정의

오픈 소스 소프트웨어(open source software, OSS)란 소스 코드를 공개해 누구나 특별한 제한 없이 그 코드를 보고 사용할 수 있는 오픈 소스 라이선스를 만족하는 소프트웨어로 통상 오픈 소스라고 말하기도 한다. 하지만 위와 같은 정의를 잘못 생각하여 착각 하는 경우가 많은데, 오픈 소스는 코드를 자발적, 공개적 협업 (public collaboration)의 정신으로 발전 가능성에 투자하여 코드를 공유, 공개 하는 것이지 그러한 모든 소스를 ‘Free Wares’ 라고 생각하면 안 된다. 즉, 오픈 소스를 공짜 코드라고 인식하는 것은 명백히 틀린 인식이라는 것이다. 오픈 소스 모델이나 다양한 독립 정보원의 협동 개발은 한 기업이 개발하거나 유지할 수 있는 것보다 훨씬 더 다양한 디자인 관점의 범위를 만든다.

===========================================================================

자유 소프트웨어의 정의

<https://ko.wikipedia.org/wiki/%EC%9E%90%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4%EC%9D%98_%EC%A0%95%EC%9D%98>

[리차드 스톨먼](https://ko.wikipedia.org/wiki/%EB%A6%AC%EC%B0%A8%EB%93%9C_%EC%8A%A4%ED%86%A8%EB%A7%8C)이 작성하고 [자유 소프트웨어 재단](https://ko.wikipedia.org/wiki/%EC%9E%90%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4_%EC%9E%AC%EB%8B%A8)(FSF)이 발행한 자유 소프트웨어의 정의는 [자유 소프트웨어](https://ko.wikipedia.org/wiki/%EC%9E%90%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4)를 최종 사용자가 소프트웨어를 사용하고, 연구하고, 공유하고, 수정할 자유를 보장하는 소프트웨어로 정의한다. 용어 "자유(Free)"는 "표현의 자유(Free Sppech)"에 해당하는 의미로 사용되지 "무료(Free of Charge)"의 의미는 아니다. 이 정의에 대해 알려진 최초의 발행은 지금은 존재하지 않는 GNU의 FSF 회보 1986년 2월 판에 있었다. 문서의 공식적인 출처는 [GNU 프로젝트](https://ko.wikipedia.org/wiki/GNU_%ED%94%84%EB%A1%9C%EC%A0%9D%ED%8A%B8) 웹사이트의 철학 섹션에 있다. 2008년 4월 현재 이 정의는 39개 언어로 발행되었다. FSF는 이 정의를 만족하는 라이선스의 목록을 발행한다

FSF가 1986년 2월 발행한 정의는 다음 두 가지 중점을 갖는다.

우리 이름의 단어 "자유(Free)"는 가격을 의미하지 않으며, 이는 자유를 나타낸다.

첫째, 프로그램을 복제하고, 이를 자기 자신은 물론 주변 사람들이 사용할 수 있도록 재배포할 자유.

둘째, 프로그램이 우리를 조정하는 것이 아니라 우리가 프로그램을 제어할 수 있도록 프로그램을 수정할 자유. 그리고 이를 위해 소스 코드는 반드시 우리에게 공개되어야 한다.

1996년, gnu.org 웹사이트가 공개되었을 때, "자유 소프트웨어"는 소프트웨어를 연구할 자유에 대한 명시적 언급(이는 두 가지 중점의 정의에서 프로그램을 수정할 자유의 일부로 이해될 수 있었다)을 추가해 "세 가지 수준의 자유"를 의미하는 것으로 정의되었다. 스톨먼은 이후 우리는 모든 자유가 필요하다 말하며 수준에 대해 생각하는 것이 오해를 불러올 수 있고 "수준"이라는 단어를 사용하지 않았다.

마지막으로 사용자가 프로그램을 수행할 수 있어야 한다고 명시적으로 언급하는 또 다른 자유가 추가되었다. 기존의 자유가 이미 1부터 3까지 숫자가 메겨져 있었지만, 새로 추가된 자유는 가장 앞에 와야 했기 때문에 "자유 0"으로 추가되었다.[[6]](https://ko.wikipedia.org/wiki/%EC%9E%90%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4%EC%9D%98_%EC%A0%95%EC%9D%98#cite_note-6)

현재의 정의는 소프트웨어를 받은 사람이 다음 네 가지 자유를 가졌는지 여부에 따라 자유 소프트웨어를 정의한다. [[7]](https://ko.wikipedia.org/wiki/%EC%9E%90%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4%EC%9D%98_%EC%A0%95%EC%9D%98#cite_note-7)

•프로그램을 어떠한 목적을 위해서도 실행할 수 있는 자유 (자유 0).

•프로그램의 작동 원리를 연구하고 이를 자신의 필요에 맞게 변경시킬 수 있는 자유 (자유 1). 이러한 자유를 위해서는 소스 코드에 대한 접근이 선행되어야 합니다.

•이웃을 돕기 위해서 프로그램을 복제하고 배포할 수 있는 자유 (자유 2).

•프로그램을 향상시키고 이를 공동체 전체의 이익을 위해서 다시 환원시킬 수 있는 자유 (자유 3).

이러한 자유를 위해서는 소스 코드에 대한 접근이 선행되어야 합니다.

자유 1과 3은 소스 코드 없이 소프트웨어를 연구하거나 수정하는 것이 매우 비현실적이기 때문에 [소스 코드](https://ko.wikipedia.org/wiki/%EC%86%8C%EC%8A%A4_%EC%BD%94%EB%93%9C)가 공개될 것을 요구한다.

오픈 소스의 정의와 비교

[자유 소프트웨어 운동](https://ko.wikipedia.org/wiki/%EC%9E%90%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4_%EC%9A%B4%EB%8F%99)과 오픈 소스 운동 간의 근본적인 철학적 차이에도 불구하고 [자유 소프트웨어 재단](https://ko.wikipedia.org/wiki/%EC%9E%90%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4_%EC%9E%AC%EB%8B%A8)의 [자유 소프트웨어](https://ko.wikipedia.org/wiki/%EC%9E%90%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4)에 대한 공식 정의와 [오픈 소스 이니셔티브](https://ko.wikipedia.org/wiki/%EC%98%A4%ED%94%88_%EC%86%8C%EC%8A%A4_%EC%9D%B4%EB%8B%88%EC%85%94%ED%8B%B0%EB%B8%8C)의 [오픈 소스 소프트웨어](https://ko.wikipedia.org/wiki/%EC%98%A4%ED%94%88_%EC%86%8C%EC%8A%A4_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4)에 대한 정의는 몇 사지 사소한 예외는 있지만, 기본적으로 같은 소프트웨어 라이선스들을 언급한다. 철학적 차이를 강조하면서 자유 소프트웨어 재단은 다음과 같은 의견을 개진했다.

어떤 사람들은 자유 소프트웨어와 거의 같은 부류를 나타내고자 용어 “오픈 소스” 소프트웨어를 사용한다. 오픈 소스 소프트웨어는 자유 소프트웨어와 정확히 같은 종류의 소프트웨어는 아니다. 오픈 소스 소프트웨어는 우리가 너무 제한적이라고 여기는 라이선스를 받아들이기도 하며, 그들이 인정하지 않는 자유 소프트웨어 라이선스가 존재하기도 한다. 하지만 부류의 범위에 대한 차이는 작다. 거의 모든 자유 소프트웨어는 오픈 소스이고, 거의 모든 오픈 소스 소프트웨어는 자유이다.

— 자유 소프트웨어 재단—

===========================================================================

[https://ko.wikipedia.org/wiki/%EC%9E%90%EC%9C%A0\_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4\_%EC%9E%AC%EB%8B%A8](https://ko.wikipedia.org/wiki/%25EC%259E%2590%25EC%259C%25A0_%25EC%2586%258C%25ED%2594%2584%25ED%258A%25B8%25EC%259B%25A8%25EC%2596%25B4_%25EC%259E%25AC%25EB%258B%25A8)

(위키피디아 FSF – 자유 소프트웨어 재단)

/\* 바로 위에서 언급된 FSF에 대한 내용\*/

자유소프트웨어재단([영어](https://ko.wikipedia.org/wiki/%EC%98%81%EC%96%B4): Free Software Foundation 줄여서 **FSF**)은 [미국](https://ko.wikipedia.org/wiki/%EB%AF%B8%EA%B5%AD)의 자유 소프트웨어 관련 재단이다. [자유 소프트웨어](https://ko.wikipedia.org/wiki/%EC%9E%90%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4)의 생산과 보급을 장려하기 위해 [리처드 스톨만](https://ko.wikipedia.org/wiki/%EB%A6%AC%EC%B2%98%EB%93%9C_%EC%8A%A4%ED%86%A8%EB%A7%8C)이 세운 재단으로, 주로 [컴퓨터 소프트웨어](https://ko.wikipedia.org/wiki/%EC%BB%B4%ED%93%A8%ED%84%B0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4)를 만들어 배포하고 수정하는 보편적인 자유를 제고한다. 설립 이후부터 1990년대 중반까지 자유 소프트웨어 재단 기금은 [GNU 프로젝트](https://ko.wikipedia.org/wiki/GNU_%ED%94%84%EB%A1%9C%EC%A0%9D%ED%8A%B8)의 [자유 소프트웨어](https://ko.wikipedia.org/wiki/%EC%9E%90%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4)를 작성하기 위해 소프트웨어 개발자를 고용하는데 대부분 사용되었다. 1990년대 중반 이후로 이 재단의 직원들과 자발적인 기여자들은 대개 자유 소프트웨어 운동과 [자유 소프트웨어 커뮤니티](https://ko.wikipedia.org/wiki/%EC%9E%90%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4_%EC%9A%B4%EB%8F%99)를 위한 법적, 구조적 문제에 대한 작업을 처리하고 있다. [2002년](https://ko.wikipedia.org/wiki/2002%EB%85%84) [11월 25일](https://ko.wikipedia.org/wiki/11%EC%9B%94_25%EC%9D%BC), 자유 소프트웨어 재단은 자유 소프트웨어 재단 연합 멤버십 프로그램을 시작했다. 목표를 지속하기 위해 오직 [자유 소프트웨어](https://ko.wikipedia.org/wiki/%EC%9E%90%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4)만이 FSF의 컴퓨터에 사용된다.

=============================================================================

OSI

<http://www.opensource.org/>

Open Source Initiative(OSI)-opensource.org 는 공개 소스 정의(OSD)의 관리 및 촉진을 담당하는 비영리 조합이다. OSI가 인증하는 공개 소스 소프트웨어(OSS) 인증 마크를 통해 소프트웨어가 실제로 공개 소스라는 것을 증명하고, 공개 소스의 복제도 가능하다. 소프트웨어의 소스 코드를 자유롭게 읽고, 재배포 및 개조를 가능하게 함으로써 소프트웨어가 향상되고, 한 사람이 느린 속도로 소프트웨어를 개발하는 것보다 여러 사람들이 고치고 쓰고 버그를 개선하는 것이 보다 빠를 수 있다는 공개 소스의 기본 이념을 구현하는 단체이다.

=============================================================================

[http://itnp.kr/roller/home/entry/%EC%98%A4%ED%94%88%EC%86%8C%EC%8A%A4\_open\_source\_%EA%B0%9C%EC%9A%94](http://itnp.kr/roller/home/entry/%25EC%2598%25A4%25ED%2594%2588%25EC%2586%258C%25EC%258A%25A4_open_source_%25EA%25B0%259C%25EC%259A%2594) (오픈소스(Open-Source)의 개요)

-오픈소스의 조건 10가지 (OSD : Open Source Definition)-

<http://korea.gnu.org/documents/copyleft/osd-korean.html> (오픈 소스의 정의)

오픈 소스의 정의

/\* 여기가 중요한 파트 오픈소스 정의하는 10가지\*/

1.9판

각각의 문단마다 이유라는 단어와 함께 이탤릭체로 들여쓰기 된 부분은 오픈 소스의 정의(OSD)에 대한 주석이며 OSD의 일부가 아닙니다**.** 주석이 포함되지 않은 OSD 원판은 [이곳에서 참고](http://korea.gnu.org/documents/copyleft/osd-plain-korean.html)할 수 있습니다.

소개

오픈 소스란 단지 원시 코드를 이용할 수 있다는 것만을 의미하는 것이 아닙니다. 오픈 소스 소프트웨어의 배포 조건은 다음과 같은 기준들을 만족시켜야만 합니다.

1. 자유로운 재배포

오픈 소스 사용 허가(license)는 몇 개의 다른 출처로부터 모아진 프로그램들로 구성된 집합 저작물 형태의 배포판의 일부로 소프트웨어를 판매하거나 무상 배포하는 것을 제한해서는 안됩니다. 또한 그러한 판매에 대해 사용료나 그밖의 다른 비용을 요구해서도 안됩니다.

이유**:** 사용 허가에 자유로운 재배포를 규정하도록 강제함으로써 우리는 단기간의 적은 판매 수익을 얻기 위해 많은 장기적인 이익을 포기하는 유혹을 없앨 수 있습니다. 만약 이렇게 하지 않는다면, 협력자들에게 많은 변심의 압력이 있을 것입니다.

2. 원시 코드

오픈 소스 프로그램에는 원시 코드(source code)가 포함되어야 하며, 컴파일된 형태 뿐 아니라 원시 코드의 배포도 허용되어야 합니다. 만약 원시 코드가 함께 제공되지 않는 제품이 있다면 원시 코드를 복제하는데 필요한 합당한 비용만으로 원시 코드를 구할 수 있는 널리 알려진 방법이 제공되어야만 합니다. 이러한 경우에 있어 가장 권장할 만한 방법은 별도의 비용없이 인터넷을 통해 원시 코드를 다운받을 수 있도록 하는 것입니다. 원시 코드는 프로그래머가 이를 개작하기에 용이한 형태여야 하며, 고의로 복잡하고 혼란스럽게 만들어진 형태와 선행 처리기나 번역기에 의해 생성된 중간 형태의 코드는 인정되지 않습니다.

이유**:** 원시 코드를 불분명하지 않은 형태로 제공하도록 규정하는 이유은 프로그램을 발전시키기 위해서 원시 코드에 대한 개작이 선행되어야 하기 때문입니다. 우리의 목적은 발전을 용이하게 만들기 위한 것이므로 개작이 용이하게 이루어 질 수 있는 방법을 요구합니다.

3. 파생 저작물

오픈 소스 사용 허가에는 프로그램의 개작과 2차적 프로그램의 창작이 허용되어야 하며, 이러한 파생 저작물들이 원프로그램에 적용된 것과 동일한 사용 허가의 규정에 따라 배포되는 것을 허용해야만 합니다.

이유**:** 단순히 원시 코드를 열람할 수 있는 것만으로는 독립된 등위 검토(peer review)와 빠른 발전 경쟁에서의 생존을 지원할 수 없습니다. 프로그램을 빠르게 발전시키기 위해서는 사람들에게 개작된 프로그램을 실험하고 재배포할 수 있도록 허용할 필요가 있습니다.

4. 저작자의 원시 코드 원형 유지

오픈 소스 사용 허가는 바이너리를 생성할 시점에서 프로그램을 수정할 목적으로, 원시 코드를 수반한 ``패치 파일''의 배포를 허용한 경우에 한해서 패치로 인해 변경된 원시 코드의 배포를 제한할 수 있습니다. 그러나 이 경우에도 변경된 원시 코드를 통해 만들어진 소프트웨어의 배포는 명시적으로 허용해야만 합니다. 오픈 소스 사용 허가는 파생 저작물에 최초의 소프트웨어와 다른 판 번호(version)와 이름이 사용되도록 규정할 수 있습니다.

이유**:** 소프트웨어에 많은 향상이 이루어지도록 장려하는 것은 좋은 일입니다. 그러나 사용자에게는 그들이 사용하고 있는 소프트웨어를 누가 책임지고 있는 지를 알 권리가 있습니다. 또한 저작자와 관리자에게도 반대 입장에서 사용자들이 그들에게 어떤 지원을 요구하고 있는 지를 알 권리와 그들의 명성을 보호할 권리가 있습니다.

따라서 오픈 소스 사용 허가는 원시 코드가 쉽게 이용될 수 있도록 보증해야만 하지만 변형되지 않은 최초의 원시 코드가 패치 파일와 함께 배포되도록 규정할수도 있습니다. 이러한 방법을 통해 ``비공식'' 수정들을 이용할 수 있으면서 원시 코드의 원형이 쉽게 구별될 수 있습니다.

5. 개인 및 단체에 대한 차별 금지

오픈 소스 사용 허가는 특정 개인이나 단체를 차별해서는 안됩니다.

이유**:** 오픈 소스의 공정으로부터 최대의 이익을 끌어내기 위해 최대한 다양한 개인과 단체에게 오픈 소스에 기여할 수 있는 동등한 자격이 부여되어야 합니다. 따라서 우리는 어떠한 오프 소스 사용 허가도 특정인을 오픈 소스의 공정으로부터 제외시키는 것을 금지합니다.

아메리카 합중국을 포함한 몇몇 국가에서는 특정한 종류의 소프트웨어에 대한 수출이 금지되고 있습니다. OSD를 준수하는 사용 허가는 피양도자에게 이러한 종류의 제한에 대해 경고하고 해당 법률을 준수해야 한다는 사실을 상기시킬 수 있습니다. 그러나 사용 허가 자체에 그러한 제한이 통합되어서는 안됩니다.

6. 사용 분야에 대한 차별 금지

오픈 소스 사용 허가는 프로그램이 특정 분야에서 사용되는 것을 금지하는 제한을 설정해서는 안됩니다. 예를 들면, 기업이나 유전학 연구에 프로그램을 사용할 수 없다는 등과 같은 제한을 설정해서는 안됩니다.

이유**:** 이 조항의 주된 목적은 오픈 소스가 상업적으로 이용되지 못하게 방해하는 규정이 사용 허가에 포함되는 것을 금지하기 위한 것입니다. 우리는 상업 이용자들도 오픈 소스 공동체에 동참하기를 원하며 이들이 공동체로부터 소외감을 느끼지 않기를 바랍니다.

7. 사용 허가의 배포

프로그램에 대한 권리는 배포에 따른 각 단계에서 배포자에 의한 별도의 사용 허가 없이도 프로그램을 재배포받은 모든 사람에게 동일하게 인정되어야만 합니다.

이유**:** 이 조항은 비공개 계약을 요구하는 것과 같은 간접적인 수단을 통해 소프트웨어가 제한되는 것을 금지하기 위한 것입니다.

8. 특정 제품에만 유효한 사용 허가의 금지

프로그램에 대한 권리는 프로그램이 특정한 소프트웨어 배포판의 일부가 될 때에 한해서만 유효해서는 안됩니다. 만약 특정 배포판에 포함되어 있던 프로그램을 별도로 분리한 경우라 하더라도 프로그램에 적용된 사용 허가에 따라 프로그램이 사용되거나 배포된다면 프로그램을 재배포받은 모든 사람에게 최초의 소프트웨어 배포판을 통해 프로그램을 배포받은 사람과 동일한 권리가 보장되어야만 합니다.

이유**:** 이 조항은 또다른 형태의 사용 허가상의 제한을 방지하기 위한 것입니다.

9. 다른 소프트웨어를 제한하는 사용 허가의 금지

오픈 소스 사용 허가는 오픈 소스 사용 허가가 적용된 소프트웨어와 함께 배포되는 다른 소프트웨어에 대한 제한을 포함해서는 안됩니다. 예를 들면, 사용 허가 안에 동일한 매체를 통해 배포되는 다른 소프트웨어들이 모두 오픈 소스 소프트웨어여야 한다는 제한을 두어서는 안됩니다.

이유**:** 오픈 소스 소프트웨어의 배포자들은 그들의 소프트웨어에 대한 스스로의 선택 권리를 갖고 있습니다.

물론, GPL은 이러한 규정을 충족시키고 있습니다. GPL 라이브러리와 결합되는 소프트웨어는 하나의 단일 저작물을 형성할 때에 한해서 GPL이 계승되는 것이지 단순히 함께 배포된다는 것만으로 GPL 소프트웨어가 되어야 하는 것은 아닙니다.

-오픈소스 라이센스-

<http://www.opensource.org/licenses/>

오픈소스를 시작하는 것은 정치적 이슈이자 라이선스 이슈이지 기술적 이슈는 아니다.

다음의 3가지를 정의 하는 것

Patent 특허권 , Copyright 저작권, Trade Secret 영업비밀

공통 사항 (3가지)

저작권 관련 문구 유지

오픈소스의 문구를 제거하면 안된다.

제품명 중복 방지 (상표권)

서로 다른 라이선스의 조합 확인

<http://www.gnu.org/philosophy/license-list.html>

GPLv2와 아파치 라이선스는 서로 상충되는 라이선스 조건 때문에

하나의 소스코드에 결합하여 사용하는 것이 불가능하다

주요 오픈소스 라이선스

GPL(General Public License)

'GPL에 의해 배포된다’는 사실을 명시해

소스 코드를 수정하는 경우 반드시 '개작 부분은 공개'

소스 코드를 링크하는 경우에도 '모두 소스 코드를 공개'해야 한다

전세계 라이선스의 60% 이상을 차지(2007년 기준)

LGPL(Lesser General Public License)

GPL과 동일

소스 코드를 링크하는 경우 소스를 공개할 필요가 없다.

MPL(Mozilla Public License)

소스 코드의 수정 시 소스 코드 공개는 필수적이지만,

결합해 프로그램을 만들 경우 MPL 코드를 제외한 결합 프로그램에 대한 소스 코드는 공개할 의무가 없다

BSD(Berkeley Software Distribution)

수정 부분에 대해 소스 코드 공개는 의무 없음

소스 코드를 상용 프로그램과 조합하는 것도 허용

AL(Apache License)

수정 부분에 대해 소스 코드 공개는 의무 없음

소스 코드를 상용 프로그램과 조합하는 것도 허용

Apache’라는 표장에 대한 상표권을 침해하지 말아야 한다.

참고사이트

[한국저작권 위원회](http://www.olis.or.kr/index.do)

오픈소스 라이센스의 세계도 자유소프트웨어 vs 오픈소스 진영으로 의견이 나뉜다

"GPL을 지지하는 측에서는 BSD라이센스를 선택하는 것은 기업들이 소스를 변형하여 2차 저작물을 작성 하여 이에 대해 독점적인 라이센스 정책을 취할 것이며, 일반인들의 자유사용이 제한되므로 진정한 자유소프트웨어는 될 수 없을 것이라고 비판한다."

오픈소스는 나와 상관없는 이야기가 아니다.

많은 오픈소스가 자신도 모르는 사이에 시용되고 있어 앞으로 더 많이 사용될 것이다.

===========================================================================

[https://ko.wikipedia.org/wiki/%EC%82%AC%EC%9C%A0\_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4#cite\_note-linfo-2](https://ko.wikipedia.org/wiki/%25EC%2582%25AC%25EC%259C%25A0_%25EC%2586%258C%25ED%2594%2584%25ED%258A%25B8%25EC%259B%25A8%25EC%2596%25B4%23cite_note-linfo-2) (위키피디아 – 사유 소프트웨어/독점 소프트웨어)

/\* 맨 뒤에 클로즈드 소스 비교부분과 정의 부분 중에서 글에 흐름에 편한부분에 넣으시면 되요\*/

사유소프트웨어(私有-, [영어](https://ko.wikipedia.org/wiki/%EC%98%81%EC%96%B4): proprietary software) 또는 클로즈드소스소프트웨어([영어](https://ko.wikipedia.org/wiki/%EC%98%81%EC%96%B4): closed source software)는 저작권 소유자의 예외적 법적 권한 하에 허가된 컴퓨터 소프트웨어이다.[[1]](https://ko.wikipedia.org/wiki/%EC%82%AC%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4#cite_note-1) 독점소프트웨어라고도 한다. 또, [오픈 소스](https://ko.wikipedia.org/wiki/%EC%98%A4%ED%94%88_%EC%86%8C%EC%8A%A4) 소프트웨어의 반대말로 오픈 소스 소프트웨어의 정의에 부합하지 않는 [저작권](https://ko.wikipedia.org/w/index.php?title=%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4_%EC%A0%80%EC%9E%91%EA%B6%8C&action=edit&redlink=1)을 갖는 프로그램을 일컫는다. 특정한 조건으로 소프트웨어를 사용할 권한이 주어지는데, 수정, 다른 곳으로의 배포, [역공학](https://ko.wikipedia.org/wiki/%EC%97%AD%EA%B3%B5%ED%95%99)과 같은 기타 이용은 제한된다.[[2]](https://ko.wikipedia.org/wiki/%EC%82%AC%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4#cite_note-linfo-2) 그러므로 일반적으로는 소프트웨어를 기술적으로 변형하거나 변조할 수 없도록 저작권을 통해 [소스 코드](https://ko.wikipedia.org/wiki/%EC%86%8C%EC%8A%A4_%EC%BD%94%EB%93%9C)로 접근하는 것을 막고 [이진 파일](https://ko.wikipedia.org/wiki/%EC%9D%B4%EC%A7%84_%ED%8C%8C%EC%9D%BC) 형태로만 제공되는 프로그램을 가리킨다. 보통 그러한 프로그램의 소스 코드는 제작사의 [기업 비밀](https://ko.wikipedia.org/wiki/%EA%B8%B0%EC%97%85_%EB%B9%84%EB%B0%80)로 간주된다. 제 삼자가 소스 코드를 사용해야 할 때에는 비공개 협약이 필요한 경우가 많다.

“클로즈드 소스”라는 용어는 프로그램의 소스 코드가 공개되지 않는 “비공개 소스”와 혼동될 수 있다. 마이크로소프트의 [공유 소스](https://ko.wikipedia.org/wiki/%EA%B3%B5%EC%9C%A0_%EC%86%8C%EC%8A%A4)는 소스 코드가 공개되긴 하지만 “오픈 소스”는 아닌 예이다. 다시 말해, “클로즈드 소스”라는 용어를 “오픈 소스”의 반대 개념으로 사용한다면 “공유 소스”는 “클로즈드 소스”의 한 형태에 해당된다.

오픈소스의 경제효과

<http://leesgab.blog.me/156516878>

/\*경제쪽에서 오픈소스 관련 내용인대 파트를 나누기 어려워서 따로 분류해둠\*/

오픈 소스 소프트웨어 운동을 사회적 현상으로 이해해야 하는가, 아니면 단순한 경제적 현상으로 이해해야 하는가? 경제학에서는 전통적으로 무임승차(free riding)가 생산 의욕을 저해하고 경제적 효율성을 떨어뜨린다는 주장이 일반적이었다. 하지만 오픈소스 소프트웨어는 과거 효율성 개념을 넘어선 새로운 공공재로 인식되고 있다.

제로섬 게임

오픈소스 소프트웨어 운동을 사회적현상으로 이해해야 하는가, 아니면 단순한 경제적현상으로 이해해야 하는가? 사실은 두가지가 다 내포돼 있다.  
먼저 경제적 측면에서 보면, 오픈 소스 소프트웨어의 직접적 효과는 이용자 측에서 발생한다. 그것은 TCO(Total Cost of Ownership)로측정되는 총소유비용 절감효과로 나타난다. 만일오픈소스 소프트웨어의 TCO가 정말로 낮다면 이용자의 비용은 당연히 절감되겠지만, 공급자입장에서는 매출이 줄어드는 것을 의미한다. 따라서 직접적인 경제적효과로만 보면 오픈소스는 사실상 제로섬게임에 가깝다.

오픈 소스 소프트웨어임과 동시에 무료 소프트웨어를 표방한 리눅스의 수익모델은 특이하다. 리눅스는 소프트웨어 자체에 대해 값을 받지 않는다. 수익의 원천은 대부분 장비유통, 운영체제 컨설팅 및 유지보수 서비스에 대한 수입이다. 반면에 상용 소프트웨어는 소프트웨어 자체에 대해 값을 받는다. 그리고 그 값에는사실상 A/S와 유지보수비용이 포함돼 있는 경우가 많다.

리눅스와 같은 오픈 소스 소프트웨어의 TCO가 과연 상용소프트웨어보다 낮은지에 대해서는 아직도 논란이 많다. TCO의 계산은 그리 간단한 문제가 아니다. 마이크로소프트와 같은 상용 소프트웨어 기업과 리눅스컨설팅 기업들은 각자 자신에게 유리한 기준으로 측정한TCO를 제시해 고객을 유인하려고 한다.  
마이크로소프트는 리눅스의 TCO가 높다고 하고, 리눅스는 윈도의 TCO가 높다고 주장한다. 구매기업 입장에서는 어느 쪽이 맞는지 매우 혼란스러울 수 있다. 최근에는 사실상 큰 차이가 없다는 주장도 힘을 얻고있다.

혜택은 소비자에게

 오픈 소스는 공유의 철학을 표방하고 있지만, 종종 상업적 활용도 허용한다. 특히 GNU의 GPL(General Public License)은 일정한 단서하에 상업적활용을 허용한다. GPL을 따르되 상업적이익을 추구하는 업체들은 다른 라이선스, 예컨대 BSD나 아파치 라이선스에 의존하면서 공유의 철학을 일부 포기하기도 한다.

그렇다면, 구글의 안드로이드 운영체제는 어디까지가 오픈 소스일까? 구글이 무료로 가져다 쓴 리눅스커널은 GPL을 따르므로 철저하게 오픈 소스로 남는다. 그러나 리눅스 커널을 바탕으로 구글이 직접 개발한 프레임워크와 동적연계라이브러리(DLL) 등은 GPL을 따르지 않는다. 따라서 공개할 의무가 없다.  
특히 삼성전자처럼 모바일 기기를 직접 생산하는 회사는 운영체제와 장치 사이의 통신 모듈은 철저하게 비공개 원칙을 고수하고 있다. 그런 의미에서 리눅스는 구글과 삼성전자를 통해 재탄생하는 과정에서 반쪽짜리 오픈소스가 됐다.

그렇지만, 결과적으로 구글과 삼성전자는 아주 효율적인 무임승차(free riding)를 한 것이다. 그리고 오늘날 안드로이드폰을 이용하는 상당수의 사람들이 그 혜택을 고스란히 누리고 있기도 하다. 결국 오픈소스가 사회적 후생을 더욱 증가시키는 데에 기여한 것이다. 반면 소프트웨어 공개에 대해 전혀 다른 입장에 서 있는 마이크로소프트는 이런 무임승차의 혜택을 보지 못한 채 모바일 시장에서 여전히 고전을 면치 못하고 있다.

경제학에서는 전통적으로 무임승차(free riding)가 생산 의욕을 저해하고 경제적 효율성을 떨어뜨린다는 주장이 일반적이었다. 하지만 오픈 소스 소프트웨어는 특이한 공공재다. 그것은 사용하는 사람이 많을수록 효용이 감소하는 클럽재(club goods)가 아니다. 누구나 무료로 이용할 수 있는 클럽시설은 이용하는 사람이 많을수록 개인의 입장에서는 효용이 줄어든다. 그러나 오픈소스는 사용하는 사람과 보완에 참여하는 사람이 많을수록 가치가 늘어난다. 즉, 네트워크 재화다. 오픈 소스에서는 무임승차의 부작용보다 네트워크의 긍정적인 가치창출효과가 더욱 크다.

진화의 가속

오픈소스 운동의 진정한 효과는 이익창출이 아니라 진화(evolution)의 가속화에 있다. 오픈메커니즘은 돌연변이(mutation)와 교배(cross-over)의 가능성을 한껏 확대한다. 종의 다양성이 지닌 아름다움이 바로 다윈(Darwin) 진화론의 본질이다. 소프트웨어의 진화도 그와 마찬가지다. 오픈 소스는 소프트웨어종의 다양성을 촉진함으로써 이용자에게 더욱 풍요로운 기회를 제공한다. TeX시스템의 개발과 공개로 1980년대에 진행된 오픈소스 운동을 주도한 또 다른인물인 커누스(Donald E. Knuth) 교수는 다음과 같이 말한적이 있다.(2008.4.25)

오픈 소스 소프트웨어는 경제가 생산물 중심에서 서비스 중심으로 바뀌어가면서 더욱 대세가 될것이며, 소스를 개선하는 데에 참여하는 자원 봉사자들은 날로 늘어날 것입니다. (중략) 개인 사용자들이 자신의 사용환경과 취향에 맞추어 사용할 수 있는 소프트웨어는 점점 늘어날 것입니다. 반면에 상용 소프트웨어는 이런 수요에 절대로 부응할 수 없고 단지 몇 개의 획일화된 버전을 내놓은 데에 그칠 것입니다.

소수의 획일화된 버전으로 대변되는 상용 소프트웨어는 제조업 중심이자 공급자 중심이었던 20세기의 패러다임이다. 그것은 20세기 초에 부상한 포드 자동차의 철학이다. 사용자는 공급자가 내놓는 모델을 군말 없이 써야 한다. 그리고 공급자 내부 인력으로만 추진하는 혁신은 동종교배(同種交配)와 같다. 마이크로소프트의 혁신이 질곡에 빠진 이유가 바로 여기에 있다. 오픈 소스 진영 앞에서 고전하던 마이크로소프트가 최근에 마지못해 비주얼베이직(Visual Basic)의 소스를 공개하겠다고 선언하기까지 했다.

2007년 노벨 경제학상 수상자인 매스킨(Erik Maskin) 교수는 베센(James Bessen)과 공저한 논문1)에서 소프트웨어의 특허는 혁신과 경제성장을 오히려 저해한다고밝힌바 있다. 특허가 경제성장을 촉진한다고 주장했던 통설과 반대 주장이다. 그는 소프트웨어의 연속적이고 보완적인 특성을 감안할 때 경쟁이야말로 가치를 창출할 수 있는 첩경임을 주장했다. 오픈 소스는 여기에서 한 걸음 더 나아간다. 공개와 수정을 허용한다는 특성 때문에, 개발자 사이에 경쟁과 협조가 동시에 이루어지는 코피티션 메커니즘이 존재한다. 누구나 참여할 수 있음과 동시에 누구나 개선할 수 있다. 진화는 이렇게 가속화한다. 오픈소스의 진정한 경제적 효과는 당장의 경제적 이익의 창출이 아니라 진화의 가속화를 통한 사회적 가치의 창출에 있다.

오픈소스는 제로섬게임

소비자입장 비용절감 --> 공급자입장 매출감소 ==> 전체 경제적 효과로 보면 제로썸

But, 오픈소스의 비용절감의 효과는 소비자에게로.

경제적으로

free riding (무임승차) --> 생산의욕을 저해, 생산적 효율성을 떨어 뜨린다.

하지만 현상을 그렇지 않다. 오픈소스는 효율정 경제성을 넘어선 새로운 공공재라 할 수 있다.

즉, 무임승차의 부작용보다 네트워크의 긍정적인 가치창출효과가 더욱 크다

시대 패러다임의 변화

커누스(Donald E. Knuth) 교수는 (2008.4.25)

오픈 소스 소프트웨어는 경제가 생산물 중심에서 서비스 중심으로 바뀌어가면서 더욱 대세가 될것이며, 소스를 개선하는 데에 참여하는 자원 봉사자들은 날로 늘어날 것입니다. 개인 사용자들이 자신의 사용환경과 취향에 맞추어 사용할 수 있는 소프트웨어는 점점 늘어날 것입니다. 반면에 상용 소프트웨어는 이런 수요에 절대로 부응할 수 없고 단지 몇 개의 획일화된 버전을 내놓은 데에 그칠 것입니다.

정보의 경제 vs 생산물 중심의 경제

정보의 경제(The Economics of Information)는 일반적인 공산품 과 관련된 경제와 큰 차이가 있다. 정보의 경제에서는 매우 적은 경비로도 컴퓨터 프로그램 같은 정보를 복제할 수 있다. 빵은 1 파운드 정도의 밀가루 없이는 복제할 수 없다.

역사

자유 소프트웨어 운동은 1980년대 리처드 스톨만이 소프트웨어의 정보 공유 방식을 복원하고자 하여 실행한 운동이다. 이 운동의 핵심작업은 “운영체제” 를 만들어 여러 사람들의 손을 거쳐 점진적으로 완성도 높은 소프트웨어를 만드는 것이며 이 운동의 핵심적인 목표는 그러한 “운영체제” 에서만 그치는 것이 아닌 모든 SW를 자유 SW로 만드는 것이 목표였다.

1984년 리차드 스톨만이 GNU(그누) 프로젝트를 시작하면서부터누구나 자유롭게 "실행, 복사, 수정, 배포"할 수 있고,누구도 그런 권리를 제한하면 안 된다는 사용 허가권(License) 아래 소프트웨어를 배포한다.

이후 리처드 스톨먼은 1985년 FSF 재단을 설립, ‘소프트웨어는 공유돼야 하며 프로그래머는 소프트웨어로 돈을 벌어서는 안 된다’는 내용의 GNU 선언문을 제정하기도 하며, 이를 지원하기 위해 저작권(copyright)에 대응하는 카피레프트 운동도 주창했다.

1990년대 들어서면서 인터넷의 보급과 더불어 GNU GPL(General Public License)로 배포된 Linux의 보급으로 자유소프트웨어 운동이 확산됐고, 1998년에는 MS의 웹 브라우저인 IE에 밀려 어려움을 겪고 있던 넷 스케이프(Netscape)사가 웹 브라우저 모질라의 소스코드를 공개하는 결정을 하게 됐다.

이때 “자유 소프트웨어”가 “오픈 소스 소프트웨어”로 용어가 변경된다. 이는 자유(free)란 용어 때문에 일반인들이 무료라고 인식하고 있다는 점, GPL 조항의 엄격성 때문에 소프트웨어 개발이 용이하지 않다는 점을 탈피하기 위해서 용어가 변경되었다.

-참고-

<https://ko.wikipedia.org/wiki/%EC%9E%90%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4%EC%9D%98_%EC%A0%95%EC%9D%98#.EC.98.A4.ED.94.88_.EC.86.8C.EC.8A.A4.EC.9D.98_.EC.A0.95.EC.9D.98.EC.99.80_.EB.B9.84.EA.B5.90>

1997년 7월 브루스 페렌스(Bruce Perens)는 데비안 자유 소프트웨어 가이드라인을 발행했다. 이는 또한 "오픈소스의정의"라는 이름으로 [오픈 소스 이니셔티브](https://ko.wikipedia.org/wiki/%EC%98%A4%ED%94%88_%EC%86%8C%EC%8A%A4_%EC%9D%B4%EB%8B%88%EC%85%94%ED%8B%B0%EB%B8%8C)(OSI)가 사용했다. 이 때 변경된 유일한 부분은 "자유 소프트웨어"라는 용어 대신 이에 대한 OSI의 다른 용어 "오픈 소스 소프트웨어"가 사용되었다는 것 뿐이다.

이후 1998년 오픈소스 소프트웨어를 인증하는 OSI(Open Source Initiative, www.opensource.net)가 에릭 레이몬드(Eric Raymond) 등에 의해 결성되면서 오픈소스 소프트웨어 운동은 궤도에 오르게 된다.

OSI 단체가 정한 오픈소스 소프트웨어의 기준을 OSD(Open Source Definition)이라 하는데, 이 기준을 만족해야만 오픈소스 소프트웨어로 인정받게 된다. 그 요건은 다음과 같다.

/\*위에 오픈소스 정의 10가지 중복\*/

▲자유로운 재배포(Free Redistribution)

▲소스코드 공개(Source Code)

▲2차적 저작물 허용(Derived Works)

▲저작자의 소스코드의 온전함(Integrity of The Author's Source Code)

▲차별금지(No Discrimination Against Persons or Groups 및 No Discrimination Against Fields of Endeavor)

▲라이선스의 배포(Distribution of License)

===========================================================================

리차드매튜스톨만([영어](https://ko.wikipedia.org/wiki/%EC%98%81%EC%96%B4): Richard Matthew Stallman, [1953년](https://ko.wikipedia.org/wiki/1953%EB%85%84) [3월 16일](https://ko.wikipedia.org/wiki/3%EC%9B%94_16%EC%9D%BC) ~)은 [자유 소프트웨어 운동](https://ko.wikipedia.org/wiki/%EC%9E%90%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4_%EC%9A%B4%EB%8F%99)의 중심 인물이며, [GNU](https://ko.wikipedia.org/wiki/GNU) 프로젝트와 [자유 소프트웨어 재단](https://ko.wikipedia.org/wiki/%EC%9E%90%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4_%EC%9E%AC%EB%8B%A8)의 설립자이다. 그는 이 운동을 지원하기 위해 [카피레프트](https://ko.wikipedia.org/wiki/%EC%B9%B4%ED%94%BC%EB%A0%88%ED%94%84%ED%8A%B8)의 개념을 만들었으며, 현재 널리 쓰이고 있는 [일반 공중 사용 허가서](https://ko.wikipedia.org/wiki/%EC%9D%BC%EB%B0%98_%EA%B3%B5%EC%A4%91_%EC%82%AC%EC%9A%A9_%ED%97%88%EA%B0%80%EC%84%9C)(GPL) 소프트웨어 라이선스의 개념을 도입했다.

그는 또한 탁월한 프로그래머이기도 하다. 그는 [문서 편집기](https://ko.wikipedia.org/wiki/%EB%AC%B8%EC%84%9C_%ED%8E%B8%EC%A7%91%EA%B8%B0)인 [Emacs](https://ko.wikipedia.org/wiki/Emacs), [GNU 컴파일러 모음](https://ko.wikipedia.org/wiki/GNU_%EC%BB%B4%ED%8C%8C%EC%9D%BC%EB%9F%AC_%EB%AA%A8%EC%9D%8C) 컴파일러, [GDB](https://ko.wikipedia.org/wiki/GDB) 디버거 등 많은 프로그램을 만들었으며, 이들 모두를 GNU 프로젝트의 일부로 만들었다.

그는 자유 소프트웨어 운동의 도덕적, 정치적, 법적인 기초를 세우는 데 본질적인 영향을 준 인물이며, 이는 [독점 소프트웨어](https://ko.wikipedia.org/wiki/%EB%8F%85%EC%A0%90_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4) 개발과 공급에 대한 대안이 되었다.

/역사 파트는 리차드 스톨먼이 중요 인물인 관계로 간략한 인물소개 “스톨만” 과 “스톨먼” 중에서 “먼”으로 하는게 적당한듯\*/

자유 소프트웨어 대 오픈 소스

리처드 스톨만의 정치적이고 도덕적인 동기는 그를 매우 논쟁적인 인물로 만들었다. 코드를 공유하자는 개념에 동의하는 프로그래머들 중 많은 수가 스톨만의 도덕주의적인 입장과 개인적인 철학에는 동의하지 않았다. 이러한 논쟁의 한 결과로 자유 소프트웨어 운동의 대안인 [오픈 소스](https://ko.wikipedia.org/wiki/%EC%98%A4%ED%94%88_%EC%86%8C%EC%8A%A4) 운동이 생겨났다.

===========================================================================

<https://ko.wikipedia.org/wiki/%EC%9E%90%EC%9C%A0_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4_%EC%9A%B4%EB%8F%99>

자유소프트웨어운동([영어](https://ko.wikipedia.org/wiki/%EC%98%81%EC%96%B4): Free Software Movement)은 [리처드 스톨먼](https://ko.wikipedia.org/wiki/%EB%A6%AC%EC%B2%98%EB%93%9C_%EC%8A%A4%ED%86%A8%EB%A8%BC)이 [1980년대](https://ko.wikipedia.org/wiki/1980%EB%85%84%EB%8C%80)에 소프트웨어의 본래 생산 유통 방식인 정보 공유의 방식을 복원하고자 한 운동이다. 이 프로젝트의 핵심 작업은 [운영 체제](https://ko.wikipedia.org/wiki/%EC%9A%B4%EC%98%81_%EC%B2%B4%EC%A0%9C)를 만들어 여러 사람들의 손을 거쳐 더 완성도 높은 소프트웨어를 만드는 것이다. 그리고 이 운동의 목표는 운영 체제만이 아닌 모든 소프트웨어를 자유소프트웨어로 만드는 것이다.

자유 소프트웨어 운동은 다음 몇가지의 원칙을 기반으로 하는데,

1. 소프트웨어의 작동 원리를 연구하고 이를 자신의 필요에 맞게 변경시킬 수 있는 자유이며,

2. 소프트웨어를 이웃과 함께 공유하기 위해서 이를 복제하고 배포할 수 있는 자유,

3. 소프트웨어를 향상시키고 이를 공동체 전체의 이익을 위해서 다신 환원시킬 수 있는 자유이다.

즉, 소스 코드 공개를 통해 누구나 소프트웨어를 수정할 수 있게 하며, 자유로운 복제와 배포를 허용하는 것이다.

**GNU** 선언문(GNU Manifesto)은 [리처드 스톨만](https://ko.wikipedia.org/wiki/%EB%A6%AC%EC%B2%98%EB%93%9C_%EC%8A%A4%ED%86%A8%EB%A7%8C)이 [1985년](https://ko.wikipedia.org/wiki/1985%EB%85%84) 3월, [GNU](https://ko.wikipedia.org/wiki/GNU) 프로젝트의 목표를 설명하고 다른 사람들의 참여와 지원을 요청하기 위해 [닥터 돕스 저널](https://ko.wikipedia.org/wiki/%EB%8B%A5%ED%84%B0_%EB%8F%95%EC%8A%A4_%EC%A0%80%EB%84%90)(*Dr. Dobb's Journal of Software Tools* **10**(3))[[1]](https://ko.wikipedia.org/wiki/GNU_%EC%84%A0%EC%96%B8%EB%AC%B8#cite_note-1)에 실은 글이다. 처음에는 프로젝트의 진행에 따라 조금씩 개정되었지만, 현재는 있는 그대로를 보존하는 것이 낫다는 판단하에 그대로 유지되고 있다.

<https://www.gnu.org/gnu/manifesto.ko.html>

/\*선연문은 사이트에서 확인해주세요\*/

===========================================================================

Git 과 역사

<http://platum.kr/archives/55437> (Platum - 오픈소스의 확산을 이끈 깃허브 이야기)

개발자들 사이에서는 예전부터 시작된 오픈소스 운동

오픈소스 운동은 개발자들 사이에서는 오래전부터 진행되어 왔습니다. 개발자들은 자기들이 만든 소스를 서로 공유하고 그 안에서 버그를 찾고 그 버그를 고쳐서 그것을 다시 공유함으로 서로의 소스의 품질을 높히는 작업을 지속적으로 해왔습니다. 개발자들이 보통 자기가 만든 소스에 대해서는 버그를 잘 찾지를 못합니다. 개선점을 찾기가 어렵지요. 그래서 그것을 다른 개발자에게 공개하고 자기가 아닌 다른 사람으로부터 버그를 찾고 개선점을 추천받는 작업을 합니다. 해외의 경우에는 이런 작업들이 많이 일반화 되었습니다. 시스템이 아닌 서비스를 제공하는 기업의 경우 플랫폼을 오픈소스로 된 것으로 만들고 거기에 자기들이 필요한 기능을 더하고 커스터마이징을 해서 서비스화 시켜서 제공합니다. 기반이 되는 기능은 오픈소스로 된 솔루션을 이용함으로 전세계의 수많은 개발자들로부터, 또 커뮤니티로부터 버그에 대한 버그픽스(BugFix, 버그를 수정한 내용)를 받아서 적용함으로 손쉽게 문제점을 해결합니다. 그리고 서비스는 그 기반 위에서 자기들이 별도로 만든 기능을 더해서 제공하는 방식을 많이 이용합니다. 그것이 UX가 되었건 아니면 특별한 어떤 기능이 되었건간에 그렇습니다.

한가지 예를 들면 이런 것입니다. 어느 회사에서 온라인 쇼핑몰을 만든다고 할 때 예전 같았으면 온라인 쇼핑몰이 되는 웹서버와 그 안에 필요한 다양한 CMS(컨텐츠 관리 시스템), 결제 시스템 등을 다 만들어야 했습니다. 하지만 요즘의 경우라면 웹서버에 들어가는 하드웨어는 어디 조립용 PC를 이용할 수도 있을 것이고 그 안에 들어가는 OS는 Linux를 쓸 것이며 DB는 MySQL을 쓰고 웹서버는 아파치를 사용할 것입니다. 그리고 CMS는 워드프레스를 사용하며 거기에 워드프레스용으로 나온 쇼핑몰용 플러그인을 사용할 것입니다. 쇼핑몰을 운영할 회사가 할 일은 어떤 식으로 쇼핑몰을 꾸밀 것인지 하는 디자인에 대한 작업과 어떤 것을 팔 것인지에 대한 컨텐츠만 만들면 됩니다. 서비스를 제공하는데 있어서 오픈소스를 이용하면 이렇게 간단하게 작업이 가능합니다.

오픈소스가 기업에 미치는 영향

오픈소스가 기업에 미치는 영향도 대단합니다. 앞서 개발자들 사이에서 먼저 시작된 오픈소스 이야기를 했는데 최근 많은 기업들은 자사의 솔루션을 오픈소스화 시키고 그 솔루션을 기반으로 하는 서비스를 제공하는 방식으로 회사의 수익모델을 취하고 있습니다. 앞서 쇼핑몰의 예도 그렇지만 어떤 시스템을 만드는데 있어서 예전같았으면 시스템을 구성하는 하드웨어와 그 안의 소프트웨어 모두를 만들어야 했지만 요즘에는 오픈소스를 이용해서 만들기 때문에 그렇게 하지 않습니다. 기반이 되는 소프트웨어 플랫폼(OS나 웹서버, CMS, 그 외의 플랫폼이 되는 소프트웨어 들)은 오픈소스를 가져다가 쓰고 그 플랫폼에 해당 회사가 필요한 기능을 만들어서 집어넣고 그것을 구동시키는 하드웨어는 직접 만들어서 파는 방식을 이용합니다. 결국 기업은 하드웨어와 회사가 따로 만든 기능에 대한 비용, 그리고 그 시스템을 유지보수하는 비용을 받는 것입니다. 물론 플랫폼도 직접 만들어서 함께 제공한다면 그만큼 해당 시스템에 더 높은 비용을 책정할 수 있고 수익도 더 많이 나오겠지만 그 플랫폼을 만드는데 들어가는 시간 및 인력 등의 비용과 그 플랫폼을 유지보수하는데 들어가는 비용 보다는 자사가 필요로 하는 기능에 집중해서 나오는 비용이 더 효과가 좋다는 것을 알기 때문에 기업들도 오픈소스를 적극 활용하는 추세입니다. 그리고 또 자사가 만든 플랫폼을 오픈소스화 시키는 경우도 있는데 앞서 얘기했듯 자기들이 만든 플랫폼에 대해서 버그나 개선사항을 도출하는데 있어서 자사 안에서 해결하는데는 한계가 있기 때문에 소스를 공개하여 전세계의 개발자들을 대상으로 오픈 테스트를 시키는 것으로 봐도 좋을 것입니다. 물론 소스를 오픈시켰기 때문에 다른 기업이 그 소스를 가져다가 사용할 수도 있습니다(이 부분 때문에 오픈소스에도 앞서 언급한 것처럼 다양한 라이선스 정책이 존재합니다). 하지만 가져다 쓴 기업은 그 소스를 그대로 상품화 할 수는 없습니다(라이선스 정책 때문도 있고 소스가 오픈되어 있기 때문에 다른 회사도 같은 방식으로 상품화할 것이며 경쟁이 그만큼 치열하기 때문에 쉽지 않은 부분입니다). 그 위에 기능을 얹어서 차별화시켜야 경쟁력이 생기기 때문에 해당 플랫폼에 그래도 가장 많이 아는 회사가 유리하다고 할 수 있습니다. 그래서 많은 기업들이 자사의 솔루션을 오픈소스화 시키고 그 위에 자사만의 기능을 더 얹어서 패키징하여 판매하는 방식을 이용합니다. 라이선스 정책으로 인해 오픈소스화 된 부분은 소스를 공개해야 하니 그 부분은 공개하지만 그 위에 얹어진 기능은 비공개로 진행이 가능하니, 또 그 솔루션을 기반으로 만들어진 서비스에 대해서는 유료 정책을 적용할 수 있으니 그렇게 서비스 사업모델을 만들어 진행하는 것입니다.

앞서 기업에 미치는 영향에 대해서 얘기했는데 기업의 이미지 개선 및 연계 솔루션의 매출 증대에도 영향을 미칠 수 있습니다. 최근 MS가 자사의 운영 플랫폼인 닷넷(.NET) 프레임워크를 오픈소스화 시켰습니다. 원래는 윈도(MS Windows)에만 운영되는 솔루션입니다. 윈도는 유료 OS이기 때문에 닷넷 프레임워크를 이용하려면 유료 OS인 윈도를 구입해야만 했습니다. 하지만 닷넷 프레임워크를 오픈소스화 시킨 후 많은 개발자들이 이 소스를 이용하여 오픈소스 OS인 리눅스(Linux)용으로 포팅(다른 시스템에 이식시키는 작업)을 진행하고 있습니다. 즉, 닷넷 프레임워크 위에 돌아가는 솔루션들이나 서비스들을 이제는 리눅스에서도 구동시킬 수 있게 된 것입니다. MS는 자사의 운영 플랫폼인 닷넷 프레임워크를 기존보다 더 많은 개발자들에게 제공할 수 있게 되었으며 기업들도 유료 OS인 윈도의 구입에 대한 압박을 받지 않고 무료 OS인 리눅스를 이용하여 닷넷 프레임워크로 개발하고 서비스를 제공할 수 있게 됨으로 비용 절감의 효과를 가져오기에 더 많은 기업들이 닷넷 프레임워크를 도입하게 됩니다. 이는 닷넷 프레임워크의 점유율이 올라가고 동시에 MS에 대한 이미지가 좋아진다는 의미며 그 이미지를 활용하여 더 많은 MS 제품들을 구입하고 이용할 수 있는 마케팅 효과를 가져오게 됩니다.

이렇듯 오픈소스 운동이 미치는 영향은 단순히 개발자들이 공개된 소스를 가져다가 쓸 수 있다는 것에 끝나지 않습니다. 개발자들은 자기들의 기술을 업그레이드 할 수 있는 기회가 되는 것이며 기업 입장에서는 좀 더 효율적인 시스템, 서비스, 솔루션 개발을 가져갈 수 있고 플랫폼을 만드는 기업이나 개발자 입장에서는 더 효과적인 유지보수가 가능하게 되며 이미지 개선 및 부가적으로 연계 서비스 및 솔루션 판매에 좋은 영향을 끼치는 등의 선순환적인 효과를 가져올 수 있다는 점에서 오픈소스 운동은 현재 전세계적으로 크게 일어나고 있다고 볼 수 있습니다.

깃의 대중화를 이끈 깃허브

오픈소스 소프트웨어 깃이란

앞서 얘기한 오픈소스를 체계적으로 관리하기 위해서는 소스 버전 관리 시스템을 이용합니다. 오픈소스 뿐만이 아니라 기업에서도 자사의 솔루션을 체계적으로 관리하기 위해 소스 버전 관리 시스템을 이용합니다. 이전에는 CVS를 많이 이용했고 요즘에는 SVN을 많이 이용합니다. 그 외에도 많은 소스 버전 관리 시스템이 존재합니다. 깃(Git)은 CVS, SVN과 같은 소스 버전 관리 시스템으로 오픈소스 OS인 리눅스의 소스를 관리하기 위해 리눅스 커널의 개발자이자 창시자인 리누즈 토발즈에 의해서 만들어진 분산 소스 버전 관리 시스템입니다.

리눅스의 창시자인 리누즈 토발즈는 리눅스 커널을 압축파일과 패치로만 관리를 해왔습니다. 그러다가 체계적인 소스 관리를 위해 2002년에 상용 소스 버전 관리 시스템인 BitKeeper를 이용하게 됩니다. 하지만 2005년에 리누스 토발즈와 BitKeeper의 관계는 틀어지게 됩니다. 오픈소스 OS인 리눅스의 커널 소스 관리를 상용 소스 버전 관리 시스템인 BitKeeper에 맡기다보니 개념 및 의견 충돌이 지속적으로 나왔던 것입니다. 결국 리누즈 토발즈는 BitKeeper의 사용 경험 및 교훈을 바탕으로 오픈소스에 적합한 소스 버전 관리 시스템을 만들게 되는데 빠른 속도와 단순한 구조, 비선형적 개발(동시다발적으로 수천개의 프로젝트를 관리할 수 있도록 브런치 생성)과 완벽한 분산, 대규모 프로젝트에 적합한 규모 등을 목표로 깃을 만들게 됩니다.

편리한 호스팅 서비스 깃허브

깃허브(GitHub)는 깃을 사용하는 웹 기반의 호스팅 서비스다. 즉, 웹 기반의 분산 소스 관리 시스템인데 그 소스 관리 시스템의 엔진을 깃을 사용한다는 것입니다. 깃허브는 루비 온 레일스(Rubi on Rails)를 기반으로 만들어졌으며 영리적인 솔루션 개발을 위한 프로젝트나 오픈소스 프로젝트에 상관없이 무료로 다 제공해주는 서비스입니다.

일반적으로 소스 버전 관리 시스템은 설치해서 사용하게 되는데 기업에서 CVS나 SVN을 이용할 때에도 PC든 서버든 하드웨어에 윈도(Windows)나 리눅스를 설치하고 그 위에 CVS나 SVN 서버 프로그램을 설치해서 사용하게 됩니다. 그러다보니 초기 구축 비용이 들어가게 되고 유지하는데 비용도 들어가게 됩니다. 하지만 오픈소스를 개발하는 커뮤니티 입장에서는 이런 관리 비용을 집행하기가 어려운 것이 사실입니다. 깃허브는 앞서 깃의 목표 중의 하나인 비선형적 개발 기능을 바탕으로 무료로 손쉽게 웹 기반으로 제공해주기 때문에 수많은 오픈소스가 깃허브를 통해 관리되기 시작하면서 오픈소스 진영에서 가장 인기있는 소스 버전 관리 호스팅 서비스로 자리잡게 됩니다. 특히 웹 기반으로 제공해주기 때문에 SVN이나 CVS처럼 따로 클라이언트를 설치해서 사용할 필요가 없이 웹브라우저만 있어도 관리 및 소스 다운로드가 가능하다는 점 때문에 더 인기를 끌게 됩니다(물론 깃허브도 깃허브 데스크탑이라는 클라이언트를 설치하게 되면 웹 화면을 거치지 않고 CVS나 SVN처럼 바로 소스를 수정하고 관리할 수 있습니다). 이제 깃허브는 오픈소스 프로젝트의 소스 관리 시스템의 대명사로 불리는 수준에 오게 됩니다.

깃허브의 특징 및 장점

깃허브가 다른 소스 버전 관리 시스템과 달리 인기를 끌 수 있었던 이유는 앞서 얘기했던 것처럼 무료 호스팅이라는 점과 웹 기반이라는 점이 한몫 했습니다. 하지만 그것만으로는 깃허브가 이렇게 인기를 끌지는 못했을 것입니다. 깃허브의 가장 큰 장점은 손쉬운 참여입니다. 특히나 Fork 기능을 통해 인기있는 깃허브의 오픈소스 프로젝트를 내 저장소로 끌어와서 손쉽게 소스를 다운로드 하거나 문제가 되는(이슈) 부분을 제기하고 혹은 수정해서 제시할 수 있다는 점이 오픈소스의 손쉬운 접근 및 대중화를 이끈 핵심 기능이라고 할 수 있습니다. 기존에는 오픈소스 프로젝트에 참여하기 위해 해당 커뮤니티에 가입하고 또 소스 수정을 위해 권한을 받아야 하는 등 여러가지 해야 할 작업들이 많았습니다. 즉, 참여하기가 어렵다는 불편한 점이 있었습니다. 하지만 깃허브에 있는 오픈소스 프로젝트들은 Fork 기능을 통해 누구나 해당 오픈소스 프로젝트에 참여할 수 있게 되었고 이슈를 제기하고 또 자신이 수정한 이슈 사항을 올려서 손쉽게 반영할 수 있게 함으로 누구나 오픈소스 프로젝트에 참여할 수 있도록 손쉬운 접근 방법을 제시해줬습니다. 오픈소스가 최근 활성화되고 또 규모가 커진 이유 중 하나가 바로 손쉬운 참여라고 할 수 있는데 깃허브는 그 공로자 중 하나라고 할 수 있을 듯 합니다.

=============================================================================[https://git-scm.com/book/ko/v1/%EC%8B%9C%EC%9E%91%ED%95%98%EA%B8%B0-%EC%A7%A7%EA%B2%8C-%EB%B3%B4%EB%8A%94-Git%EC%9D%98-%EC%97%AD%EC%82%AC](https://git-scm.com/book/ko/v1/%25EC%258B%259C%25EC%259E%2591%25ED%2595%2598%25EA%25B8%25B0-%25EC%25A7%25A7%25EA%25B2%258C-%25EB%25B3%25B4%25EB%258A%2594-Git%25EC%259D%2598-%25EC%2597%25AD%25EC%2582%25AC) (git - 시작하기 - 짧게 보는 Git의 역사)

짧게 보는 Git의 역사

인생을 살다 보면 여러 가지 일들이 벌어지듯이 Git의 삶 또한 창조적인 파괴와 모순 속에서 시작되었다. 리눅스 커널은 굉장히 규모가 큰 오픈소스 프로젝트다. 리눅스 커널의 일생에서 대부분 시절은 패치와 단순 압축 파일로만 관리했다. 2002년에 드디어 리눅스 커널은 BitKeeper라고 불리는 상용 DVCS를 사용하기 시작했다.

2005년에 커뮤니티가 만드는 리눅스 커널과 이익을 추구하는 회사가 개발한 BitKeeper의 관계는 틀어졌다. BitKeeper의 무료 사용이 제고된 것이다. 이 사건은 리눅스 개발 커뮤니티(특히 리눅스 창시자 리누스 토발즈)가 자체 도구를 만드는 계기가 됐다. Git은 BitKeeper를 사용하면서 배운 교훈을 기초로 아래와 같은 목표를 세웠다:

• 빠른 속도

• 단순한 구조

• 비선형적인 개발(수천 개의 동시 다발적인 브랜치)

• 완벽한 분산

• 리눅스 커널 같은 대형 프로젝트에도 유용할 것(속도나 데이터 크기 면에서)

Git은 2005년 탄생하고 나서 아직도 초기 목표를 그대로 유지하고 있다. 그러면서도 사용하기 쉽게 진화하고 성숙했다. Git은 미친 듯이 빨라서 대형 프로젝트에 사용하기도 좋다. Git은 동시다발적인 브랜치에도 끄떡없는 슈퍼 울트라 브랜칭 시스템이다

===========================================================================

Software Or Project?

대표적인 오픈 소스 소프트웨어로는 모바일 플랫폼으로서 각광을 받고 있는 리눅스 커널(linux kernel, 점유율 약 50%),서버시장의 약 30%의 점유율을 보이고 있는 리눅스 OS, 전세계 웹 서버의 60% 이상을 차지하고 있는 아파치(apache), DB 분야의 선두주자인 MySQL, 통합개발환경을 제공하는 이클립스(eclipse) 등이 있다.

우리나라에서 이름난 오픈 소스 소프트웨어로는 과거 NHN이 인수해 전세계의 웹 콘텐츠 플랫폼 시장의 약 60%를 휩쓸고 있는 익스프레스 엔진 ( XpressEngine, XE )이 존재한다.

<http://www.itworld.co.kr/slideshow/79432> (IT WORLD Korea - 2012년 가장 성공적인 오픈 소스 프로젝트 10선)

/\*사이트에 예 있으니 꼭 확인해주세요\*/

Pros & Cons?

장점

* 사용자가 프로그래밍 언어를 아는 경우 소스가 공개되어 있다면 본인이 직접 소프트웨어의 문제를 수정하거나 개선을 할 수 있다

단점

* 상용화가 어렵다 (하지만, 오픈 소스 소프트웨어가 추구하는 방향이 상용화가 아니기 때문에 이것을 장점으로 볼 수 있는 시각도 존재함.)
* 소스를 도용 당할 수 있다.
* 원칙적으로 사후 지원, AS 제공의 의무가 없다.

=====================================================================================

<https://crowdsourcedtesting.com/resources/pros-cons-open-source-software/> (crowdsourcedtesting - Pros and cons of open source software)

장점

무료 또는 저렴한

당신이 지불하고있는 독점적 인 소프트웨어 패키지의 오픈 소스 버전을 찾는 것은 기업에게 진정한 이익이 될 수 있습니다. 값 비싼 라이센스가 필요없는 운영은 특히 중소기업의 경우 모든면에서 효과적 일 수 있습니다. 오픈 소스 소프트웨어는 매년 기업에 엄청난 돈을 절약합니다.

지속적인 개선

오픈 소스 소프트웨어는 개발자가 버그를 수정하고 개조하고 추가 할 때마다 항상 진화하고 있습니다. 개선에 참여하는 대규모 팀이 있으며 모든 사람이 소프트웨어 개선을 위해 주도권을 쥐고 있습니다.

당신이 원하는 것을해라.

라이센스에 대해 걱정할 필요없이 모든 시스템에 설치할 수있을뿐만 아니라 사용자의 요구에 맞출 수 있습니다. 원하는 경우 수정할 수 있습니다.

잠겨 있지 않음

서로 호환되는 제품을 설계하는 공급 업체와 독점 시스템에 얽매이지 않습니다. 또한 공급 업체의 로드맵에서 벗어나므로 항상 필요한 기능을 추가 할 때까지 기다릴 필요가 없습니다. 오픈 소스 소프트웨어는 개별 기업과 독립적이기 때문에 활동적인 커뮤니티가있는 한 원래 창업자가 사업을 중단하거나 버리는 것은 중요하지 않습니다.

단점

숨겨진 비용

많은 기업들은 오픈 소스 소프트웨어가 무료라는 사실에 처음에 매료되었지만 소프트웨어 배포 및 통합 비용을 고려해야합니다. 지속적인 유지 보수 및 지원 비용도 고려해야합니다.

학습 곡선

선택한 오픈 소스 제품을 효과적으로 사용할 수있는 내부 전문가가 있습니까? 반드시 잘 문서화 된 도움말을 찾거나 전화 지원을받을 수는 없습니다. 오픈 소스 소프트웨어는 개발자에게 우선 순위가 아니기 때문에 종종 사용자 친화적이지 않습니다. 기술 차이를 채우기 위해 고용하고 훈련해야 할 수도 있습니다.

포기의 위험

주요 프로그래머가 관심을 잃거나 더 이상 프로젝트 작업을 할 시간이 없다면 매우 빨리 포기 될 수 있습니다. 오픈 소스 프로젝트가 가끔씩 사라집니다. 소프트웨어에 많은 투자를했다면 업데이트가 중단되면 심각한 문제가 될 수 있습니다.

보안 위험

당신은 소프트웨어가 개방되어있는 것이 유익하지만 그것이 다른 사람들에게도 열려 있다는 것을 의미합니다. 악의적 인 사용자는 코드를보고 악용 및 취약점을 찾을 수 있습니다. 회사가 보안 취약성을위한 오픈 소스 코드를 분석하고이를 완화하기위한 조치를 취하지 않으면 안전하다고 생각하는 것은 위험합니다. 보안 위험을 무시하면 심각한 노출로 이어질 수 있습니다.

장점의 잠재적 이점은 단점의 위험성보다 큽니다. 이것이 오픈 소스 소프트웨어가 풍경을 지배하는 이유입니다. 그러나 사례별로 분석하고 보안을 염두에 두는 것이 중요합니다. 많은 기업들도 오픈 소스 사용 및 직원 기여에 대한 공식 정책이 부족합니다. 많은 경우 오픈 소스를 채택하는 것이 현명한 방법이지만 전략이 완전히 고려되었는지 확인하십시오.

=============================================================================

<http://elliottyoung.co.uk/2014/01/06/the-pros-and-cons-of-closed-source-software/> (elliottyoung - The pros and cons of closed source software)

클로즈드 소스 소프트웨어의 장단점

다음 비즈니스 소프트웨어 솔루션을 조사 할 때 폐쇄 형 소스 소프트웨어를 비즈니스에 활용할 때 다음 장단점을 명심하십시오.

폐쇄 된 소스 소프트웨어 전문가

폐쇄 형 소스 소프트웨어를 구입할 때 디스크 이상을 얻게됩니다.

당신은 뭔가를 구입했다는 내용의 문서를 얻을 것이고 약속 된대로 작동 할 것이라고 기대할 것입니다. 그렇게하지 않으면 공급자에게 법적 조치를 취할 수 있습니다.

소금에 합당한 소프트웨어 제공 업체는 구입 한 응용 프로그램에 대한 서비스 / 지원을 어느 정도 제공합니다.

닫힌 소스 소프트웨어 단점

소프트웨어 제공 업체는 끊임없이 제품을 업그레이드해야하는 부담을 안고 있습니다. 즉, 업계를 따라 잡거나 업계를 앞서 가기 위해 막대한 투자가 필요합니다. 당연히 이러한 비용은 사용자에게 전달됩니다.

폐쇄 형 소스 소프트웨어의 대부분은 준비되기 훨씬 전에 시장에 출시됩니다. 많은 소프트웨어 회사는 제품의 개발 / 테스트에 결정적인 것으로서 이러한 관행을 봅니다 (인정할 것 같지는 않지만).

조기 소프트웨어 배포는 신속한 테스트 및 빠른 버그 수정을 의미 할 수 있지만 조기 사용자는 IT 무결성에 심각한 타협을함으로써 제품 개발에 대한 가혹한 가격을 지불하게 될 수도 있습니다.

버그를 수정하는 가장 일반적인 방법은 소프트웨어 제작자가 패치를 제공하는 것입니다. 그러나 사용자는 종종 패치를 설치하는 것이 좋지 않습니다. 즉, 버그가 발견 된 후에도 해당 시스템과 회사의 시스템이 취약한 상태로 유지됩니다.

닫힌 소스 회사는 제품을 만들고 개발할 수있는 유일한 조직이기 때문에 특히 현재 사용중인 컴퓨터의 수를 고려하여 일반적으로 사용되는 소프트웨어의 버전이 적습니다. 즉, 변형 된 바이러스 작성자는 수천 명의 컴퓨터를 악용하여 전체 네트워크를 손상시키고 때로는 기업을 정지 상태로 만들 수 있습니다.

===========================================================================

<https://www.coredna.com/blogs/comparing-open-closed-source-software> (Core DNA - Comparing Open Source vs Closed Source Software)

오픈 소스 소프트웨어 (OSS)는 다른 사용자 및 조직이 컴퓨터 코드를 공유, 확인 및 수정할 수있는 라이센스 계약에 따라 배포됩니다.

또는 더 사용자 편한 언어로, 오픈 소스 소프트웨어는 일반인이 원래 디자인에서 무료로 사용하고 수정할 수 있습니다. 이것이 의미하는 바는 소프트웨어는 세계의 다른 개발자가 진화하고 반복 할 수 있다는 것입니다. 이상적으로 이것은 소프트웨어가 시간이 지남에 따라 향상된다는 것을 의미하지만, 종종 그 모든 진화와 함께 흥미로운 우회로 및 회전을 충분히 취할 수 있으며 형태와 형태를 완전히 바꿀 수 있습니다.

오픈 소스에 대한 아이디어와 오픈 소스 소프트웨어의 정신을 주도하는 자치 원칙에 대해 자세히 설명한다. 개방적이고 피어 투 피어 지향 웹은 철학적으로 박수를 쳤지만 자신의 이익을 위해 일을 깰 수있는 불량 개발자에게 취약해질 수 있습니다. 따라서 경고 라벨이 필요합니다.

바로 그때. 그래서 Closed Source는 무엇입니까?

폐쇄 형 소스 소프트웨어는 허가 된 사용자에게 개인적 수정, 복사 및 재 게시 제한을 통해 라이센스 계약에 따라 배포 된 독점 소프트웨어로 정의 할 수 있습니다.

또는 평신도 용어로 볼 때 소스 코드는 누구나 보거나 변경할 수있는 공개 소스와 공유되지 않습니다 . 닫힌 소스는 오픈 소스의 반대입니다.

닫힌 소스는 실제로 대부분의 비즈니스에서 기대할 수있는 정렬 방식으로, 제품을 보호하고 브랜드에 대한 통제력과 고객에게 제공되는 사용자 경험을 유지하려고합니다. Android보다는 Apple을 생각하십시오.

따라서 오픈 소스 또는 폐쇄 소스 (독점 소프트웨어)를 고려할 때 결정을 내리기 전에 고려해야 할 주요 차이점은 무엇입니까? 우리는 오픈 소스 및 폐쇄 소스 소프트웨어에서 서비스 / 지원, 혁신, 유용성 및 보안을 살펴보고 두 소프트웨어 시스템의 장단점에 대해 간략히 설명합니다.

오픈 소스 소프트웨어와 오픈 소스 소프트웨어의 주요 차이점은 무엇입니까?

일반적으로 개방과 폐쇄 간의 주요 차별화 요소는 다음과 같은 몇 가지 요인에 달려 있습니다.

. 비용

. 서비스

. 혁신

. 유용성

. 보안

장단점과 장단점이 있으며, 앞으로 나아갈 방향은 크게 5 가지 요인 각각에 대한 우선 순위에 달려 있습니다. 이러한 우선 순위는 오픈 소스 사용과 클로즈드 소스 CMS 사용시기를 결정하는 데 도움이됩니다.

개방형 소스 소프트웨어와 폐쇄 형 소프트웨어의 차이 # 1 : 비용

오픈 소스 소프트웨어의 주요 이점 중 하나는 비용입니다. 그러나 OSS에 적용될 때 "자유"라는 용어는 전반적인 비용과 관련이 없으며 제한으로부터 자유 로워집니다.

직원이 소프트웨어를 유지 관리 할 수있는 사내 역량 및 기술 전문 지식과 리소스, 구현, 교육 및 지원을 제공하는 경우 오픈 소스가 조직에 가장 경제적 일 수 있습니다. 그러나 회사의 발전, 기술 변화 및 요구가 커짐에 따라 구현, 혁신, 지원 제공 및 인프라 투자에 대한 장기적인 비용을 고려해야합니다.

개방형 소프트웨어 제공 업체는 추가 기능, 통합 및 추가 서비스와 같은 추가 비용을 청구하는 경우가 있습니다. 이로 인해 경우에 따라 비용 절감 이점이 없어 질 수 있습니다. 결국 무료가 아니라 오픈 소스 소프트웨어로 서비스 비용을 지불하고 있습니다.

폐쇄 형 소스 CMS의 경우 시스템 복잡성에 따라 소프트웨어, 통합 및 서비스에 대한 기본 요금 및 연간 라이센스 / 지원 비용을 포함하여 수 천 달러에서 수십만 달러 사이의 비용이 달라질 수 있습니다. 하드 비용이 더 높을 수는 있지만 신뢰할 수있는 브랜드의 더 맞춤형 제품, 높은 수준의 보안 및 기능, 지속적인 혁신, 뛰어난 확장 성, 지속적인 교육 및 지원, 기술 기술에 대한 요구 사항은 낮습니다.

오픈 소스 대 소프트웨어 소스 차이 # 2 : 서비스

오픈 소스 소프트웨어는 포럼 및 블로그를 통해 지원을 제공하기 위해 충성도가 높은 온라인 사용자 커뮤니티에 의존하지만, 이러한 지원은 많은 소비자가 기대하는 높은 수준의 응답을 제공하지 못하는 경우가 있습니다 (독점 소프트웨어로받을 수 있음).

이러한 커뮤니티는 웹에서도 찾아 져야하며 일부 커뮤니티는 사용자의 문제를 해결할 인센티브가 없다고 주장합니다.

아마도 독점 소프트웨어 (폐쇄 형)를 사용하면 서비스 및 지원이 가장 큰 장점 일 것입니다. 지속적인 기술 지원은 기술력이 거의없는 사용자에게 중요한 판매 포인트이며 오픈 소스 소프트웨어에 비해 폐쇄 소스를 선택하는 주된 이유 중 하나입니다.

지원에는 제품 및 서비스에 정통한 전문가와 함께 실행 가능한 회사의 즉각적인 지원을위한 사용자 매뉴얼 및 연락 담당자가 포함됩니다.

오픈 소스 대 소프트웨어 소스 차이 # 3 : 혁신

오픈 소스 소프트웨어는 제한없이 소프트웨어를 변경할 수있는 많은 유연성과 자유를 제공합니다. 그러나이 혁신은 모든 사용자에게 전달 될 수는 없으며 원본 소스 코드에 대한 사용자 지정 변경이 소프트웨어의 향후 지원 및 성장을 제한 할 수 있는지 여부가 논의됩니다. 오픈 소스 소프트웨어 제공 업체들은 대규모 연구 개발을 유치하기 위해 종종 노력합니다.

개방형 소스의 제한없는 유연성과 비교할 때 폐쇄 형 소스 소프트웨어에서 소스 코드를 보거나 변경할 수 없다는 단점이 있습니다. 그러나이 제한은 모든 사용자에게 완전히 테스트되고 제공되는 독점 소프트웨어의 보안 및 안정성을 보장합니다.

한 번 더, 맞춤형 소프트웨어는 특정 사용자가 사용할 수 있습니다. 오픈 소스와 달리 독점 소프트웨어는 정기적으로 새로운 제품과 업그레이드를 제공하기 위해 더 많은 양의 R & D를 유치합니다.

오픈 소스 소프트웨어와 마찬가지로 클로즈드 소스 소프트웨어는 포럼과 설문 조사를 통해 아이디어와 전략을 공유하고 혁신을 촉진하고 제품이 변화하는 요구에 적응할 수 있도록하는 온라인 커뮤니티를 제공합니다.

개방형 소스 소프트웨어와 폐쇄 형 소프트웨어의 차이 # 4 : 유용성

기술은 일반적으로 유용성 전문가에 의해 검토되지 않고 대다수의 일반인 사용자보다는 개발자에게 적합하기 때문에 사용성은 종종 오픈 소스 소프트웨어에 대한 비판의 주요 영역입니다. 사용자 가이드는 법으로 요구되지 않으므로 종종 무시됩니다. 매뉴얼이 작성되면, 종종 따르기 어려운 전문 용어로 가득 차 있습니다.

닫혀 있거나 독점적 인 소프트웨어의 경우, 사용성은 더 많은 타겟 고객을위한 전문 사용성 테스트로 인해 높은 판매 포인트 (Apple이 대표적)입니다. 지원 서비스는 소프트웨어 사용을 극대화하는 데 도움이되는 반면 사용자 설명서는 즉각적인 참조 및 빠른 교육을 위해 제공됩니다. 타사 시스템과 개발자는 "닫힌"소스 소프트웨어를 향상시키기 위해 다양한 메커니즘을 사용할 수도 있습니다.

열린 소스 소프트웨어와 닫힌 소스 소프트웨어의 차이 # 5 : 보안

소프트웨어가 통제 된 환경에서 항상 개발되는 것은 아니기 때문에 오픈 소스의 보안은 대기업의 관심사입니다.

전 세계의 개별 사용자가 소프트웨어를 개발할 때 효과적인 의사 소통을 방해하는 연속성과 공통적 인 방향이 없습니다. 다시 한 번, 소프트웨어가 항상 동료 검토 또는 유효성 검사를받는 것은 아니므로 프로그래머가 사용자가 현명하지는 않지만 백도어 트로이 목마를 소프트웨어에 포함시킬 수 있습니다.

이러한 잠재적 인 위험을 줄이는 한 가지 방법은 강력한 온라인 커뮤니티가 지원하는 집중된 개발 팀과 함께 평판이 좋은 브랜드를 채택하는 것입니다.

독점 또는 폐쇄 형 소프트웨어는 일반적으로 공통된 방향으로 집중된 팀에 의해 통제 된 환경에서 개발되기 때문에보다 안전하다고 판단됩니다. 이 팀은 소스 코드를 보거나 편집 할 수있는 유일한 그룹이며, 심하게 감사되며 백도어 트로이 목마 또는 버그의 위험이 감소합니다.

개방형 소스 소프트웨어와 폐쇄 형 소스 소프트웨어의 주요 장점 및 단점은 소프트웨어 유지 관리 및 업데이트에 사용할 수있는 기술 전문 지식과 리소스에 크게 좌우됩니다. 현재와 미래의 귀사 요구에 맞는 올바른 소프트웨어에 대한 더 나은 아이디어를 얻으려면이 기사에 요약 된 5 가지 사항을 고려하십시오.

[http://itnp.kr/roller/home/entry/%EC%98%A4%ED%94%88%EC%86%8C%EC%8A%A4\_open\_source\_%EA%B0%9C%EC%9A%94](http://itnp.kr/roller/home/entry/%25EC%2598%25A4%25ED%2594%2588%25EC%2586%258C%25EC%258A%25A4_open_source_%25EA%25B0%259C%25EC%259A%2594)

(Blog 사람과 기술 – 오픈 소스의 개요)

[https://ko.wikipedia.org/wiki/%EC%98%A4%ED%94%88\_%EC%86%8C%EC%8A%A4\_%EC%9D%B4%EB%8B%88%EC%85%94%ED%8B%B0%EB%B8%8C](https://ko.wikipedia.org/wiki/%25EC%2598%25A4%25ED%2594%2588_%25EC%2586%258C%25EC%258A%25A4_%25EC%259D%25B4%25EB%258B%2588%25EC%2585%2594%25ED%258B%25B0%25EB%25B8%258C)

(위키피디아 OSI – 오픈소스 이니셔티브) [1] [2]

[https://ko.wikipedia.org/wiki/%EC%98%A4%ED%94%88\_%EC%86%8C%EC%8A%A4\_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4](https://ko.wikipedia.org/wiki/%25EC%2598%25A4%25ED%2594%2588_%25EC%2586%258C%25EC%258A%25A4_%25EC%2586%258C%25ED%2594%2584%25ED%258A%25B8%25EC%259B%25A8%25EC%2596%25B4)

(위키피디아 Open Source Software – 오픈소스 소프트웨어) [2]

<http://www.ddaily.co.kr/news/article.html?no=115248>

(디지털 데일리 – [SW법 바로알기 3] 오픈 소스 소프트웨어의 역사)