

오픈 소스 프로젝트에 BSD 형식의 라이선스를 사용해야 하는 이유

Bruce Montague <brucem@alumni.cse.ucsc.edu>

Revision: [filedate](#)

FreeBSD는 FreeBSD 재단의 등록 상표입니다.

Intel, Celeron, Centrino, Core, EtherExpress, i386, i486, Itanium, Pentium, 그리고 Xeon은 미국 및 그 이외의 국가에서 Intel Corporation 혹은 그 자회사의 상표 또는 등록 상표입니다.

생산자 및 판매자들이 그들의 제품을 구별하기 위해 사용하는 많은 표기들은 상표권으로 보호받습니다. 이 문서에 그런 표기들이 나타날 때, FreeBSD 프로젝트는 그러한 상표권에 대해 알고 있기 때문에, 해당 표기 뒤에 “™” 또는 “®” 기호를 사용합니다.

2018-09-29 15:47:35 UTC by root.

차례

1. 개요	1
2. 오픈 소스의 아주 간략한 역사	1
3. BSD 라이선스의 관점에서 바라본 유닉스	2
4. FreeBSD 라이선스와 BSD 라이선스의 현재 상황	2
5. GPL의 기원	3
6. 리눅스와 LGPL의 기원	3
7. 오픈 소스 라이선스와 방치 문제	4
8. 라이선스가 할 수 없는 일	4
9. GPL의 장단점	4
10. BSD의 장점	5
11. BSD 라이선스를 사용하면 좋은 경우	6
12. 맺음말	7
13. 추가 정보	7

1. 개요

이 문서는 소프트웨어와 데이터에 BSD 형식의 라이선스를 사용해야 하는 이유를 설명합니다; 구체적으로 이 문서는 GPL 대신 BSD 형식의 라이선스를 사용할 것을 권장합니다. 이 문서는 또한 BSD와 GPL 오픈 소스 라이선스에 대한 소개 및 간단한 요약이기도 합니다.

2. 오픈 소스의 아주 간략한 역사

“오픈 소스”라는 용어가 사용되기 오래 전에, 소프트웨어는 프로그래머들의 자발적인 공동체에 의해 개발되고 자유롭게 주고받아졌습니다. 1950년대 초반을 시작으로, [SHARE](#)와 [DECUS](#)와 같은 단체들은 컴퓨터 하드웨어 회사들이 자사의 제품에 동봉한 소프트웨어의 상당수를 개발했습니다. 그 당시 컴퓨터 회사들은 주로 하드웨어 사업을 하고 있었습니다; 소프트웨어의 비용을 낮추고 더 많은 프로그램들을 사용 가능하도록 제공하는 것은 곧 하드웨어 회사들의 경쟁력이 되었습니다.

이러한 방침은 1960년대에 바뀌었습니다. 1965년에 ADR은 하드웨어 제조사와는 독립적으로 라이선스된 최초의 소프트웨어 제품을 개발했습니다. ADR은 IBM의 고객들에 의해 개발된 무료 IBM 패키지와 경쟁하고 있었습니다. ADR은 1968년에 그들의 소프트웨어에 대한 특허를 출원했습니다. 그들의 프로그램이 (불법으로)

공유되는 것을 막기 위해, 그들은 그들의 프로그램을 (사용하기 위해서는 비용을 지불해야 하는) 임대 형식의 장비에서만 제공했습니다. 그럼으로써 ADR은 소프트웨어에 대한 소유권을 얻었고, 소프트웨어의 재배포와 재사용을 하지 못하도록 통제할 수 있었습니다.

1969년에 미국 법무부는 IBM이 그들의 하드웨어에 무료 소프트웨어를 동봉하여 시장을 독점하려 한 것에 대해 벌금을 부과했습니다. 이 소송의 결과로, IBM은 더 이상 그들의 제품에 소프트웨어를 동봉하지 않았습니다; 즉, 소프트웨어는 하드웨어와는 독립적인 제품이 되었습니다.

1968년에 Informatics는 최초로 상용 killer-app을 발표하고 소프트웨어 제품, 소프트웨어 회사, 그리고 많은 이윤에 대한 개념을 빠르게 정착시켜 나갔습니다. Informatics는 현재 컴퓨터 산업의 표준과도 같은 영구 라이선스를 개발했습니다. 이 라이선스는 소프트웨어의 소유에 대한 권리가 소비자에게 주어지지 않는다는 특징이 있습니다.

3. BSD 라이선스의 관점에서 바라본 유닉스

원본 유닉스 구현을 소유하고 있던 AT&T는 반독점법에 의해 공식적으로 규제를 받고 있었습니다; 그 회사는 합법적으로 소프트웨어 제품을 판매할 수 없었습니다. 그러나 학술 기관에 저장 매체의 비용만을 받고 소프트웨어를 제공하는 것은 가능했습니다.

OS conference에서 유닉스를 사용 가능하다고 발표한 이후에, 대학교들은 그것을 빠르게 채택했습니다. 유닉스는 대단히 저렴한 16비트 컴퓨터인 PDP-11에서 작동했고, 시스템 프로그래밍에 명백하게 유리한 고급 언어로 작성되어 있었습니다. 이는 대단히 유용한 특징이었습니다. DEC PDP-11은 소비자들이 그들만의 운영 체제를 작성(그 당시에는 꽤나 보편적인 일이었습니다)하기 쉽도록 하드웨어 인터페이스를 공개하고 있었습니다. DEC의 창립자인 Ken Olsen이 했던 말은 유명합니다, “당신이 좋은 하드웨어를 가지고 있다면 소프트웨어는 하늘이 내려 준다”.

유닉스의 개발자인 Ken Thompson은 1975년에 그의 모교인 캘리포니아 버클리 대학교(UCB)로 돌아왔고, 커널을 한줄씩 가르쳤습니다. 이는 궁극적으로 BSD(Berkeley Standard Distribution)라는 시스템으로 발전했습니다. UCB는 유닉스를 32비트로 변환시켰고, 가상 메모리와 TCP/IP 스택(인터넷의 핵심 기술)을 구현했습니다. UCB는 BSD를 “BSD 라이선스”라고 알려지게 된 규정에 따라 저장 매체의 비용만 받고 누구에게나 제공했습니다. 소비자들은 AT&T에서 유닉스를 구입했고 UCB에서 BSD 테이프(당시의 저장 매체)를 주문했습니다.

1980년대 중반에 정부의 ATT에 대한 독점 소송은 ATT가 회사의 사업부를 분할하는 것으로 끝났습니다. ATT는 여전히 유닉스를 소유하고 있었고 이제는 판매도 할 수 있었습니다. ATT는 공격적인 라이선스 정책을 추진했으며 오늘날의 주요한 상용 유닉스들은 ATT 기반이 되었습니다.

1990년대 초반에 ATT는 BSD에 관련된 라이선스의 위반으로 UCB에 소송을 제기했습니다. UCB는 ATT가 자신의 제품에 BSD의 향상된 기능을 아무런 승인이나 대가 지불 없이 포함시킨 것을 발견했고, 주로 ATT와 UCB 사이의 기나긴 소송 사건이 잇따라 발생했습니다. 이 기간 동안 UCB의 프로그래머들 중 일부는 BSD에 포함된 ATT 코드를 모두 다시 작성하는 작업에 착수했습니다. 이 작업의 결과로 bsd 4.4-lite라고 불리는 시스템이 만들어졌습니다 (이것은 완전한 시스템이 아니었기 때문에 lite라는 표현이 사용되었습니다; 이는 6개의 핵심적인 ATT 파일이 빠졌기 때문입니다).

얼마 후에, Dr. Dobbs의 잡지에 BSD 파생의 386 PC 버전 유닉스를 설명하는 긴 글이 수록되었습니다. 이것은 4.4 lite에 결손된 6개의 파일들을 BSD 라이선스의 것으로 대체한 버전이었습니다. 386BSD라고 이름지어진 이 시스템은 전 UCB 프로그래머인 William Jolitz에 의한 것이었습니다. 이것은 오늘날 사용되는 PC 호환 BSD들의 기본적인 바탕이 되었습니다.

1990년대 중반에, Novell이 ATT의 유닉스에 대한 권리를 인수했고 (그 당시 비밀이었던) 협정에 의해 소송은 종료되었습니다. UCB는 곧 BSD에 대한 지원을 종료했습니다.

4. FreeBSD 라이선스와 BSD 라이선스의 현재 상황

지난 몇 년동안 FreeBSD에 적용되어 온 소위 [new BSD license](#)는 여러분이 프로그램 또는 그 소스를 가지고 무엇이든 해도 되지만, 아무런 보증도 되지 않으며 저작자 중 아무도 책임지지 않음(즉, 여러분은 아무도 고소할 수 없습니다)을 나타내는 사실상 하나의 선언입니다. 이 new BSD license는 제품의 상용화를 장려하고 있습니다.

니다. 어떤 BSD 코드라도 여러분의 코드의 사용 가능성 또는 여러분의 미래의 행동에 아무런 제약을 받지 않고 판매되거나 상용 제품에 포함될 수 있습니다.

new BSD 라이선스를 “공공재”와 혼동하지 마십시오. 공공재에 속한 대상 역시 모두가 무료로 사용할 수 있지만, 그것은 소유자가 없습니다.

5. GPL의 기원

1980년대 말과 1990년대 초에 유닉스의 미래가 혼란스러웠을 동안, 또 다른 주요 라이선스인 GPL이 개발의 결실을 맺었습니다.

Emacs의 개발자인 Richard Stallman이 MIT의 스태프 멤버였을 때, 그의 연구실은 시스템을 직접 개발하는 방식에서 상용 제품을 사용하는 방식으로 바꾸었습니다. Stallman은 그가 합법적으로 시스템을 개선할 수 없다는 사실을 알고 화가 났습니다. (Stallman의 동료들 대다수는 MIT에서 개발하고 라이선스한 소프트웨어를 기반으로 하는 두 회사를 창업하기 위해 떠났습니다; 이 소프트웨어의 소스 코드에 대한 관점에서 의견의 불일치가 있었던 것으로 보입니다). Stallman은 상용 소프트웨어 라이선스에 대한 대안을 마련했고 이를 GPL, "GNU Public License"라고 이름지었습니다. 그는 또한 [Free Software Foundation](#) (FSF)라고 불리는 비영리 단체를 만들었는데, 이는 상용 라이선스에 종속적이지 않으면서 관련 소프트웨어를 포함하는 완전한 운영 체제 개발을 목표로 했습니다. 이 시스템은 "GNU is Not Unix"라는 뜻의 GNU라고 불렸습니다.

GPL은 표준적인 상용 라이선스의 정반대 역할을 하도록 디자인되었습니다. 이 관점에서, GPL 프로그램에 어떤 수정이라도 가하면 (사용자에게 소스 코드를 제공하도록 요구하는 방식으로) GPL 커뮤니티에 환원해야 하고 GPL 코드를 사용하거나 링크한 프로그램은 모두 GPL 라이선스를 사용해야 합니다. GPL은 소프트웨어가 상용 제품이 되는 것을 막도록 의도하고 있습니다. GPL 본문의 마지막 문단은 다음과 같이 말하고 있습니다:

"General Public License는 여러분의 프로그램을 상용 프로그램에 포함시키는 것을 허용하지 않습니다." [1]

GPL은 복잡한 라이선스이기 때문에 GPL을 사용하는 데에는 몇 가지 원칙이 있습니다:

- 여러분은 소프트웨어를 배포하고, 지원하고, 관련된 문서에 대해 필요한 요금을 원하는 만큼 책정할 수 있지만, 소프트웨어 그 자체를 판매할 수는 없습니다.
- 프로그램을 컴파일하는 데 GPL 코드가 필요하다면, 그 프로그램은 GPL을 따라야 합니다. GPL 라이브러리에 정적으로 링크된 프로그램도 GPL을 따라야 합니다.
- GPL은 GPL 소프트웨어 및 연관된 모든 개발품을 누구든지 자유롭게 사용할 수 있도록 라이선스할 것을 요구하고 있습니다.
- 단순히 소프트웨어들을 한데 모아 놓는 것은, 예를 들어 여러 프로그램들이 하나의 디스크에 저장되는 경우, GPL 프로그램을 비 GPL 프로그램에 포함시키는 것으로 간주되지 않습니다.
- 프로그램의 출력 결과는 파생 작업으로 간주되지 않습니다. 이는 gcc 컴파일러를 법적 문제 없이 상용 환경에 사용할 수 있도록 해 줍니다.
- 리눅스 커널이 GPL이기 때문에, 리눅스 커널에 정적으로 링크된 코드는 모두 GPL을 따라야 합니다. 동적으로 링크되고 로드할 수 있는 커널 모듈을 통해 이 요구사항을 우회할 수 있습니다. 이 허용은 회사로 하여금 바이너리 드라이버를 배포할 수 있도록 해 주지만, 특정한 버전의 리눅스 커널에서만 동작하게 된다는 단점도 있습니다.

이 복잡성 때문에, 오늘날 리눅스 및 그와 연관된 소프트웨어에서는 GPL의 법적 효력이 무시되고 있습니다. 이에 대한 장기적인 결과는 불분명합니다.

6. 리눅스와 LGPL의 기원

상용 유닉스 전쟁이 벌어지는 동안, 리눅스 커널은 PC 유닉스의 클론으로 개발되었습니다. Linus Torvalds는 리눅스의 탄생에 대한 공로를 GNU C 컴파일러 및 그에 연관된 GNU 도구들에 돌리고 있습니다. 그는 리눅스 커널을 GPL로 만들었습니다.

GPL은 GPL 코드에 정적으로 링크된 모든 코드 역시 GPL을 따르도록 요구하고 있다는 사실을 기억해 주십시오. 즉 해당 프로그램의 소스 코드가 사용자에게 제공되어야 합니다. 그러나 동적 링크는 GPL 위반으로 간주되지 않습니다. 리눅스에서 상용 프로그램을 사용하는 경우가 점점 더 많아졌습니다. 이러한 프로그램들은 대체로 시스템 라이브러리와 링크되어야 합니다. 이 결과로 **LGPL** ("Library", 나중에 "Lesser"로 다시 명명된, GPL)라고 하는 GPL의 수정판이 나타났습니다. LGPL은 상용 코드를 GNU C 라이브러리인 glibc와 링크하는 것을 허용합니다. 여러분은 LGPL 라이브러리와 동적으로 링크된 코드를 공개해야 할 의무는 없습니다.

만약 여러분이 임베디드 시스템에서 흔히 요구되는 것처럼 프로그램과 glibc를 정적으로 링크하고자 한다면, 해당 프로그램을 사유 재산으로 유지할 수는 없습니다. 즉, 소스 코드는 반드시 공개되어야 합니다. GPL과 LGPL 모두 해당 코드를 직접 수정한 것은 공개하도록 요구하고 있습니다.

7. 오픈 소스 라이선스와 방치 문제

사유 소프트웨어와 연관된 중요한 문제들 중 하나는 “방치”(orphaning)이라고 알려져 있습니다. 이는 하나의 이는 하나의 사업 실패나 제품 전략 변경이 피라미드 형태로 이에 종속된 많은 시스템과 회사들을 그들이 대처할 수 있는 범위 너머의 이유로 망하게 할 때 일어납니다. 수십 년에 걸친 경험은 잠깐 성공한 소프트웨어 공급자가 그 소프트웨어를 언제까지나 사용 가능하게 해 주지는 않을 것이라는 사실을 보여 주었습니다. 이는 현재의 시장 조건과 전략이 빠르게 변화할 수 있기 때문입니다.

GPL은 사유 지적 재산과의 링크를 막음으로써 방치 현상을 방지하고자 하고 있습니다.

BSD 라이선스는 중소기업에게 아무런 법적 의무 또는 비용 없이 소프트웨어를 맡겨 놓는 것과 같은 기능을 합니다. 만약 BSD 라이선스 프로그램의 개발이 중단되면, 회사는 그에 종속적인 프로그램을 단순히 사유 재산을 넘겨받듯 계속 사용할 수 있습니다. 보다 나은 상황은 BSD 라이선스의 코드가 비공식 협회에 의해 유지되는 것입니다. 이러면 개발 과정이 하나의 회사 또는 제품 라인의 생존에 종속적이지 않게 됩니다. 개발 팀이 지속적으로 이어지는 것은 단순히 소스 코드를 얻을 수 있는가의 여부보다 훨씬 더 중요합니다.

8. 라이선스가 할 수 없는 일

어떠한 라이선스도 미래의 소프트웨어 사용 가능성을 보장해 주지는 않습니다. 저작권자가 언제든지 저작권의 내용을 바꿀 수 있지만, BSD 커뮤니티에서 그러한 시도가 있다면 단순히 소스 코드를 fork하게 됩니다.

GPL은 라이선스를 무효로 하는 것을 명시적으로 금지하고 있습니다. 그런데 그것이 실제로 일어났습니다. 회사(Mattel)가 GPL 저작권을 인수(cphack)하고, 저작권 전체를 무효화시킨 뒤, 법정에 가서, 승소한 경우[2]입니다. 즉, 그들은 합법적으로 해당 배포본 전체와 모든 파생물에 대한 저작권을 무효화시켰습니다. 더 크고 널리 퍼져 있는 배포본에 대해서도 이런 일이 일어날 수 있는지는 알 수 없습니다; 뿐만 아니라 특정 소프트웨어가 진짜 GPL인지에 대한 혼란이 있기도 합니다.

다른 예시로는, Red Hat이 FSF 컴파일러 도구들의 개발을 인수한 기술 회사인 Cygnus를 인수한 경우가 있습니다. Cygnus는 그들이 GNU 소프트웨어에 대한 지원을 판매하는 사업 모델을 개발했기 때문에 이렇게 할 수 있었습니다. 이는 그들로 하여금 50여 명의 엔지니어들을 고용하고 많은 수정을 할 수 있는 우세함을 통해 프로그램의 개발 방향을 원하는 대로 할 수 있게 하였습니다. Donald Rosenberg가 언급하기를 "GPL과 같은 라이선스를 사용하는 프로젝트들은 누군가 더 나은 코드를 만들어 원 소유자에 비해 빠르게 일을 해나가는 방법을 프로젝트를 손에 넣을지도 모른다는 지속적인 위협 속에서 살아간다." [3]

9. GPL의 장단점

GPL을 사용하는 흔한 이유 중 하나는 gcc 컴파일러를 수정하거나 확장할 때입니다. 이것은 모든 소프트웨어의 비용이 비싼 데 비해 결과로 만들어진 컴파일러를 다른 사람이 사용할 가능성이 거의 없는 CPU를 개발할 때 특히 적합합니다.

GPL은 CD를 판매하는 작은 회사들에게도 매력적일 수 있습니다. 이윤을 충분히 남기면서도 사용자에게 비싸지 않은 제품을 제공할 수 있습니다. GPL은 또한 GPL 지적 재산에 대한 문서 등의 다양한 기술 지원을 제공하는 회사들에게도 매력적입니다.

GPL의 잘 알려지지 않고 의도적이지도 않은 사용은 대기업들이 소프트웨어 회사들의 가치를 싸게 평가하는데 좋다는 것입니다. 다시 말해, GPL은 잠재적으로 전체의 경제적 이익을 저해하고 독과점에 기여하는 마케팅 무기에 적합합니다.

GPL은 소프트웨어를 상업화하고 이윤을 창출하고자 하는 사람들에게는 현실적인 문제가 될 수 있습니다. 예를 들어, GPL은 대학원생이 자신의 연구 결과를 상업화하기 위해 회사를 차리는 것을 어렵게 하거나, 해당 학생이 연구 결과를 상업화해줄 것으로 기대하는 회사에 입사하는 것을 어렵게 합니다.

여러 소프트웨어 표준들의 정적 링크 구현을 사용해야 하는 사람들에게 GPL은 좋지 못한 라이선스인데, 이는 상용으로 구현된 표준의 사용을 막기 때문입니다. 그래서 GPL은 GPL 표준을 사용해야 빌드할 수 있는 프로그램의 수를 최소화하고 있습니다. GPL은 하나의 상용 제품이 표준이 되는 방식을 막으려는 의도를 가지고 있습니다. (리눅스 애플리케이션에는 적용되지 않는데, 이는 그것들이 정적 링크 대신 trap-based API를 사용하기 때문입니다.)

GPL은 프로그래머들이 프로그램의 발전에 기여하고, 이들의 배포와 지원으로 경쟁하도록 하는 것을 시도하고 있습니다. 이 상황은 필요한 많은 코어 시스템 표준에는 현실적이지 않은데, 이는 이러한 시스템이 기존의 비 GPL 라이선스 표준으로 상용화되거나 그러한 표준과 결합되는 환경에 적용될 수 있기 때문입니다. 실시간 시스템들은 대개 정적으로 링크되기 때문에, 많은 임베디드 시스템 회사들에게 GPL과 LGPL은 단연코 잠재적 문제점으로 여겨집니다.

GPL은 연구 및 개발 단계에서, 수요에 관계없이, 노력을 유지시키기 위한 시도입니다. 이는 더 널리 배포하는데 알 수 없는 비용이 드는 연구자 및 개발자들에게 최대한의 혜택을 제공합니다.

GPL은 연구 결과가 상용 제품으로 변모하는 것을 막기 위해 고안되었습니다. 이는 고전적인 기술이 마침내 다 다르게 되는 종착역과 같으며 이렇게 상용화되는 것을 막는 것은 일반적으로 어렵습니다; GPL은 그러한 과정을 봉쇄하도록 만들어졌습니다.

10. BSD의 장점

BSD 형식의 라이선스는 장기간의 연구 또는 다음과 같은 개발 환경이 필요한 프로젝트에 이상적입니다:

- 거의 비용이 들지 않는 경우
- 오랜 기간 동안 발전할 경우
- 누구나 법적 제약 없이 최종 결과물을 상용화하는 것을 허용하고자 할 경우

마지막 조건은 가장 지배적인 조항인데, 이는 Apache 프로젝트가 그들의 라이선스에 중점을 둔 사항이기도 합니다:

“This type of license is ideal for promoting the use of a reference body of code that implements a protocol for common service. This is another reason why we choose it for the Apache group – many of us wanted to see HTTP survive and become a true multiparty standard, and would not have minded in the slightest if Microsoft or Netscape choose to incorporate our HTTP engine or any other component of our code into their products, if it helped further the goal of keeping HTTP common... All this means that, strategically speaking, the project needs to maintain sufficient momentum, and that participants realize greater value by contributing their code to the project, even code that would have had value if kept proprietary.”

개발자들은 BSD 라이선스를 선호하는 경향이 있는데, 이는 그들이 코드를 다룸에 있어 법적 분쟁으로부터 벗어나 그들이 하고 싶은 대로 하도록 허락하기 때문입니다. 반면, 시스템을 개발하기보다 주로 사용할 것으로 예상되는 사람들, 혹은 다른 사람들이 코드를 개선해 주기를 기대하는 사람들, 혹은 (국가 공무원과 같이) 시스템 작업으로 생계를 유지하지는 않는 사람들의 경우 GPL을 선호하는데, 이는 다른 사람이 개발한 코드를 자신이 사용할 수 있고 그들의 상관이 저작권을 가지고 있는 일을 막을 수도 있으며 결과적으로 잠재적으로 소프트웨어를 “무용지물”로 만들거나 방치하지 않도록 할 수 있기 때문입니다. 만약 여러분이 경쟁자로 하여금 여러분을 돕도록 강제하고 싶다면, GPL은 매력적인 선택지가 될 것입니다.

BSD 라이선스는 단순한 선물인 것은 아닙니다. “왜 우리는 우리의 경쟁자를 돕거나 그들이 우리가 만든 결과물을 훔쳐 가도록 내버려 두어야 하나요?”라는 질문은 BSD 라이선스에 관련된 질문에서 자주 볼 수 있습니다.

BSD 라이선스 하에서, 만약 어떤 회사가 제품 시장을 독점하고 다른 이들이 이를 전라적이었다고 여길 경우, 다른 회사들은 최소한의 노력만으로 시장 경쟁과 공정성을 증대시킬 경쟁 BSD 파생본에 기여하는 작은 협회를 만들 수 있습니다. 이는 각 회사들로 하여금 이것이 제공하는 이점으로부터 이윤을 창출할 수 있을 것이라는 믿음을 주고, 경제적 유연성과 효율성에도 기여할 수 있습니다. 회원들이 더 빠르고 쉽게 이에 협력할수록, 더욱 성공적일 것입니다. BSD 라이선스는 그러한 행동을 가능하게 해 주면서도 가장 덜 복잡한 라이선스입니다.

완전하고 경쟁력 있는 오픈 소스 시스템을 단지 저장 매체의 가격만으로 널리 쓸 수 있게 한다는 GPL의 핵심적인 효과는 합리적인 목표입니다. BSD 형식의 라이선스는, 협회 창설을 통한 개인의 연합을 통해, 기술 개발 과정에 대한 경제적 기대를 저버리지 않고 이 목표를 달성할 수 있습니다.

11. BSD 라이선스를 사용하면 좋은 경우

- BSD 라이선스는 연구 결과를 널리 배포하고 경제적 이윤을 창출할 수 있도록 하는 데 적합합니다. 그것으로서, NSF, ONR 그리고 DARPA와 같은 연구 개발 기관은 자금을 투자받은 프로젝트의 초기 단계 연구를 위해 소프트웨어, 데이터, 결과, 그리고 오픈 하드웨어에 BSD 형식의 라이선스를 적용할 것을 장려해야 합니다. 그들은 또한 오픈 소스 시스템 구현과 진행중인 오픈 소스 프로젝트를 기반으로 표준을 정할 것을 장려해야 합니다.
- 정부 정책은 비용과 연구 결과를 실무에 적용할 때의 어려움을 최소화해야 합니다. 가능하다면, 연구 결과를 상용화에 친화적인 BSD 형식의 라이선스로 사용 가능하게 할 것을 요구해야 합니다.
- 많은 경우에, 저작권 또는 특허에 의해 상용 대학 라이선스에 종속되는 것보다 BSD 형식 라이선스의 형식을 취하는 것이 장기적으로 보았을 때 대학의 연구 목표에 더 근접합니다. 대학 입장에서는 장기적으로 보았을 때 연구 결과를 공개하고, 경제적으로 성공한 졸업생들에 의해 기부받는 것이 금전적으로 더 이득이 되는 실제적인 사례들이 존재합니다.
- 회사들은 사실상의 표준을 만드는 것이 마케팅 기술의 핵심이 된다는 것을 오랜 경험에 걸쳐 알아왔습니다. 만약 회사가 시스템을 발전시키는 데 독특한 강점이 있다면, BSD 라이선스는 이 역할을 잘 수행합니다. 라이선스는 회사의 전문 기술 덕분에 그들이 통제할 수 있으면서도 많은 사람들에게 법적인 관점에서 매력적입니다. 다른 사람들을 방해하거나 그들의 것을 빼앗고자 할 목적으로 그러한 표준을 만드는 경우 GPL은 적절한 선택이 될 수 있습니다. 그러나 GPL은 상용으로 적용 가능한 표준을 지향하지 않고 방해합니다. 그러한 GPL suite를 사용하는 것은 지속적으로 상용화와 법률 상의 문제를 발생시킵니다. GPL 표준과 그렇지 않은 것을 함께 활용하는 것은 대개 가능하지 않습니다. 진정한 기술 표준은 기술적이지 않은 이유 때문에 다른 표준을 배제할 것을 강제해서는 안 됩니다.
- 다른 회사들의 상용 제품의 핵심이 될 수 있는 표준을 개발하고 장려하는 데 관심이 있는 회사는 GPL에 대해 조심해야 합니다. 사용된 라이선스에 관계없이, 결과가 되는 소프트웨어는 기술적인 변화의 대부분과 시스템의 상태를 가장 잘 아는 사람들에게 맡겨질 것입니다. GPL은 단지 결과에 추가적인 법적 제약을 가할 뿐입니다.
- 오픈 소스 코드를 개발하는 큰 회사는 오픈 소스를 옹호하는 프로그래머를 조심해야 하는데, 이는 그들이 다른 곳으로 이직할 경우에도 여전히 그 소프트웨어를 사용할 수 있기 때문입니다. 일부 회사들은, 특히 해당 소프트웨어가 핵심 전략에 직접적으로 연관되어 있지 않을 때, 이러한 행동을 직원들의 특권으로써 장려합니다. 이는 사실 직접적인 비용 손실 없이 잠재적인 기회 비용만을 잃는 퇴직 혜택입니다. 직원을 회사 외부로 위해 일하도록 장려하는 것은 종종 회사가 손해 없이 제공할 수 있는 간단한 혜택이 되기도 합니다.
- 소프트웨어가 방치될 것이 걱정되는 작은 개발 회사는, 가능하다면 BSD 라이선스를 사용하는 것을 고려해볼 필요가 있습니다. 회사들은 그 규모에 관계없이 BSD 형식 오픈 소스 프로젝트를 구성함으로써 법적인, 그리고 구조적인 간접 비용을 최소화하는 상호간의 이익을 도모할 수 있습니다.
- 비영리단체들은 가능하다면 오픈 소스 프로젝트에 참여해야 합니다. 서로 다른 라이선스의 코드를 함께 사용하는 것과 같은 소프트웨어상의 문제를 최소화하기 위해, BSD 형식의 라이선스가 장려되어야 합니다. 소프트웨어 개발과 밀접하게 관련있는 비영리단체는 GPL을 더욱 경계해야 할 것입니다. 법률을 적용하는 일이 많은 비용을 요구하는 일부 지역에서는, GPL에 비해 단순한 BSD 라이선스가 고려해볼 만한 이점이 될 것입니다.

12. 맺음말

오픈 소스 코드의 상용화를 막기 위해 만들어진 GPL과는 달리, BSD 라이선스는 미래의 행동에 최소한의 제약만을 가합니다. 이는 프로젝트나 회사의 필요에 따라 BSD 코드를 오픈 소스로 유지하거나 상용 솔루션에 사용될 수 있도록 허락합니다. 다시 말해서, BSD 라이선스는 개발 과정에서 법적인 시한 폭탄이 되지 않습니다.

더불어 BSD 라이선스는, GPL이나 LGPL 라이선스와는 달리 복잡한 법적 의무가 주어지지 않기 때문에, 개발자와 회사가 라이선스 위반 여부에 대해 걱정하는 대신 좋은 코드를 작성하고 홍보하는 데 집중할 수 있도록 해 줍니다.

13. 추가 정보

[1] <http://www.gnu.org/licenses/gpl.html>

[2] <http://archives.cnn.com/2000/TECH/computing/03/28/cyberpatrol.mirrors/>

[3] Open Source: the Unauthorized White Papers, Donald K. Rosenberg, IDG Books, 2000. Quotes are from page 114, ``Effects of the GNU GPL''.

[4] In the "What License to Use?" section of
<http://www.oreilly.com/catalog/opensources/book/brian.html>

This whitepaper is a condensation of an original work available at
http://alumni.cse.ucsc.edu/~brucem/open_source_license.htm

