오픈소스 소프트웨어

2014036817 홍순상

2017011685 김다인

2017011930 모지환

2017012124 송현일

2017012506 정충호

1. 오픈소스의 정의
    1. 자유 소프트웨어
    2. 오픈소스 소프트웨어
    3. 오픈소스 라이선스
    4. 자유 소프트웨어와 오픈 소스
2. 오픈소스의 역사
    1. 리처드 스톨먼
    2. 자유 소프트웨어 운동
    3. GNU 프로젝트
    4. 오픈소스
    5. 오픈소스 소프트웨어 깃(Git)
    6. 깃허브(GitHub)
3. 오픈소스 vs 클로즈드 소스
    1. 클로즈드 소프트웨어
    2. 오픈 소스 vs 클로즈드 소스
4. 유명 오픈소스 SW들

1. 오픈 소스의 정의

자유 소프트웨어

1980년대 리차드 스톨먼은 소스 코드를 공개해 누구나 소프트웨어를 수정, 복제, 배포를 허용하도록 하는 자유 소프트웨어 운동을 창시하였다. 여기서, 자유 소프트웨어의 정의는 1986년 2월, 자유 소프트웨어 재단(이하 FSF)이 GNU의 FSF 회보를 통해 아래의 내용과 같이 발행하였다.

"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."

위의 정의는 1996년 gnu.org 웹사이트가 공개되었을

때, 자유 소프트웨어는 소프트웨어를 연구할 자유에 대한 명시적 언급(이는 프로그램을 수정할 자유의 일부로 이해될 수 있었다)을 추가해 "세 가지 수준의 자유"를 의미하는 것으로 바뀌었다. 이후 리차드 스톨먼은 우리는 모든 자유가 필요하다 말하며 수준에 대해 생각하는 것이 오해를 불러올 수 있기에 '수준'이라는 단어를 사용하지 않았다.

마지막으로, 사용자가 프로그램을 수행할 수 있어야 한다고 명시적으로 언급하는 또 다른 자유가 추가되었다. 새로 추가 되었지만 가장 앞에 와야 했던 이 자유는 "자유 0"으로 추가되었다.

현재는 소프트웨어를 받은 사람이 다음 네 가지 자유를 가졌는지 여부에 따라 자유 소프트웨어를 정의한다.

•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 코드가 공개될 것을 요구한다.

즉, 자유 소프트웨어란, 최종 사용자가 소프트웨어를 사용하고, 연구하고, 수정하고, 공유할 자유를 보장하는 소프트웨어를 의미한다.

1998년, 자유 소프트웨어는 오픈 소스 소프트웨어(이하, 오픈 소스)로 용어가 변경된다. 이는 자유(free)란 용어 때문에 일반인들이 무료라고 인식하고 있다는 점, 라이선스의 엄격성 때문에 소프트웨어 개발이 용이하지 않다는 점을 탈피하기 위해서 였다. 용어의 변경과 더불어 자유 소프트웨어의 코드를 공유하자는 개념에는 동의하지만 리차드 스톨먼의 개인적인 철학에는 동의하지 않은 프로그래머들이 자유 소프트웨어 운동의 대안으로 오픈 소스 운동을 창시하였다.

오픈 소스 소프트웨어

'오픈 소스 정의'의 관리 및 촉진을 담당하는 비영리 단체, 오픈 소스 이니셔티브(OSI)는 오픈 소스의 정의(Open Source Definition)를 아래와 같이 내렸다.

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

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 그 밖의 다른 비용을 요구해서도 안됩니다.

2. 원시 코드

오픈 소스 프로그램에는 원시 코드(source code)가 포함되어야 하며, 컴파일 된 형태뿐 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 제공 되어야만 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.

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.

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.

5. 개인 및 단체에 대한 차별 금지

오픈 소스 사용 허가는 특정 개인이나 단체를 차별해서는 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.

7. 사용 허가의 배포

프로그램에 대한 권리는 배포에 따른 각 단계에서 배포자에 의한 별도의 사용 허가 없이도 프로그램을 재배포 받은 모든 사람에게 동일하게 인정 되어야만 합니다.

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 재배포 받은 모든 사람에게 최초의 소프트웨어 배포판을 통해 프로그램을 배포 받은 사람과 동일한 권리가 보장 되어야만 합니다.

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. 』

위 기준을 통과한 소프트웨어는 OSI가 인증하는 공개 소스 소프트웨어 인증 마크를 통해 소프트웨어가 실제로 공개 소스라는 것을 증명하고, 공개 소스의 복제도 가능하게 한다.

OSI 인증 마크

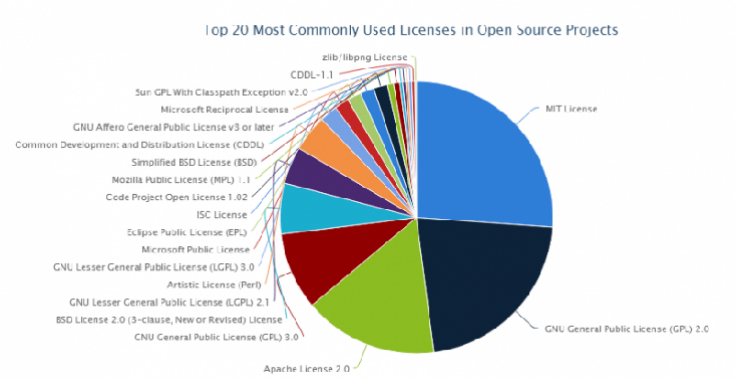
즉, 오픈 소스란, 소스 코드를 공개하여 누구나 특별한 제한 없이 그 코드를 보고 사용 할 수 있게 하는 '오픈 소스 사용허가'를 만족하는 소프트웨어를 뜻한다. 그렇다면 '오픈 소스 사용 허가'란 무엇일까?

오픈 소스 라이선스

오픈 소스 사용 허가라고도 하는 오픈 소스 라이선스는 오픈소스 개발자와 이용자 간에 이용 방법 및 조건의 범위를 명시한 계약이다.

오픈소스 라이선스는 각각의 라이선스 마다 조금씩 다르지만 크게 3가지 공통 사항이 있다. 개발자 정보를 임의로 수정하거나 삭제하여서는 안된다는 '저작권 관련 문구 유지', 제품명이 상표권으로 보호받는 '제품명 중복 방지', 기존에 만들어진 코드를 재사용하거나 결합할 때 각 코드의 서로 다른 라이선스의 조합을 확인해야 하는 '서로 다른 라이선스의 SW 조합 시 조합 가능 여부 확인' 등이 있다. 선택적으로는 '소스코드 공개', '특허관련 사항 준수' 등이 있다.

다음은 주요 오픈소스 라이선스들이다.



오픈소스 라이선스 이용 비율

GNU GPL(General Public License)

자유 소프트웨어 재단에서 만든 자유 소프트웨어 라이선스로 리처드 스톨먼이 GNU-프로젝트로 배포된 프로그램의 라이선스로 사용하기 위해 작성했다. 'GPL에 의해 배포된다'는 사실을 명시해야 하며, 소스 코드를 수정하는 경우 반드시 개작 부분은 공개해야 하고, 소스 코드를 링크하는 경우에도 모든 소스 코드를 공개해야 한다. 현재 GPL은 전체 오픈소스 가운데 2번째로 많이 사용되고 있다.

LGPL(Lesser General Public License)

GPL보다 완화된 라이선스로, 전체적으론 GPL과 동일하나 소스 코드를 링크하는 경우 소스를 공개할 필요가 없고 상용 소프트웨어로 판매가 가능하다.

MIT License

MIT 라이선스(MIT License)는 미국 매사추세츠 공과대학교(MIT)에서 해당 대학의 소프트웨어 공학도들을 돕기 위해 개발한 라이선스다. 해당 라이선스가 붙은 소프트웨어를 개조하였을 때 반드시 오픈 소스로 배포해야 한다는 규정이 없으며 GPL의 엄격함을 피하려는 사용자들에게 인기가 있다. 대표적 소프트웨어로 X 윈도 시스템이 있다.

AL(Apache License)

아파치는 고성능의 하이퍼텍스트 전송 규약(HTTP) 서버로 미국 일리노이 대학의 전미 슈퍼컴퓨터 응용 연구소(NCSA)에서 만든 'NCSA-httpd 1.3'이라는 전시용 프로그램을 근거로 기능 추가와 개량을 거듭해서 개발된 프로그램 등을 대표하는 라이선스이다. 코드의 수정 부분에 대해 소스 코드 공개의 의무는 없으며 소스 코드를 상용 프로그램과 조합하는 것도 허용한다. 단, '아파치'라는 표장에 대한 상표권을 침해 하지 말아야 한다.

BSD(Berkeley Software Distribution)

유닉스의 양대 뿌리 중 하나인 버클리의 캘리포니아 대학에서 배포하는 공개 소프트웨어의 라이선스이다. GNU 자유 소프트웨어 재단의 일반 공중 라이선스(GPL)보다 훨씬 개방적인 4개항의 간단한 문구로 되어 있다. 수정 부분에 대해 소스 코드 공개는 의무 없으며 소스 코드를 상용 프로그램과 조합하는 것도 허용한다.

자유 소프트웨어와 오픈 소스

자유 소프트웨어 운동과 오픈 소스 운동 간의 근본적인 철학적 차이에도 불구하고 자유 소프트웨어 재단의 자유 소프트웨어에 대한 공식 정의와 오픈 소스 이니셔티브의 오픈 소스 소프트웨어에 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."

2. 오픈 소스의 역사

리처드 스톨먼

리차드 스톨먼은 자유 소프트웨어, 오픈 소스에 매우 중요한 인물이고, 역사 부분에서 계속해서 언급 될 인물이므로 먼저 간단하게 소개하고 넘어가겠다.

리차드 매튜 스톨먼(Richard Matthew Stallman, 1953년 3월 16일 ~)은 자유 소프트웨어 운동의 도덕적, 정치적, 법적인 기초를 세우는 데 본질적인 영향을 준 인물이며, 이는 독점 소프트웨어 개발과 공급에 대한 대안이 되었다. 스톨먼은 GNU 프로젝트와 자유 소프트웨어 재단의 설립자이며 이를 지원하기 위해 카피레프트의 개념을 만들었다. 현재 널리 쓰이고 있는 일반 공중 사용 허가서(GPL) 소프트웨어 라이선스의 개념 또한 도입하였다.

스톨먼은 탁월한 프로그래머이기도 하다. 그는 문서 편집기인 Emacs, GNU 컴파일러 모음 컴파일러, GDB 디버거 등 많은 프로그램을 만들었으며, GNU 프로젝트의 일부로 만들었다.

리차드 스톨먼 (1953년 3월 16일 ~)

자유 소프트웨어 운동

오픈 소스의 역사는 자유 소프트웨어 운동과 함께 시작되었다. 자유 소프트웨어 운동 (Free Software Movement)은 리처드 스톨먼이 1980년대에 소프트웨어의 본래 생산 유통 방식인 정보 공유의 방식을 복원하고자 한

운동이다. 이 프로젝트의 핵심 작업은 운영 체제를 만들어 여러 사람들의 손을 거쳐 더 완성도 높은 소프트웨어를 만드는 것이다. 그리고 이 운동의 목표는 운영 체제만이 아닌 모든 소프트웨어를 자유 소프트웨어로 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.

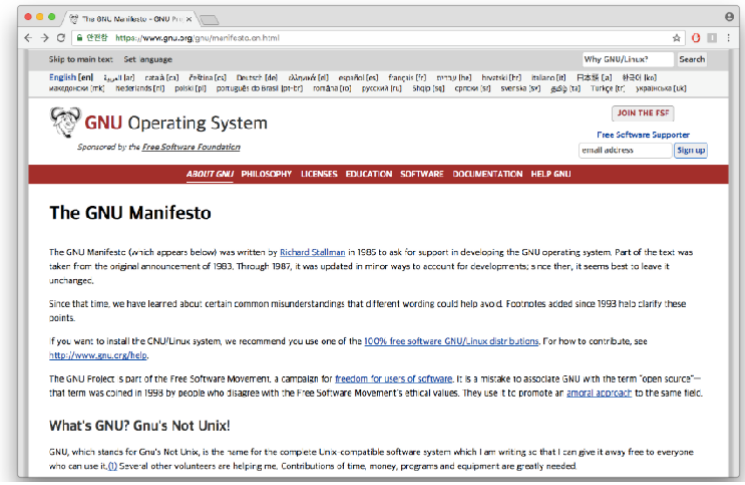
GNU 프로젝트

1984년 리차드 스톨먼은 GNU(그누) 프로젝트를 시작하면서 누구나 자유롭게 "실행, 복사, 수정, 배포"할 수 있고, 누구도 그런 권리를 제한하면 안 된다는 라이선스(GNU 일반 공중 사용 허가서, GNU General Public

License) 아래 소프트웨어를 배포했다.

GNU란, 'GNU's Not Unix'의 약자로 유닉스와 완벽하게 호환되는 소프트웨어 시스템의 이름이며, 원하는 모든 사람이 자유롭게 사용할 수 있도록 만들어진 것이다. 유닉스는 '제법 쓸 만한 운영체제' 지만 유닉스의 비공개 협정이나 소프트웨어 사용권 계약에 동의 할 수 없었던 스톨먼은 유닉스와 호환되면서 사용이 제한되는 소프트웨어 없이도 작업을 해 나갈 수 있는 충분한 자유 소프트웨어를 만들기로 결심했다. 이것이 GNU 프로젝트로 앞서 말했듯이 운영체제에서 끝나는 것이 아니라 모든 소프트웨어를 자유 소프트웨어로 만드는 것이 이 프로젝트의 목적이다.

이후 리처드 스톨먼은 1985년 GNU프로젝트를 철학적, 법률적, 금융적으로 지원하기 위해 자유 소프트웨어 재단(Free Software Foundation, FSF) 재단을 설립하였고, FSF는 GNU 선언문을 제창하였다.



GNU 선언문 https://gnu.org

GNU 프로젝트의 목표인 '소프트웨어는 공유돼야 하며 프로그래머는 소프트웨어로 돈을 벌어서는 안 된다'는 내용과 다른 사람들의 참여와 지원을 요청하기 위함이였다. 또한 이를 지원하기 위해 독점적인 의미의 저작권(copyright)에 대응하는 카피 레프트 운동도 주창했다. 카피레프트는 저작권에 기반을 둔 사용 제한이 아니라 저작권을 기반으로 한 정보의 공유를 위한 조치이다.

1990년까지 GNU 시스템엔 확장 가능한 문서 편집기(이맥스), 뛰어난 최적화 컴파일러(GCC), 그리고 표준 유닉스 배포판의 핵심 라이브러리와 유틸리티가 있었다.

하지만, 여기엔 주요 구성요소인 커널이 빠져 있었다. 이에 대해 스톨먼은 "기본적인 커널은 있지만 유닉스를 흉내 내려면 아직 더 많은 기능이 필요하다"라 말했다.

1991년 핀란드의 대학생이었던 리누스 토발즈(Linus Torvalds)는 GNU 개발도구를 이용해서 리눅스 커널을 개발하는데, 그의 커널은 그동안 개발은 되었지만 많은 부분 문제가 있었던 GNU 프로젝트 커널을 대체하면서 실체화가 가능한 운영체제로 거듭나게 된다. 이것이 바로 오늘날 운영체제 계보에 있어 가장 커다란 영향력을 행사하고 있는 리눅스(Linux)이다. 이후 다른 여러 프로그래머들은 인터넷을 통해 리눅스를 더욱 발전시켰고, 1992년 리눅스는 GNU 시스템과 통합되었고, 이로써 완전한 공개 운영 체제가 탄생되었다.

GNU GPL로 배포된 Linux의 보급과 인터넷의 보급은 자유소프트웨어 운동을 확산하였다.

1997년 7월 브루스 페렌스(Bruce Perens)는 데비안 자유 소프트웨어 가이드라인을 발행하였다. 이 가이드라인은 차후 "오픈 소스의 정의"라는 이름으로 오픈 소스 이니셔티브가 사용하게 된다. 이 때 변경된 유일한 부분은 "자유 소프트웨어"라는 용어 대신 이에 대한 OSI의 다른 용어 "오픈 소스 소프트웨어"가 사용되었다는 것뿐이다.

오픈소스

1998년 넷스케이프는 마이크로소프트의 웹 브라우저인 인터넷 익스플로러에 자사의 웹 브라우저 모질라가 밀려 어려움을 겪고 있었다. 결국 넷스케이프(Netscape)사는 모질라의 소스코드를 공개하는 결정을 하게 됐다.

모질라의 소스코드를 공개하는 과정에서 "자유 소프트웨어"가 "오픈 소스 소프트웨어"로 용어가 변경된다. 이는 '소개'에서도 언급했지만 자유(free)란 용어 때문에 일반인들이 무료라고 인식하고 있다는 점, GPL 조항의 엄격성 때문에 소프트웨어 개발이 용이하지 않다는 점을 탈피하기 위해서 용어가 변경되었다.

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 앞서 오픈 소스 정의에서 소개하였다.

▲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)

오픈소스 소프트웨어 깃(Git)

다수의 인원이 참여하는 오픈 소스 프로젝트를 체계적으로 관리하기 위해선 소스 버전 관리 시스템이 필요하다. 소스 버전 관리 시스템은 파일 변화를 시간에 따라 기록했다가 나중에 특정 시점의 버전을 다시 꺼내 올 수 있는 시스템이다. 이 버전 관리 시스템을 사용하면 각 파일을 이전 상태로 되돌릴 수 있고, 프로젝트를 통째로 이전 상태로 되돌릴 수 있고, 시간에 따라 수정 내용을 비교해 볼 수 있고, 누가 문제를 일으켰는지도 추적할 수 있고, 누가 언제 만들어낸 이슈인지도 알 수 있다. 이 시스템을 사용하면 파일을 잃어버리거나 잘못 고쳤을 때도 쉽게 복구할 수 있다.

CVS, SVN 등 많은 소스 버전 관리 시스템이 존재하며 대표적인 소스 버전 관리 시스템인 깃(Git)은 오픈소스 OS

인 리눅스의 소스를 관리하기 위해 리눅스 커널의 개발자이자 창시자인 리누즈 토발즈에 의해서 만들어졌다.

리눅스 커널은 굉장히 규모가 큰 오픈소스 프로젝트다. 리눅스 커널은 개발 초기부터 패치와 단순 압축 파일로만 관리했다. 2002년부터 리눅스 커널은 상용 소스 버전 관리 시스템인 BitKeeper를 사용하기 시작했다. 하지만 2005년 리누스 토발즈와 BitKeeper의 관계는 틀어지게 된다. 오픈소스 OS인 리눅스의 커널 소스 관리를 상용 소스 버전 관리 시스템인 BitKeeper에 맡기다 보니 개념 및 의견 충돌이 지속적으로 나왔던 것이다. 이 사건은 리눅스 개발 커뮤니티(특히 리눅스 창시자 리누스 토발즈)가 자체 도구를 만드는 계기가 되었다. Git은 BitKeeper를 사용하면서 배운 교훈을 바탕으로 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 2005년에 개발 된 이후 아직도 초기 목표를 그대

로 유지하면서 사용하기 쉽게 진화하였다.

Git 로고

깃허브(GitHub)

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

일반적으로 소스 버전 관리 시스템은 설치 후 사용하는데 기업에서 CVS나 SVN을 이용할 때에도 PC든 서버든 하

드웨어에 윈도우(Windows)나 리눅스를 설치하고 그 위에 CVS나 SVN 서버 프로그램을 설치해서 사용한다. 그러다 보니 초기 구축 비용이 들어가고 유지 비용도 들어간다. 하지만 오픈소스를 개발하는 커뮤니티 입장에서는 이런 관리 비용을 집행하기가 힘들다. 하지만 깃허브는 앞서 깃의 목표 중의 하나인 비선형적 개발 기능을 바탕으로 무료로 손쉽게 웹 기반으로 제공한다. 덕분에 수많은 오픈소스가 깃허브를 통해 관리되기 시작하면서 오픈소스 진영에서 가장 인기있는 소스 버전 관리 호스팅 서비스로 자리 잡는다. 특히 웹을 기반으로 제공하기 때문에 SVN이나 CVS처럼 따로 클라이언트를 설치 하여 사용할 필요가 없이 웹 브라우저만 있어도 관리 및 소스 다운로드가 가능하다는 점 때문에 더 인기를 끌게 된다. 이제 깃허브는 오픈소스 프로젝트의 소스 관리 시스템의 대명사로 불리는 수준까지 오게 된다.

깃허브의 특징 및 장점

깃허브가 다른 소스 버전 관리 시스템과 달리 인기를 끌 수 있었던 이유는 위에서 언급했던 것처럼 무료 호스팅이라는 점과 웹 기반이라는 점이 /한몫 했다. 하지만 깃허브가 인기를 끌게 된 가장 큰 이유는 손쉬운 참여가 가능하다는 점이다. 특히 Fork 기능을 통해 인기 있는 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 말 할 수 있다. 기존에는 오픈소스 프로젝트에 참여하기 위해 해당 커뮤니티에 가입하고 또 소스 수정을 위해 권한을 받아야 하는 등 여러가지 해야 할 작업들이 많았다. 즉, 참여하기가 어렵다는 불편한 점이 있었다. 하지만 깃허브에 있는 오픈소스 프로젝트는 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 제시해주었다. 오픈소스가 최근 활성화되고 또 규모가 커진 이유 중 하나가 바로 손쉬운 참여라고 할 수 있는데 깃허브는 그 공로자 중 하나라고 할 수 있다.

GitHub 로고

3. 오픈소스 vs 클로즈드 소스

클로즈드 소프트웨어

클로즈드 소스 소프트웨어(이하 클로즈드 소스), 또는 사유 소프트웨어는 허가 된 사용자에게 개인적 수정, 복사 및 재 게시 제한을 통해 라이센스 계약에 따라 배포 된 독점 소프트웨어로 정의 할 수 있다. 일반적으로는 소프트웨어를 기술적으로 변형하거나 변조할 수 없도록 저작권을 통해 소스 코드로 접근하는 것을 막는다. 클로즈드 소스는 실제로 대부분의 비즈니스에서 기대할 수 있는 방식으로, 프로그램의 소스코드는 제작사의 기업 비밀로 간주되어 이진 파일 형태로만 제공된다

클로즈드 소스라는 용어는 프로그램의 소스 코드가 공개되지 않는 비공개 소스와 혼동될 수 있다. 마이크로소프트의 공유 소스는 소스 코드를 공개 하지만 몇몇 국가 정부와 IT기업, 학교 등 일부에게만 공개하고 코드의 수정, 재배포가 불가능 하기에 "공유 소스"이지만 "오픈 소스"는 아닌 예이다. 다시 말해, 클로즈드 소스라는 용어를 오픈 소스의 정의에 부합하지 않는 저작권을 갖고 있어서 오픈 소스의 반대 개념으로 사용한다면 공유 소스는 클로즈드 소스의 한 형태에 해당된다.

대표적인 클로즈드 소스와 오픈소스

오픈 소스 vs 클로즈드 소스

그렇다면 오픈 소스와 클로즈드 소스는 구체적으로 어떻게 다른 것일까? 오픈 소스와 클로즈드 소스의 차이는 크게 5가지, 비용, 서비스, 혁신, 유용성, 보안으로 나누어 살펴 볼 수 있다.

1. 비용

오픈 소스 소프트웨어의 주요 이점 중 하나는 비용이다. 오픈 소스 커뮤니티에서 공개된 소스를 가져옴으로써 돈을 지불해야 하는 클로즈드 소프트웨어에

비해 비용을 훨씬 절감할 수 있다.

하지만 단순히 가져오는 것에서 끝나는 것이 아니라, 소프트웨어 배포 및 통합 비용, 지속적인 유지 보수 및 지원 비용 등 장기적인 비용도 고려 해야 한다. 또한, 오픈 소스 제공 업체는 추가 기능, 통합 및 추가 서비스와 같은 추가 비용을 청구하는 경우도 있다. 이로 인해 경우에 따라 '비용 절감'이라는 이점이 없어 질 수 있으며 결국 오픈 소스는 무료가 아닌 서비스 비용을 지불하는 것이다.

클로즈드 소스의 경우 소프트웨어에 따라 소프트웨어와 통합 서비스에 대한 기본 요금 및 연간 라이선스 비용은 무료에서 수십만 달러까지 천차만별이다. 클로즈드 소스는 오픈 소스에 비해 비용이 들지만 뛰어난 확장성, 지속적인 교육 및 지원을 제공하며, 사용자에게 요구 되는 기술 수준이 낮다.

2. 서비스

오픈 소스 소프트웨어는 포럼 및 블로그를 통해 지원을 제공하기 활발한 온라인 사용자 커뮤니티에 의존하지만, 이러한 커뮤니티는 웹사이트에서 검색을 거쳐야 하며 소비자가 기대하는 간편하고 빠른 응답을 제공하지 못하는 경우가 있다. 또한 일부 커뮤니티는 사용자의 문제를 해결하여도 받는 이득이 없다고 주장한다.

반면, 클로즈드 소스의 가장 큰 장점은 서비스 및 지원일 것이다. 이 지원에는 제품 및 서비스에 정통한 전문가와 즉각적인 지원을 위한 사용자 매뉴얼 및 연락 담당자를 포함한다. 지속적인 기술 지원은 기술 배경 지식이 거의 없는 사용자에게 중요한 판매 포인트이며 사용자가 오픈 소스 소프트웨어에 비해 폐쇄 소스를 선택하는 주된 이유 중 하나이다.

3. 혁신

오픈 소스는 매우 높은 자유도와 유연성을 자랑한다. 아무나, 제한 없이, 소프트웨어의 버그를 고치고, 개선하고 기능을 추가하는 등 자유롭게 소프트웨어를 변경 할 수 있는 덕분에 무궁무진한 발전과 혁신의 가능성이 있다. 하지만 이 혁신은 모든 사용자에게 전달 되지 않으며, 원본 소스 코드의 사용자 지정 변경이 해당 소프트웨어의 향후 지원 및 성장을 제한 하는지 논란이 있다.

오픈 소스의 높은 자유도와 비교 할 때 클로즈드 소스 소프트웨어는 소스 코드를 보거나 변경할 수 없다는 단점이 있다. 하지만 오픈 소스와 달리 클로즈드 소프트웨어는 정기적으로 새로운 제품과 업그레이드를 제공하기 위해 더 많은 양의 R & D를 유치한다.

클로즈드 소스는 포럼과 설문 조사를 통해 아이디어와

전략을 공유하고 혁신을 촉진하며 제품을 변화하는 요구에 적응할 수 있도록 하는 온라인 커뮤니티를 제공하며 이는 오픈소스 또한 그렇다.

4. 사용성

일반적으로 오픈소스는 사용성을 전문적으로 검수하는 사람이 없고 일반 사용자보다는 개발자에게 적합하도록 개발 되었기에 오픈 소스는 '불친절하다'는 비판을 받아왔다. 또한 사용자 가이드는 법으로 요구되지 않으므로 가이드 자체가 없기도 하며, 매뉴얼이 작성되었더라도, 일반 사용자가 이해하기 어려운 전문 용어로 가득 차 있는 경우도 많다.

클로즈드 소스의 경우, 해당 소프트웨어의 사용성은 판매율에 굉장히 크게 기여하기 때문에 예상 고객을 대상으로 한 전문 사용성 테스트를 거친다. 그렇기에 클로즈드 소프트웨어는 오픈 소스에 비해 비교적으로 사용성이 좋으며, 사용자가 소프트웨어를 금방 익힐 수 있도록 도와주는 사용자 설명서, 소프트웨어 사용 효과를 극대화하도록 도와주는 지원 서비스가 같이 제공되는 경우가 많다. 타사 시스템과 개발자는 클로즈드 소스 소프트웨어를 향상시키기 위해 다양한 메커니즘을 사용하기도 한다.

5. 보안

오픈 소스는 통제 된 환경이 아닌 누구나 자유롭게 드나들 수 있는 환경이기에 오픈 소스의 보안은 클로즈드 소스에 비해 취약하다. 공개된 소스를 통해 해당 소프트웨어의 약점 및 취약점이 발견 되고, 오픈 소스는 항상 보안 검사를 받는 것이 아니기 때문에 악의적인 사용자라면 백도어 트로이 목마 등 악성 코드를 소프트웨어에 포함 시킬 수도 있다.

이러한 잠재 된 위험을 줄이는 한 가지 방법은 강력한 온라인 커뮤니티가 지원하며 평판이 좋은 오픈 소스를 채택하는 것이다.

클로즈드 소프트웨어는 일반적으로 공통된 방향으로, 집중된 팀에 의해, 통제 된 환경에서 개발되기 때문에 보다 안전하다고 판단된다. 이 팀은 소스 코드를 보거나 편집 할 수 있는 유일한 사람들이며, 꼼꼼하게 감시하여 백도어 트로이 목마 또는 버그의 위험이 오픈 소스에 비해 적다.

4. 유명 오픈 소스 소프트웨어

리눅스(Linux)

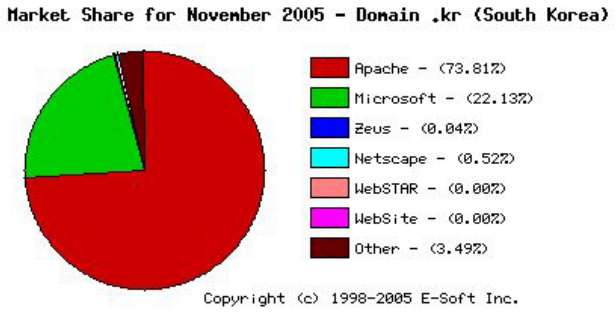
리눅스는 컴퓨터 운영 체제의 하나이며, 그 커널을 뜻하기도 한다. 리눅스는 자유 소프트웨어와 오픈 소스 개발의 가장 유명한 표본으로 들 수 있다.

리눅스는 운영 체제로서의 인기가 늘어가고 있는데, 지지자와 분석자들은 이와 같은 성공을 벤더 독립성과 적은 개발비, 보안성과 안전성에서 기인한다고 분석한다. 또한 모바일 운영체제인 안드로이드를 비롯하여, 삼성전자가 주도하는 타이젠(Tizen), 그리고 최근 새로운 대안으로 떠오르는 양대 산맥인 우분투(Ubuntu)와 모질라의 파이어폭스(Firefox) 운영체제도 모두 리눅스를 조상으로 하여 파생된 것이다.

아파치(Apache)

아파치 HTTP 서버(영어: Apache HTTP Server)는 Apache License 2.0의 따라 배포되는 무료 오픈 소스 크로스 플랫폼 웹 서버 소프트웨어이며 Apache Software Foundation의 지원 하에 열린 개발자 커뮤니티에 의해 개발되고 유지 관리된다. 아파치는 World Wide Web의 초기 성장에 핵심적인 역할을 수행했으며 2009년, 최초로 1 억 개

이상을 지원한 웹서버 소프트웨어가 되었다. 2016 년 7 월 기준으로 모든 활동 중인 웹 사이트 서버 중 46 %를 아파치가 차지한다. 우리나라에선 2005년 기준, 74%의 웹사이트의 서버를 아파치가 차지한다.



MySQL

MySQL은 세계에서 가장 많이 쓰이는 오픈 소스의 관계형 데이터베이스 관리 시스템(RDBMS)이다. 다중 스레드, 다중 사용자 형식의 구조질의어 형식의 데이터베이스 관리 시스템으로서 MySQL AB가 관리 및 지원하고 있으며, 이중 라이선스가 적용된다. 하나의 옵션은 GPL이며, GPL 이외의 라이선스로 적용시키려는 경우 전통적인 지적재산권 라이선스의 적용을 받는다.

이클립스(Eclipse)

이클립스는 다양한 플랫폼에서 쓸 수 있으며, 자바를

비롯한 다양한 언어를 지원하는 프로그래밍 통합 개발 환경을 목적으로 시작하였으나, 현재는 범용 응용 소프트웨어 플랫폼으로 진화하였다.

자바로 작성되어 있으며, 자유 소프트웨어이지만 막강한 기능을 자랑한다. 원래 IBM의 웹스피어 스튜디오 애플리케이션 디밸로퍼(WebSpheare Studio Application Developer)란 이름으로 개발되었던 것인데, 엔진부분을 오픈소스로 공개한 것을 기반으로 지금의 이클립스로 발전해 왔다.

XpressEngine

XpressEngine(eXpress+press+Engine, 구 제로보드 XE)은 고영수가 여러 자원 봉사자들과 함께 개발한 LGPL 기반 오픈 프로젝트로, 완전히 새로 개발한 웹 프레임워크이다. BBS, 블로그, 쇼핑몰, 위키 등 웹 사이트에 필요한 모든 것을 모듈로 구현해, 종합적인 웹 빌더로 사용할 수 있는 프레임워크를 목표로 개발이 진행 중이다. 이전 명칭은 '제로보드 XE'였으나, 정식으로 CMS 기능을 갖춘 1.1.0 버전 안내를 공지하면서 '보드'의 개념과 상이하다며 명칭을 변경하였다. 현재는 네이버 산하 오픈소스 프로젝트로 개발이 진행되고 있다.

출처

1. 오픈소스의 정의

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

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

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/%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/%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

http://korea.gnu.org/documents/copyleft/osd-korean.html

https://opensource.org/licenses/

https://olis.or.kr/index.do

2. 오픈소스의 역사

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

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

http://platum.kr/archives/55437

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

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

3. 오픈소스 vs 클로즈드 소스

https://ko.wikipedia.org/wiki/사유_소프트웨어

https://www.coredna.com/blogs/comparing-open-closed-

source-software

http://elliottyoung.co.uk/2014/01/06/the-pros-and-cons-of-closed-source-software/

4. 유명 오픈소스 SW들

http://www.xpressengine.com/

https://www.eclipse.org/org/

https://httpd.apache.org/