확인
Distribution	소스 코드를 제공하기 위한 의무사항 충족
Verification	배포되는 산출물이 적절한지 확인
최종	소스 코드, 고지문 공개, 약정서 제공

하기 프로세스는 “Open Source Compliance in the Enterprise”에서 제안하는 프로세스별 입력과 산출물이다. 앞의 각 단계별 산출물 표와 다음 프로세스별 산출물을 참고하여 각 단계에서 각 조직의 역할을 좀 더 구체적으로 생각해볼 수 있다.



[출처 : Open Source Compliance in the Enterprise]

다음은 컴플라이언스 프로세스를 진행하면서 발생할 수 있는 여러 케이스를 고려하여 도표화한 것인데, 이처럼 기업에서 발생할 수 있는 경우들을 이해하기 쉽게 정리하여 전파하는 것도 좋은 사례라 할 수 있다.
하기 경우는 소스 코드를 분석한 후에 발생할 수 있는 7가지 경우에 대해서 어떠한 과정을 거쳐 다시 승인될 수 있는지 혹은 최종적으로 반려될 수 있는지를 보여주고 있다.



[출처 : Open Source Compliance in the Enterprise]

4.5.2. 기여 프로세스

기업 내의 오픈소스 사용 성숙도가 높아지게 되면, 기업에서 개발한 부분에 대하여 오픈소스 프로젝트에 기여하거나 새로운 오픈소스 프로젝트를 만들 수도 있다. 기업에서는 다음과 같은 프로세스를 기반으로 오픈소스 기여에 대한 프로세스를 자체적으로 구축할 필요가 있다. 개발팀에서 오픈소스 기여에 대한 필요성 등을 요청하여 OSEC[2]와 OSRB[3]에서 검토하여 결정을 내리게 된다.

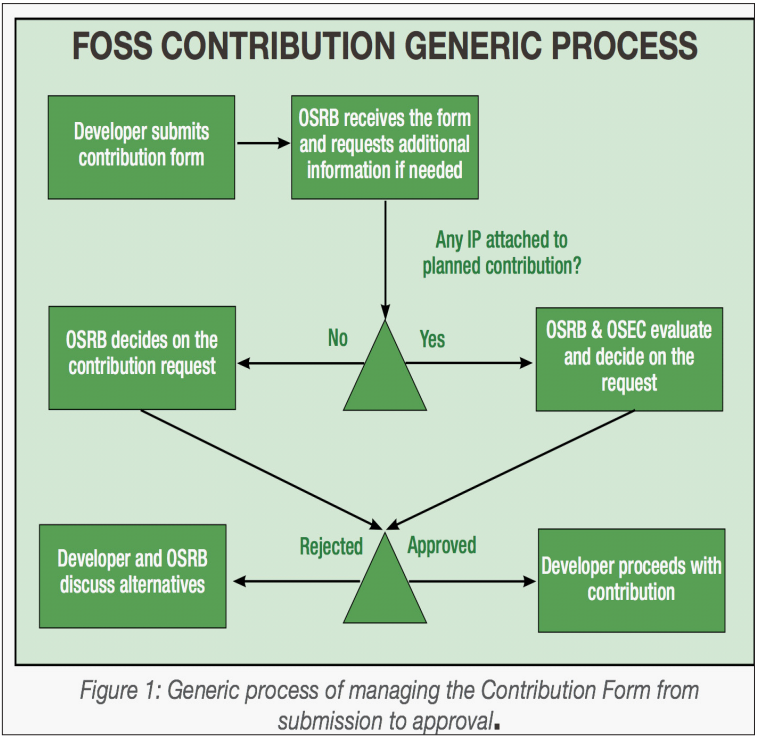


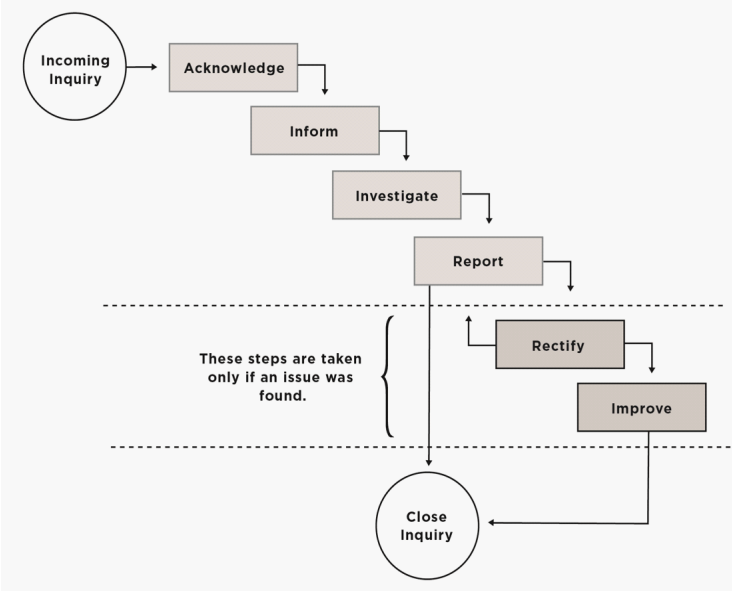
Figure 1: Generic process of managing the Contribution Form from submission to approval.

[출처 : Generic OSRB Contribution Form]

4.5.3. 문의 대응 프로세스

오픈소스 문의 및 요청에 대응하는 프로세스는 다음과 같이 구축할 수 있다. 하기 사항은 사내 문의일 수도 있고, 외부에서 들어온 문의일 수도 있으며, 중요한 것은 이러한 문의에 대한 대응을 하면서, 기업 내의 프로세스 보완이 필요한 사항을 발견할 때마다 수정 작업을 진행해야 한다는 것이다.

2) Open Source Executive Committee : 오픈소스 의사 결정 위원회
3) Open Source Review Board : 오픈소스 검토 위원회



[출처 : Open Source Compliance in the Enterprise]

2019년 6월 발행

본 교재는 집필자의 견해를 존중하여 발행하였습니다.
따라서 본 교재상의 견해는 한국저작권위원회의 공식적인 견해와는 다를 수 있습니다.