History of Open-Source

김재현 석형원 안다영 이소원 이현준

목차

오픈소스란?

컴퓨터에 관심이 있고, 컴퓨터를 전공하는 사람이라면 '코드'라는 명칭을 질리도록 들었을 것이다. 소스라는 것은 영어로 source, 정보의 출처를 의미한다.

소스는 우리가 보고서를 쓸 때 참고하는 서적, 블로그 그 모든 것이 소스가 될 수 있다. 오픈소스라는 것은 이러한 출처들을 공개함을 뜻한다. 정리하자면, 어떤 제품을 개발하는 과정에 개발하는 과정에 필요한 소스 코드나 설계도를 누구에게나 인터넷을 통하여 무상으로 공개한다.

오픈소스의 배포 조건은 다음과 같다.

1. 무료로 재배포 해야 한다.

2. 모든 설계도는 소스코드를 제공해야한다.

3. 파생된 제품들은 원 소프트웨어의 라이선스와 동일한 조건으로 배포 할 수 있어야한다.

4. 소스 코드와 함께 배치파일의 배포가 허용되는 경우 소스코드가 수정된 형태로 배포되는 것을 제한 할 수 있다.

5. 개인이나 단체에 대한 차별을 금지한다.

6. 노력의 분야에 대한 차별을 금지한다.

7. 프로그램에 첨부 된 권리는 똑같이 적용되기에 따로 라이센스를 발급 할 필요가 없다.

8. 라이센스가 특정 소프트웨어에만 한해서는 안된다. 모든 파생된 소프트웨어는 원본 소프트웨어의 라이센스를 계승받아야 하기 때문이다.

9. 라이센스로 다른 소프트웨어를 제한해서는 안된다.

10. 라이센스는 기술이기에 중립적이어야 한다.

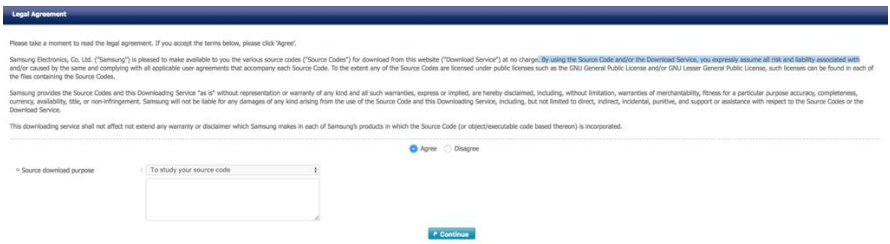
저작권을 의미하는 copyright와 반대되는 개념인 copyleft의 로고

오픈소스의 핵심 개념인 카피레프트는 다수가 정보를 공유해야 한다는 것이 요지이다.

여기서 주목해야 할 점은 무상으로 공개한다는 것과 수정이 가능하다는 것이다. 하지만 유의해야 할 점이 있다면 소수의 사람들에게만 오픈되어 있고 재배포가 불가능한 소스는 오픈소스라고 부르지 않는다.

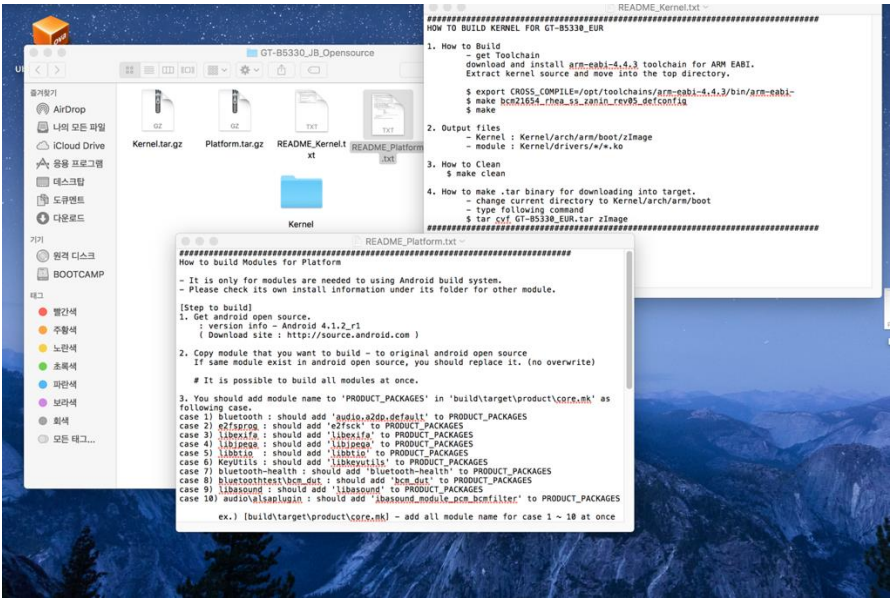
이렇게 오픈소스를 이용해 만든 프로그램으로는 우리가 너무나도 잘 알고 있는 리눅스, 파이썬, 이클립스, 자바 등이 있다. 프로그램 뿐만이 아니라 삼성이나 구글과 같은 대기업들 또한 자신들의 프로젝트를 오픈소스하여 다양한 시도를 하고 있다. 실제로 삼성은 오픈소스 사이트를 개설하였고 일반인들이 쉽게 접근할 수 있도록 이를 공개하였다.

그리고 오픈소스 코드를 다운받고자 한다면 아래와 같은 동의서가 뜨는데,



이는 GPL에 의해 자유는 보장되지만, 이 소스코드를 다운받음으로써

발생하는 문제는 책임지지 않는다는 법적 고지이다. 이에 대해 동의를 해야만 소스코드를 다운 받을 수 있다.



동의를 한다면 위와 같이 소스파일을 받을 수 있다.

오픈소스가 가능한 것은 소프트웨어 분야만이 아니다. 오픈소스 하드웨어는 기계, 장치와 같은 물리적인 것들과 같은 유형의 인공물에 대한 용어로서 누구나 그러한 것들을 제작, 수정, 배포 및 사용할 수 있는 방식으로 정의 내릴 수 있다.

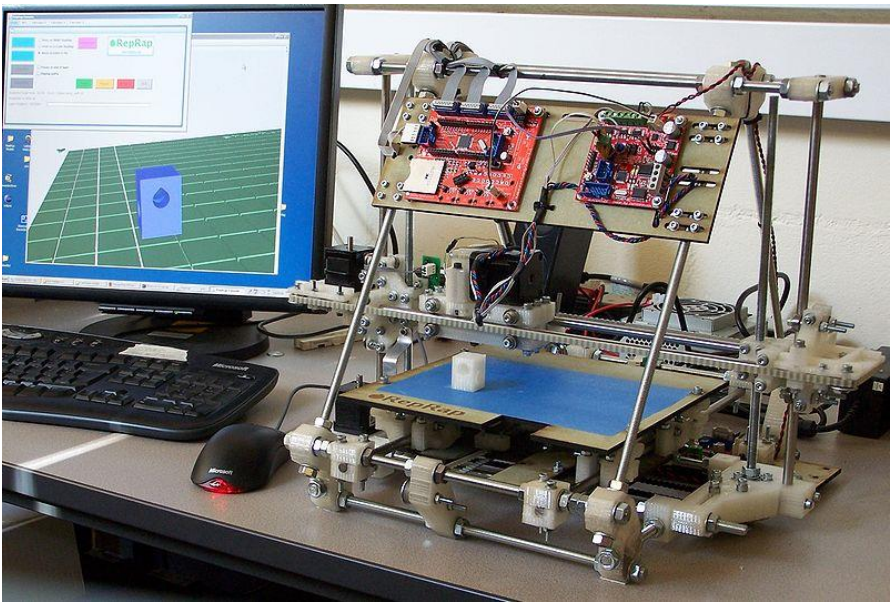
소프트웨어의 오픈소스와의 다른점이 있다면, 물리적인 자원을 물리적 상품으로 변환해야 한다는 점에서 다르다. 따라서 오픈소스 하드웨어의 라이센스하에 제품을 생산하는 개인이나 회사는 자신들이 생산한 제품이 원래 설계자의 제재를 받지 않았음을 명시해 줄 의무가 있다. 또한 원래 상품의 상표는 똑같이 따다 쓸 수는 없다.

이러한 특징을 가진 오픈소스 하드웨어의 대표적인 예시로는 아두이노가 있다.

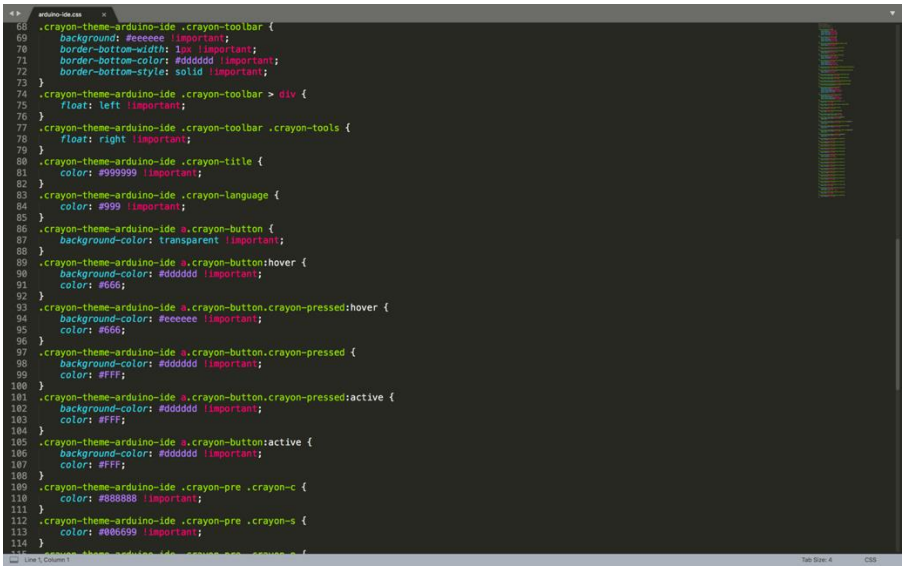
보드(상품)을 통제하기 위한 개발 도구 및 환경을 일컫는 단어이다. 아두이노는 처음에 AVR(마이크로 컨트롤러)를 기반으로 만들어졌다.

다수의 스위치나 센서로부터 값을 받아들여 외부 전자 장치들을 통제함으로써 환경과 상호작용이 가능한 물건을 만들어 낼 수 있다. 아두이노 또한 오픈소스 하드웨어이므로 오픈소스 소프트웨어와 같은 기준을 충족시켜야만 한다.

오픈소스 소프트웨어가 그러하듯, 오픈소스 하드웨어 또한 기업들 사이에서 공개하고 있다.



아두이노를 활용하여 개발되는 제품들은 셀 수 없을 만큼 많다. 그 중에서 가장 대표적인 제품은 3D 프린터가 있다.



이와 같은 제품은 소스코드와 하드웨어의 합작품으로, 오픈소스의 산물이다.

위와 같은 결과물이 오픈소스 코드를 이용하여 소프트웨어와 하드웨어를 만들어낸 사례이다.

공유란?

오픈소스가 공유된 소스라는 뜻이니 만큼, 공유라는 개념에 대해서 확실히 짚고 넘어가야한다.

공유란 두 사람 이상이 한 물건을 공동으로 소유함을 뜻하는 단어이다. 오픈소스에서의 공유는 GNU 공용 라이센스를 빼놓고 논의할 수 없다. 자유 소프트웨어 재단에서 만든 자유 소프트웨어 라이선스로, GPL과 일맥상통하는 단어이다.

GPL은 General Public License의 약자로, 가장 널리 알려진 카피레프트 사용허가이다. 또한 이 허가를 가진 프로그램을 사용하여 또 다른 프로그램을 만들면 그 신생 프로그램 역시 같은 카피레프트를 가져야 한다.

카피레프트가 저작권을 기반으로 한 정보의 공유를 위한 조치라는 것을 고려했을 때, 위의 문장을 쉽게 풀어쓴다면 신생 프로그램들도 모두 공유를 전제로 생성된다는 것이다. 출시되는 SW는 무료이며 아무도 이 자유를 빼앗을 수 없고 이를 기반으로 만든 SW를 다시 파는 것도 허용한다는 특성을 갖고 있다.

소프트웨어가 개발되기 전에도 정보 공유(또는 이와 유사한 행위)는 이루어졌다. 예를 들어 이웃끼리 김치를 맛있게 담그거나 수정과를 맛있게 만드는 방법을 이야기 했다면, 그것 또한 정보 공유의 범주에 속하는 것이다.

좀 더 구체적인 예시를 들어보자면, 자동차 개발 초기에 한 기업은 2 사이클 가솔린 엔진 특허권을 소유했었다. 이 특허를 통제함으로써, 그들은 업계를 독점하고 다른 자동차 제조 회사가 그들의 요구를 받아들이도록 강요할 수 있었고, 소송까지 가능했다. 그러던 중 타 자동차 회사가 포드의 특허권에 대해 도전장을 내밀었고, 결국 포드의 특허는 쓸모가 없게 되었다. 이때 자동차 제조업 협회가 형성되었고 새로운 협회는 미국 자동차 제조업자들간에 교차 라이센스 계약을 제정했다. 각 회사가 기술을 개발하고 특허를 출원할 지라도 이러한 특허는 공개 되었다. 이 특허를 다른 제조업체가 사용할 때 돈은 오가지 않았다. 이것이 소프트웨어가 개발 되기 전의 정보 공유였다.

그러나 소프트웨어가 개발되고 나서 부터 정보 공유의 형태는 달라지기 시작했다.

그 소프트웨어의 정보공유와 밀접한 관련을 갖는 것이 GPL이라는 프로젝트이다. GPL은 리처드 스톨만이 시작한 시작한 프로젝트였다. SW 개발초기에는 해커나 연구자들이 한 데 모여서 상호협력을 통해 소프트웨어를 개발했었다. 그러나

갈수록 개발환경이 양호해지자, 영리 목적으로 소스코드를 공유하지 않는 사람이 늘어나기 시작했고, 이 현상을 아니꼽게 보던 리처드 스톨만이 과거로 돌아가고자 시작한 프로젝트였다.

스톨만이 주장한 것은 소프트웨어의 상업화였고, 1984년에 출시된 유닉스와 비슷하면서, 더욱 효율적이며 상호협력을 할 수 있는 또다른 운영체제를 만들고자 하였다. 이듬해인 1985년, 그는 자유소프트웨어 재단을 조직하면서, 소프트웨어는 공유돼야 하며 프로그래머는 소프트웨어로 돈을 벌어서는 안된다는 내용의 GNU 선언문을 제정하기도 했고, 이를 지원하기 위해 저작권에 대응하는 카피레프트 운동도 주창하였다.

그러나 이 GNU 프로젝트는 독립된 운영체제에 필요한 커널의 부재로 큰 성공을 거두지는 못했는데, 이는 1991년 리누스 토발즈가 리눅스를 개발함으로써 문제가 해결되었다.

과거의 소프트웨어

1950년대와 1960년대엔 대부분의 소프트웨어가 공동으로 일하는 학자 및 기업 연구원에 의해 개발되었다. 그리고 개발된 소프트웨어는 주로 공용 소프트웨어(Public Domain Software)의 형태로 공유되었는데, 이는 학계에서 오랫동안 확립되어온 개방과 협력의 원칙에 따라 배포된 것이었다. 이렇듯 소프트웨어를 상품으로 여기는 경향은 존재하지 않았고, 오히려 소스코드가 소프트웨어와 함께 공유되는 것이 일반적이었다. 그 당시에는 어떤 하드웨어 또는 운영체제를 사용하는지에 따라 소프트웨어가 작동할 수도, 하지 않을 수도 있었는데, 소스코드를 소프트웨어와 함께 공유하면 소비자들이 자신의 필요에 따라 소프트웨어를 수정할 수가 있었기 때문이다. 때로는 사용자들이 버그를 수정하기도 했고 새로운 기능을 추가하기도 했기 때문에 제작자와 사용자 모두에게 이득이 되는 일이었다.

1960년대 후반에는 운영체제와 컴파일러가 발전하면서 소프트웨어의 개발 비용이 하드웨어에 비해 크게 증가했다. 점차 성장하는 소프트웨어 산업은 하드웨어 제조업체의 번들 소프트웨어 제품들과 경쟁하고 있었는데, (당시 번들 소프트웨어 제품의 비용은 하드웨어 비용에 포함되었다.) 소프트웨어와 관련해 더 많은 지원이 필요하면서 이에 대한 비용을 지불할 여건이 되는 소비자들은 하드웨어 제조업체들의 번들 소프트웨어 제품 비용이 하드웨어 비용에 포함되는 것을 원하지 않았다. 그리고 1969년 미국 정부가 IBM과의

반독점 소송에서 번들 소프트웨어가 반경쟁적이라고 기소하는 사건이 있으면서, 제한적인 라이선스 하에 판매되는 소프트웨어들이 점차 증가했다. 물론 일부 소프트웨어들은 여전히 무료로 제공되었다.

당시 자유 소프트웨어의 하락세를 보여주는 예로 UNIX를 들 수 있다. 1970년대 초, AT&T는 정부와 학술 연구자에게 Unix의 초기 버전을 무료로 배포하였다. 하지만 이 버전은 수정된 버전을 배포할 수 있는 권한이 없었기 때문에 애초에 자유 소프트웨어라고 할 수도 없었고, Unix가 널리 보급 된 1980년대 초반에 AT&T가 무료 배포를 중단하고 시스템 패치를 진행하면서 대부분의 연구원들은 상용 라이선스를 지불해야 했다.

1970년대 후반과 1980년대 초반에 컴퓨터 판매 회사와 소프트웨어 개발사들이 본격적으로 소프트웨어 라이센스를 판매하기 시작했다. 또한 저작권, 상표 등을 이용해 소프트웨어의 활용을 법적으로 제한하기 시작하면서 소프트웨어는 '상품'이 되었다. Microsoft의 설립자인 빌 게이츠가 1976년에 쓴 'Open letter to Hobbyists' 라는 제목의 에세이는 애호가(Hobbyists)들이 라이선스 비용을 지불하지 않고 자사의 소프트웨어를 공유하는 것에 대한 불만을 표현하는 내용으로, 당시 Microsoft가 소프트웨어를 '상품'으로 여기고 있었음을 보여준다. 물론 다른 기업들도 그랬다. AT&T는 수익을 얻기 위해 Unix에 라이선스를 적용했고, 1983년에 IBM은 더 이상 소스코드를 배포하지 않겠다는 정책을 발표했다. 자유 소프트웨어는 점점 가라앉고 있었다.

이러한 기업들의 제한에도 불구하고 ‘hobbyist’ 또는 ‘hackers’ 라고 불리는 사람들은 여전히 소스코드를 다른 사람들과 무료로 공유하고자 했다. 이들은 공유를 위한 그룹들을 설립하기도 했는데, 그 중 하나인 ‘SHARE’ 라는 그룹은 1955년에 설립되었으며 주로 자유 소프트웨어를 수집하고 배포하는 활동을 했다. 이 그룹이 최초로 배포를 시작한 시기는 1955년 10월로, 소프트웨어와 정보를 담은 자기 테이프를 배포하였다. 인터넷이 널리 보급되고 사용되기 이전이었기 때문에 배포를 위해선 자기 테이프, 잡지, 프로그래밍 서적과 같은 오프라인 매체를 이용해야 했다.

자유 소프트웨어 운동이 진행되던 시기인 1980년대엔 소프트웨어가 소스코드와 함께 BBS(게시판 시스템) 네트워크에서 공유되고 있었다. BASIC 및 기타 언어로 작성된 소프트웨어는 소스코드로만 배포할 수 있었는데, 사용자가 그 소스코드를 수집하고 수정에 대해 논의하기 위해 BBS 네트워크를 이용한 것은 사실상 오픈 소스 라고도 할 수 있다. 1980년대 초에 Usenet과 UUCPNet과 같은 일종의 커뮤니티의 출현으로 프로그래밍 공동체는 더욱 커졌고 프로그래머들에겐 자신의 소프트웨어를 공유하고 다른 사람들의 소프트웨어 개발에 기여할 수 있는 더욱 쉬운 길이 열렸다.

자유 소프트웨어

1979년 제록스사는 최초의 레이저 프린터 중 하나를 MIT에 있는 인공지능 연구소에 기증하였다. 그 프린터는 자주 고장을 일으켰고 인공지능연구소의 한 해커는 제록스에게 프린터를 제어할 수 있는 코드를 제공해 줄 것을 요구했다. 프로그램을 수정해서 인쇄종이가 프린터에 끼어 중단되는 것을 사용자에게 경고 메시지를 보여줄 수 있도록 만들려 하였다. 이렇게 된다면 프린터는 빠르게 복구될 수 있을 것이다. 이런 작업을 하기 위하여 해커에게는 프린터 프로그램의 소스 코드가 필요했다. 다른 사람의 코드를 빌리고 수정하기도 하면서 함께 연구하는 학구적인 분위기의 인공지능연구소에서 일하는 프로그래머인 그에게 이러한 요청은 별로 유별난 것이 아니었다. 게다가 이전에 제록스사는 같은 문제가 있었던 프린터의 소스코드를 그에게 준 적이 있기도 하였다. 그러나 이번에 제록스사는 이 요청을 거부했다. 담당자는 기밀 유지 협약에 서명하였고 회사는 소스코드의 판권을 독점했다는 것이었다. 해커는 소유권이 프로그램을 더 좋게 향상시키려는 그를 가로막고 있다는 사실에 화가 났다. 그는 제록스사가 프로그램을 독점했고 황금률을 위반했다고 말했다. 그러나 제록스는 혼자가 아니었다. 소프트웨어 산업이 점점 커지면서 실리콘 밸리는 수많은 인공지능연구소의 인재들을 빼가기 시작했고 이 프로그래머들은 소프트웨어 회사에서 일할 때 기밀 유지 협약에 서명하도록 강요받았다. 이제 코드는 저작권이 생기며 더 이상 공유될 수도 없고 향

상될 수도 없는 것이라는 사실을 그 해커는 알게 되었다. 그리고 저작권이 프로그래밍 사회를 파괴하고 있다고 결론지었다. 이 해커는 바로 처음으로 자유소프트웨어 운동을 시작한 리처드 스톨만이다.

"소프트웨어를 판매하는 사람들은 사용자를 각각 구분하고, 그들 위에 군림하고, 사용자 서로가 프로그램을 공유하는 것을 막고자 한다. 나는 이런 식으로 사용자 간의 결속이 깨지는 것을 거부한다. 나는 올바른 양심으로 비공개 협정이나 소프트웨어 사용권 계약에 서명할 수 없다. 여러 해 동안 인공지능 연구소에서 일하면서 이러한 경향과 다른 박정한 일들에 저항해 보았지만 결국 그들의 승리로 끝나고 말았다. 내 의지에 역행하는 이런 일들이 일어나는 연구소에 나는 더 이상 머무를 수가 없었다." (GNU Manifesto)

1984년 리처드 스톨만은 자유 소프트웨어 재단을 창설했다. 이 재단은 소프트웨어의 본래 생산 유통 방식인 정보 공유의 방식을 복원하기 위해 자유 소프트웨어 운동을 시작하였다. 이는 소스 코드 공개를 통해 누구나 소프트웨어를 수정할 수 있게 하며, 자유로운 복제와 배포를 허용하는 것이다. 또한 이 운동의 목표는 운영 체제만이 아닌 모든 소프트웨어를 자유소프트웨어로 만드는 것이며 이 프로젝트의 핵심 작업은 운영 체제를 만들어 여러 사람들의 손을 거쳐 더 완성도 높은 소프트웨어를 만드는 것이다. 이를 위한 선결

과제는 컴퓨터 네트워크에 이용되는 유닉스의 소스 코드를 사용하지 않으면서 유닉스와 같은 향상된 운영체제를 개발하는 것이었다. 유닉스는 1969년 벨연구소의 두 연구원에 의해서 개발되어 IBM이나 Compaq, 썬 등 12개의 다른 버전으로 사용되고 있는 운영체제의 한 종류로 이 유닉스라는 운영체제는 대형 컴퓨터에서 주로 사용되었으며 가격도 매우 비쌌다. 아무리 싸도 자동차 한 대 값은 되었으며 보통 집 한 채 가격에 해당하는 비싼 가격으로 팔려 학생이나 가난한 사람들이 유닉스를 정식으로 구입해서 사용하는 일은 불가능한 일이었다. 그러나 당시 모든 컴퓨터들은 유닉스를 기반으로 돌아가고 있었으며 학교에서는 유닉스를 기본적인 운영체제로 교육 시키고 있었다. 스톨만은 Unix에 대항해 GNU 프로젝트를 통하여 유닉스 계열 컴퓨터의 새로운 운영체제를 개발하여 GNU 라고 명명했는데 이는 'GNU is Not Unix'의 준말이다. GNU의 도전은 대단한 것이었다. 운영체제란 두 숫자를 더한다든지, 정보를 하드디스크에 옮긴다든지 하는 일에 필요한 프로그램들과 그러한 일을 직접적으로 하드웨어에 전달하는 방식을 말한다. 그러나 윈도우를 운영한다거나 프린터나 기타 장치들과 통신하는 등의 특정한 작업을 수행하는 수많은 보조 프로그램들이 없다면 그 운영체제는 무용지물이 되고 만다. 효과적인 시스템을 구축하기 위해서 GNU 프로젝트는 이러한 모든 프로그램들을 만들어 내야 했다. 페렌즈는 "이것은 창고에서 비행기를 만드는 일에 비유될 만한 것이다"고 말한다. 사람들은 이것이 불가능하다고 생각했다. 그러나 이 일을 스톨만은 해내었고 스톨만 보다 덜 뛰어난 사람이 담당했었더라면 아마 불가능했었을 것으로 생각된다. GNU 프로젝트는 자유 소

프트웨어 운동에 대해 사용자가 자유 소프트웨어를 이용하여 자유롭게 컴퓨터를 사용할 수 있도록 하는 것이 목적이라고 설명하며 자유 소프트웨어를 사용하면, 사용자 스스로가 컴퓨터를 사용하는 환경을 구성할 수 있으며 독점 자유 소프트웨어는 소프트웨어 개발자에 의존해야 할 수밖에 없음을 강조한다. 자유 소프트웨어 운동은 소프트웨어를 협업적인 방식으로 공동으로 개발하는 것을 넘어서 더 나은 사회를 만들기 위한 도덕적 활동이다. 공유와 협동이라는 사회적 결속을 증가시키기 때문에 자유 소프트웨어 운동이 사회 전체를 위한 본질적 가치라고 설명한다. 따라서 자유 소프트웨어의 4가지 요건에 대한 다음과 같이 설명한다.

*** 프로그램을 원하는 어떠한 목적으로도 실행할 수 있는 자유**

*** 프로그램이 어떻게 동작하는지 학습하고, 자신의 필요에 맞게 개작할 수 있는 자유. 이것을 위해서는 소스 코드에 대한 접근이 전제되어야 한다.**

*** 이웃을 도울 수 있도록 복제물을 재배포할 수 있는 자유.**

*** 프로그램을 개선시킬 수 있는 자유와 개선된 이점을 공동체 전체가 누릴 수 있게 그것을 발표할 자유. 이를 위해서는 역시 소스 코드에 대한 접근이 전제되어야 한다.**

리처드 스톨만은 자유 소프트웨어 운동에 대해 정식화하고 인증하기 위해 GNU 일반 공중 사용허가서와 카피레프트 개념을 제시하였다. 일반 공중 사용허가서는 모든 사람에게 프로그램을 실행하고 복사하고 수정하고 수정된 버전을 배포할 수 있는 권리는 부여하지만 자신만의 규제를 더할 수 있는 권리는 부여하지 않는다고 설명한다. 스톨만의 모든 코드는 카피레프트 되었다. 다른 사람들이 또 다른 사람들로 하여금 그들의 수정에 똑같은 수정을 가할 수 있도록 한다면, 사용자들은 누구나 자유롭게 소프트웨어를 바꿀 수 있었다. 스톨만은 카피레프트가 그들 자신의 도구를 이용하여 소프트웨어 독점자들에게 대항하는 방법이라고 말한다. GNU 운영체제를 위한 텍스트 편집기와 컴파일러를 직접 작성했다. 하지만 운영체제의 중심 모듈로서 소프트웨어 프로그램의 요청을 처리하고 요청을 컴퓨터 CPU가 이해할 수 있는 명령으로 변환하는 역할을 하는 커널(Kernel:운영체제의 핵심적인 부분)을 완성하는데 많은 어려움을 겪었다. 범용으로 사용 가능한 여러 가지 프로그램들을 만들어 내긴 했지만 GNU 운영체제의 커널을 만들어내지 못했다. 그 이유 중의 하나는 스톨만이 유닉스의 커널을 복제하지 않고 GNU 시스템의 기초를 카네기 멜런대학에서 개발한 진보적이고 경험적인 커널로 선택한 것이다. 스톨만은 새로운 커널을 개발한 몇 안 되는 사람 중의 한명이었고 이 일을 할 수 있다고 생각하는 거의 유일한 사람이었다. 그러나 혼자서 많은 코드를 타이핑하는 것이 애초부터 무리였고 결국 포기하게 되었다. 몇 년간의 고통은 키보드와 씨름하기 힘들게 만들었고 커널을 개발하기 위한 작업은 중단되었다. 스톨만은 MIT 학생들을 고용해서 계속해 나가려고 노력했지만

얼마안가 기계적으로 컴퓨터 코드를 번역하는데 염증을 느껴 그만두게 되었다.

1991년 헬싱키 대학의 21살 대학생 토발즈는 결코 뛰어난 프로그래머가 아니었다. 토발즈는 원래 앤드루 스튜어트 타넨바움 교수가 운영 체제 디자인을 가르치기 위해 만든 교육용 유닉스인 미닉스를 사용하고 있었는데, 타넨바움이 미닉스를 다른 사람이 함부로 개조하지 못하도록 제한을 두자, 미닉스의 기능에 만족하지 못했던 토발즈는 새로운 운영 체제를 개발하고자 했다. 리눅스는 본래 운영 체제 위에서 실행되는 터미널 에뮬레이터였다. 초기에는 시리얼 포트를 이용한 간단한 신호를 주고 받는 작업밖에 할 수 없었지만, 토발즈는 디스크의 파일도 읽고, 쓰게 하고 싶었다. 이처럼 완전한 파일 제어가 가능해지자, 토발즈는 이것을 포직스(POSIX)에 호환되는 운영 체제 커널로 발전시키기로 마음먹고 이를 기반으로 리눅스를 개발하기 시작하였다. 이것은 유닉스의 유연함과 안정성을 가졌고, 성능이 그리 좋지 않은 기종의 컴퓨터에서도 아주 유용했다. 그는 그의 운영체제를 Freax 라고 불렀다. 그 이름이 세련되지 못하다고 생각한 그의 친구들이 그 운영체제의 이름을 바꿔 부르기를 Linux라 하였다. 초기의 리눅스는 이식성이 고려되지 않은, 다만 i386계열에서 운영되는 유닉스 호환 운영체제를 목표로 하는 프로젝트였다. 초기 버전 0.01은 가장 기본적인 커널 만을 포함하고 있었으며, 실행조차 되지 않는 수준이었다. 얼마 후 리눅스 공식 버전인 0.02가 발표되었는데, bash와 gcc정도가 실행될 수 있는 수준이었다. 차차 리눅스가 버전 업이 되면서 GNU커널로 개발 중이던 Hurd의 개발이 순조롭지 않았던

리처드 스톨만은 유닉스 커널과 호환 가능한 커널인 리눅스를 GNU시스템의 커널로 채택하기로 하였다. 리눅스는 강력한 GNU C 컴파일러인 gcc로 컴파일된 많은 응용프로그램들을 가지게 되었고, 둘의 결합으로 GNU시스템은 완전한 구조를 갖추게 되었다. 그러나 이들을 결합시킨다는 것은 말처럼 간단한 작업이 아니었다. 설치와 함께 즉시 사용할 수 있는 배포 본으로서 시스템을 구축하는 것은 무척이나 방대한 작업이었기 때문이다. 완성된 전체 시스템을 구성하는 것이 처음이었기 때문에 GNU C 라이브러리를 리눅스 커널에 맞게 재작성해야 했으며, 이전에 다룰 수 없었던 설치와 부팅 시에 고려되어야 할 CPU의 번지 지정에 대한 기계어 차원의 문제들이 해결되어야 했다. 다양한 배포본의 구성을 위해서 노력했던 많은 사람들의 수고 덕분에, 결국 리눅스와 이에 맞게 수정된 GNU 시스템은 완성된 운영체제로서의 모습을 갖출 수 있게 되었다. 이것이 우리는 흔히 리눅스라고 호칭하는 시스템인 GNU/리눅스이다. 이 리눅스는 자유소프트웨어로써 독점 소프트웨어의 횡포를 견제할 수 있다는 하나의 가능성을 제공했다. 자유소프트웨어 운동의 그룹들은 독점 소프트웨어를 자유소프트웨어로 대체하기를 원했다. 그리고 이를 실행하기 위해 엄청난 노력을 하였다. 이들은 정보 공유정신에 입각한 소프트웨어를 통하여 이용자가 스스로 자신에 맞게 소프트웨어를 수정할 수 있는 권리가 있어야 하고 이러한 자유로운 소통을 통하여 독점적 소프트웨어를 제거하고 인위적으로 정보의 흐름을 막는 지재권에 대항해야 한다고 주장했다. 폐쇄적 개발 방식보다 열린 개발 방식이 훨씬 더 안정되고 우수한 소프트웨어를 만들 수 있다고 말해왔다. 이러한 그들의 주장들이 단순

한 이상적 몽상이 아니라는 것을 증명하기 위하여 또한 이러한 독점적 소프트웨어를 대체하기 위한 기술적 우월성을 증명하기 위하여. 자유 소프트웨어는 단순히 상업적 독점 소프트웨어에 대한 기술적 경쟁 자체가 중요하지는 않다. 하지만 어차피 사용을 위한 소프트웨어라면 기술적으로 상업적 독점 소프트웨어에 떨어지지 않는 기술 확보는 매우 중요한 문제이다. 이것을 리눅스는 해결하였고 자유소프트웨어 운동에 많은 기여를 하였다.

오픈소스 소프트웨어

1997년에 Eric S. Raymond가 해커 커뮤니티와 자유 소프트웨어 원칙에 대해 철저히 분석한 The Cathedral and the Bazaar를 발표했다. 이 자료는 1998년 초에 주목을 받았으며 Netscape Communications Corporation가

Netscape Communicator Internet suite를 무료 소프트웨어로 출시하는 일에 영감을 주었고 이 코드는 현재 Mozilla Firefox와 Mozilla Thunderbird의 기반이다. Netscape의 이러한 행동으로 인해 Raymond와 다른 사람들이 자유 소프트웨어의 원칙과 이점을 상용 소프트웨어에 가져와 쓸 수 있게 되었다. 그들은 FSP의 사회적 행동이 Netscape와 같은 회사들이 매력을 못 느낀다고 결론을 내렸고 소스 코드를 공유하는 것에 대한 잠재력을 강조하기 위해 자유 소프트웨어 운동을 다시 브랜드화시키는 방법을 모색했다.

1998 년 1 월 Netscape의 Navigator에 대한 소스 코드 발표에 대한 반응으로 캘리포니아 주 Palo Alto에서 열린 전략 세션에서 자유 소프트웨어 운동권 사람들(오픈 소스를 제안한 Christine Peterson , Todd Anderson, Larry Augustin, Jon Hall, Sam Ockman, Michael Tiemann, Eric S. Raymond) 이 오픈소스라고 이름을 정했다. 그 다음주에는 Eric S. Raymond와 다른 사람들이 오픈 소스라는 단어를 퍼뜨리기 시작했다.

이 용어는 기술 출판사 Tim O'Reilly가 1998 년 4 월에 조직 한 행사에서 큰 호응을 얻었다. 월내는 “프리웨어 서밋”이었지만 “오픈소스 서밋” 으로 바뀐 이 행사는 Linus Torvalds, Larry Wall, Brian Behlendorf, Eric Allman, Guido van을 비롯한 가장 중요한 자유 및 오픈 소스 프로젝트의 지도자들을 모았다. 이 모임에서 자유 소프트웨어라는 이름으로 인한 혼란을 제기했다. Tiemann은 "소스웨어"를 새로운 용어로, Raymond는 "오픈 소스"를 주장했다. 개발자들에게 표가 주어졌고 그 날 저녁 기자 회견에서 승자가 발표되었

다. 5일 후 Raymond는 새 용어의 채택을 위해 자유 소프트웨어 커뮤니티에 연락을 했다. 그 후 얼마 되지 않아 The Open Source Initiative(OSI)가 설립되었다.

오픈소스 운동이란 몇개의, 또는 모든 소프트웨어에 대해 오픈소스를 지원하도록 하는 운동을 말한다. 오픈소스 운동은 오픈소스 소프트웨어의 개념과 생각을 널리 퍼트렸다. 오픈소스 운동을 지지하는 프로그래머들은 소프트웨어 개발에 있어서 프로그램 코드를 자발적으로 쓰고 고침으로써 오픈소스 커뮤니티에 기여했다. 오픈소스는 이미 작성된 코드를 편집하는 것에 있어서 누구도 방해할 수 없다는 것 또는 누구에게나 작성된 코드를 차별없이 공유하는 것을 필요로 한다. 이러한 소프트웨어 개발에 대한 접근 방식으로 모두가 오픈소스 코드를 수정할 수 있게 된다. 이렇게 수정한 것은 오픈소스 커뮤니티의 개발자들에게 배포된다. 이러한 방식으로 인해 코드에 코드 수정에 참여한 모든 사람들의 개성이 나타나고 코드의 변화가 시시각각 작성된다. 이 방식은 코드의 소유권을 확실히 하기 어렵지만 오픈소스 운동의 개념을 따라간다.

오픈 소스 소프트웨어 운동의 기원은 1980년대 초반 소프트웨어의 상용화 및 그에 따른 여러 가지 제한에 반대하여 리차드 스톨만(Richard Stallman)이 자유소프트웨어 재단(Free Software Foundation)을 만들고 자유 소프트웨어 운동을 시작하면서부터이다. 그 이후 1990년대 들어서면서 인터넷과 더불어 리눅스가 성공하기 시작하였으며, 이를 토대로 상용 소프트웨어

업체들의 참여를 이끌기 위해 1998년 Open Source Initiative(OSI)가 만들어졌다. 때를 같이 하여 MS의 익스플로러에 밀려 어려움을 겪고 있던 넷스케이프사가 웹브라우저의 소스 코드를 공개하는 결정을 내리게 되었으며, Sun, IBM 등이 오픈 소스 소프트웨어에 대한 지원을 시작하였다. 최근 들어서는 아시아와 유럽을 중심으로 각국 정부에서 미국 중심의 소프트웨어 산업을 극복하고 자국의 소프트웨어 기술 수준 향상 및 산업을 성장시키기 위해 오픈 소스 소프트웨어에 대한 지원정책을 내놓고 있다.

어떤 제품을 개발하는 과정에 필요한 소스 코드나 설계도를 누구나 접근 해서 열람할 수 있도록 공개하는 것. 보통 소스가 공개된 소프트웨어를 오픈 소스 소프트웨어라고 하며, 소프트웨어 말고도 개발 과정이나 설계도가 공개되는 경우 하드웨어에도 오픈 소스 모델이 적용 가능하며, 글꼴과 같은 데이터에도 오픈 소스 개발 모델이 적용되는 경우가 있다. 단순히 소스를 공개만 하는 것이 아니라, 이를 2차 창작하는 것을 허용하기도 하고, 나아가 조건 없이 상업적 용도로도 사용할 수 있게 하는 경우가 있다.

오픈 소스 소프트웨어란 소스 코드가 공개(Open)된 프로그램이다. 대부분의 오픈 소스 소프트웨어는 무료로 사용 가능하기 때문에 프리웨어와 헷갈리는 경우가 많지만, 프리웨어는 무료로 사용 가능한 프로그램이고, 오픈 소스는 소스 코드가 공개된 프로그램이기 때문에 엄연히 다른 개념이다(예를 들어, 오픈 소스 소프트웨어를 돈 받고 파는 경우도 있다.). 자유 소프트웨어(Free Software)와 비슷하지만, 오픈 소스 소프트웨어가 자유 소프트웨어

보다 조금 더 상위 개념이다.

일반 사용자 입장에서는 프리웨어나 오픈 소스 소프트웨어나 단순히 공짜로 사용할 수 있다는 점에서는 비슷할 수 있지만, 소스코드를 보고 이해할 수 있고, 수정할 수 있는 개발자 입장에서는 크게 다르다. 예를 들어, 상용 또는 프리웨어 프로그램을 사용하는 사람들은 버그를 발견했다 하더라도 소스 코드를 모르니 수정할 수 없고, 사용자가 새로운 아이디어가 떠올랐다 해도 그것을 곧바로 프로그램에 적용시킬 수도 없다. 비교적 간단한 프로그램은 리버스 엔지니어링으로 어셈블러 수준에서 뜯어고칠 수는 있으나 코드가 공개된 것보다 몇 백 배는 어렵기도 하고, 저작권 같은 문제가 얽히고설키기에 하려는 사람은 없다고 보면 된다.

하지만 사용자가 프로그래밍 언어를 아는 경우 소스가 공개되어 있다면 본인이 직접 소프트웨어의 문제를 수정하거나 개선을 할 수 있게 되는 것이다. 또한, 개발하던 프리웨어가 개인적인 사정이나 회사의 사정에 따라 개발이 중지되면 그대로 사장되는 경우가 종종 있는데, 오픈 소스 소프트웨어는 소스가 공개되어 있기 때문에 다른 개발자/개발사에서 이를 이어 받아서 새로이 개선해 나가면서 개발하는 것이 된다. 그래서 개발자와 사용자가 일치하는 개발 및 시스템, 네트워크 분야에는 웬만한 클로즈드 소스 상용 소프트웨어는 명함도 못 내밀 정도로 고품질의 오픈 소스 소프트웨어가 넘쳐난다. 그러나 그러지 않는 분야에선 말 그대로 취미 수준에 머물러 있는 경우도 많다.

JAVA

1996년 첫 공개 이후 Java 런타임의 Java 소스코드의 부분이 무료로 다운 받을 수 있는 JDK에 포함되었지만 Java플랫폼은 오픈소스가 아니었다. Sun이 나중에 공개 소스 코드를 공개하여 Java Runtime Environment의 전체 소스 코드를 포함하는 별도의 프로그램을 통해 일반인에게 개발 되었고 Java의 컴파일러 javac 소스를 사용할 수 있게 되었다. Sun은 또한 JDK의 초기 버전을 Linux로 이식하거나 JDK를 개선한 자원 봉사자 보임인 Blackdown 자바 프로젝트에 사용할 수 있는 JDK소스 코드를 만들었다. 그러나 Sun의 허락 없이는 수정 및 재 배포가 금지 되었기때문에 이것들 모두가 오픈소스가 아니었다.

하지만 Java 플랫폼에 대한 몇가지 독립적인 부분의 재 구현이 이뤄졌으며 그 중 대부분은 GNU Compiler for Java (GCJ)와 같은 오픈소스 커뮤니티에 의해서 만들어졌다. Sun은 어느 오픈소스 클론 프로젝트에 대해 소송을 제기하지 않았다. GCJ는 Java 구현 시 GCJ를 출하 한 Fedora와 Ubuntu와 같은 배포판을 지원하는 무료 소프트웨어에서 Java에 대한 사용자 경험이 현저하게 떨어졌다. GCJ를 Sun JDK로 대체하는 방법을 사용자들이 자주 질문했다. 왜냐하면 GCJ는 불완전한 구현 이었기 때문에 호환되지 않고 버그가 있었기 때문이다.

2006 년 조나단 슈왈츠 (Jonathan I. Schwartz)는 썬 마이크로 시스템즈 (Sun Microsystems)의 CEO가 되어 오픈 소스에 대한 헌신을 표명했다. 2007 년 5 월 8 일, Sun Microsystems는 GNU General Public License에 의거하여 Java Development Kit을 OpenJDK로 발표했다. 클래스 라이브러리의 일부 (4 %)는 다른 업체의 라이센스로 인해 오픈 소스로 공개 될 수 없었고 바이너리 플러그로 포함되었다. 따라서 2007 년 6 월 Red Hat은 IcedTea를 출시하여 GNU Classpath 구현과 동등한 제약이 있는 구성 요소를 해결했다. 릴리스 이후, 대부분의 문제는 해결되어 오디오 엔진 코드와 색상 관리 시스템 만 남아 있다.

Git

Git이란 오픈 소스를 체계적으로 관리하기 위해서 소스 버전 관리 시스템을 이용하는 것이다. 오픈소스 뿐만이 아니라 기업에서도 자사의 솔루션을 체계적으로 관리하기 위해 소스 버전 관리 시스템을 이용한다. 이전에는 CVS를 많이 이용했고 요즘에는 SVN을 많이 이용한다. 그 외에도 많은 소스 버전 관리 시스템이 존재한다. 깃(Git)은 CVS, SVN과 같은 소스 버전 관리 시스템으로 오픈소스 OS인 리눅스의 소스를 관리하기 위해 리눅스 커널의 개발자이자 창시자인 리누스 토발즈에 의해서 만들어진 분산 소스 버전 관리 시스템이다. 리눅스의 창시자인 리누스 토발즈는 리눅스 커널을 압축파일과 패치로만 관리를 해왔다. 그러다가 체계적인 소스 관리를 위해 2002년에 상용 소스 버전 관리 시스템인 BitKeeper를 이용하게 된다. 하지만 2005년에 리누스 토발즈와 BitKeeper의 관계는 틀어지게 되었다. 오픈소스 OS인 리눅스의 커널 소스 관리를 상용 소스 버전 관리 시스템인 BitKeeper에 맡기다 보니 개념 및 의견 충돌이 지속적으로 나왔던 것이었다. 결국 리누스 토발즈는 BitKeeper의 사용 경험 및 교훈을 바탕으로 오픈 소스에 적합한 소스 버전 관리 시스템을 만들게 되는데 빠른 속도와 단순한 구조, 비선형적 개발(동시다발적으로 수천 개의 프로젝트를 관리할 수 있도록 branch 생성)과 완벽한 분산, 대규모 프로젝트에 적합한 규모 등을 목표로 깃을 만들게 된다.

깃허브는 깃을 사용하는 웹 기반의 호스팅 서비스다. 즉, 웹 기반의 분산 소

스 관리 시스템인데 그 소스 관리 시스템의 엔진을 깃을 사용한다는 것이다. 깃허브는 루비 온 레일스를 기반으로 만들어졌으며 영리적인 솔루션 개발을 위한 프로젝트나 오픈 소스 프로젝트에 상관없이 무료로 다 제공해주는 서비스이다. 일반적으로 소스 버전 관리 시스템은 설치해서 사용하게 되는데 기업에서 CVS나 SVN을 이용할 때에도 PC든 서버든 하드웨어에 윈도우나 리눅스를 설치하고 그 위에 CVS나 SVN 서버 프로그램을 설치해서 사용하게 된다. 그러다 보니 초기 구축하는데 비용이 들어가게 되고 유지하는데 비용도 들어가게 된다. 하지만 오픈 소스를 개발하는 커뮤니티 입장에서는 이런 관리 비용을 집행하기가 어려운 것이 사실이다. 깃허브는 앞서 깃의 목표 중의 하나인 비선형적 개발 기능을 바탕으로 무료로 손쉽게 웹 기반으로 제공해주기 때문에 수많은 오픈 소스가 깃허브를 통해 관리되기 시작하면서 오픈 소스 진영에서 가장 인기 있는 소스 버전 관리 호스팅 서비스로 자리 잡게 되었다. 특히 웹 기반으로 제공해주기 때문에 SVN이나 CVS처럼 따로 클라이언트를 설치해서 사용할 필요가 없이 웹 브라우저만 있어도 관리 및 소스 다운로드가 가능하다는 점 때문에 더 인기를 끌게 되었다. 이제 깃허브는 오픈 소스 프로젝트의 소스 관리 시스템의 대명사로 불리는 수준에 이르게 되었다. 깃허브가 다른 소스 버전 관리 시스템과 달리 인기를 끌 수 있었던 이유는 앞서 얘기했던 것처럼 무료 호스팅이라는 점과 웹 기반이라는 점이 한몫 하였다. 하지만 그것만으로는 깃허브가 이렇게 인기를 끌지는 못했을 것이다. 깃허브의 가장 큰 장점은 손쉬운 참여이다. 특히나 Fork 기능을 통해 인기 있는 깃허브의 오픈 소스 프로젝트를 내 저장소로 끌어와서 손쉽

게 소스를 다운로드 하거나 문제가 되는 부분을 제기하고 혹은 수정해서 제시할 수 있다는 점이 오픈 소스의 손쉬운 접근 및 대중화를 이끈 핵심 기능이라고 할 수 있다. 기존에는 오픈 소스 프로젝트에 참여하기 위해 해당 커뮤니티에 가입하고 또 소스 수정을 위해 권한을 받아야 하는 등 여러 가지 해야 할 작업들이 많았다. 즉, 참여하기가 어렵다는 불편한 점이 있었다. 하지만 깃허브에 있는 오픈 소스 프로젝트들은 Fork 기능을 통해 누구나 해당 오픈 소스 프로젝트에 참여할 수 있게 되었고 이슈를 제기하고 또 자신이 수정한 이슈 사항을 올려서 손쉽게 반영할 수 있게 함으로 누구나 오픈 소스 프로젝트에 참여할 수 있도록 손쉬운 접근 방법을 제시해주었다. 오픈 소스가 최근 활성화되고 또 규모가 커진 이유 중 하나가 바로 손쉬운 참여라고 할 수 있는데 깃허브는 그 공로자 중 하나라고 할 수 있다.

Android

Android 마스코트 이미지

“Android 생태계의 중심에 있는 보안”

Google의 목표는 Android를 세계에서 가장 안전한 플랫폼으로 만드는 것이다. Google이 기기, 애플리케이션 및 글로벌 생태계의 보안을 강화하는 기술과 서비스에 투자하는 이유다.

Android가 오픈소스인 이유도 여기에 있다. Android를 오픈소스로 제공하여 Android를 더욱 안전하게 만들 수 있는 혁신적인 아이디어를 제공하는 전 세계의 보안 전문가와 소통할 수 있다. 전 세계의 보안 전문가는 Android의 코드를 검토하고, 새로운 보안 기술을 개발 및 배포하고, Android의 보안 강화에 기여할 수 있다.

Android 생태계가 발전함에 따라 Google은 첨단 보안 기술에 지속적으로 투자하고 있으며 관련 지식을 여러분과 공유하고자 한다.

"언제나 자동으로 보호"

Android 기기에는 앱 인증이라는 기본 소프트웨어가 내장되어 있어 기기에 있는 모든 앱이 올바르게 작동하는지 정기적으로 검사한다. 유해한 앱이 발견되면 앱 인증에서 알림을 표시하거나 앱을 완전히 차단한다.

"지문을 사용하여 보안 절차 간소화"

Fingerprint API를 통해 사용자는 탭 한 번으로 기기를 잠금 해제하고 앱에 안전하게 로그인하고 Android Pay 및 Play 스토어를 사용할 수 있다. 또한 개발자가 타사 앱에 기능을 탑재하는 데 도움이 되도록 Android에는 Fingerprint API가 포함되어 있다.

“데이터를 활용하여 사용자 보호”

Android의 보안을 더욱 강화하기 위해 SafetyNet 유틸리티에서는 10억 개가 넘는 Android 기기의 앱, 설정, 중요 보안 데이터를 정기적으로 검사한다. 앱에서 비정상적인 작동이 감지되면 Google Play에서 자동으로 게시를 중지한다. 사용하는 Android 버전과 관계없이 언제나 사용자를 안전하게 보호한다.

“켜는 순간부터 기기 보호”

기기 소프트웨어가 시작되면 자체 검사 부팅에서 기기 소프트웨어가 손상되지 않았는지 확인한다. 이 작업은 기기 하드웨어에 내장된 키를 사용하므로 항상 신뢰할 수 있는 출처를 통해 소프트웨어가 작동하도록 한다. Android가 사용자의 데이터와 기기를 보호하는 또 다른 방법이다.

“기기 분실 시에도 보호”

내 기기 찾기를 사용하여 Google 계정과 연결된 분실 기기의 위치를 원격으로 확인할 수 있다. 또한 화면 잠금 PIN을 추가하거나, 도난 또는 분실 후 찾

을 수 없는 경우 기기 내의 모든 데이터를 삭제하여 위험한 상황을 피할 수 있다.

“모든 앱을 보호”

모든 Android 앱은 각각의 애플리케이션 샌드박스 안에서 작동한다. 앱에서 다른 앱에 있는 개인정보에 액세스하는 것과 같은 원치 않는 작업을 시도하면 샌드박스가 작업을 중단하여 사용자를 안전하게 보호한다.

“샌드박스는 Google 포토 앱을 위한 또 하나의 핵심 보안 단계다. 샌드박스는 사용자 기기의 개인 사진 및 동영상 라이브러리를 안전하게 보호해 사용자가 안심하고 추억이 담긴 사진을 앱에 보관할 수 있도록 한다.“

- INDRAJIT KHARE Google 포토 기술 책임 및 관리자, Android

Android는 광범위한 모바일 기기 및 Google이 주도하는 해당 오픈 소스 프로젝트를 위한 오픈 소스 소프트웨어 스택이다. 이 사이트와 AOSP (Android

Open Source Project) 리포지토리는 Android 스택, 포트 장치 및 액세서리의 Android 버전에 맞춤 변형을 생성하고 기기가 호환성 요구 사항을 충족시키는 데 필요한 정보와 소스 코드를 제공한다. 한 업계 선수가 다른 업체의 혁신을 제한하거나 통제 할 수 있는 중앙 실패 지점이 없는지 확인하기를 원했다. 결과적으로 사용자 정의 및 포팅을 위해 소스 코드를 열어 놓은 컨수머 제품을 위한 양질의 운영 체제가 완성되었다.

“비용없이 가치 창출”

Android Open Source Project(AOSP)에서는 이메일, 통화 등을 제공한다. 하지만 GMS는 제공하지 않는다. GMS는 Google과의 라이선스 계약을 통해서만 사용할 수 있다. 인기 앱, 클라우드 기반 서비스를 포괄적으로 제공한다. 기기에 GMS를 설치하는 데는 라이선스 비용이 따로 들지 않는다.

오픈소스와 친해지기

오픈 소스를 처음 접하는 사람들에겐 오픈 소스에 대한 이야기가 먼 나라 이야기처럼 들릴 수 있다. 하지만 프로그래머로서 오픈 소스 활동은 필수불가결한 일이기 때문에 프로그래머라면, 혹은 프로그래머를 꿈꾸는 사람이라면 반드시 오픈 소스와 친해질 필요가 있다. 그렇다면 어떻게 해야 오픈 소스와 친해질 수 있을까? 우선 세 가지를 짚고 넘어가자.

1. Github에 가입하라.

2. Git의 사용법을 익혀라.

3. 나의 관심 분야를 찾아라.

프로그래밍을 접해본 사람이라면 모르는 사람이 없을 정도로 유명한 Github는 세계에서 가장 인기있는 Git 호스팅 사이트로, 가장 인기있는 오픈 소스 코드 저장소이기도 하다. Github에 가입하는 것은 오픈 소스 활동의 첫 걸음을 내딛는 것과 다름없다. Github에 가입하고 나면 Git의 사용법을 익혀야

한다. (Github가 Git의 호스팅 사이트임을 생각하면 Git의 사용법을 익히는 것은 당연한 일이다.) Git은 소스코드 관리를 위한 분산 버전 관리 시스템인데, 저 문장으로는 Git이 무엇인지 전혀 이해하지 못 할 것이다. 하지만 직접 사용법을 익히고 나면 조금씩 이해가 될 것이니 서적 또는 인터넷에서 사용법을 찾아 익히는 것이 바람직하다. 인터넷에는 Git의 사용법에 대한 많은 양의 자료들이 있으므로 충분히 사용법을 익힐 수 있다. Git의 사용법까지 익혔다면 다음은 자신의 관심 분야에 대해 생각해 보아야 한다. 관심이 있는 분야에서 활동하는 것이 오랜 시간 작업하기에 좋고 자신의 실력을 갈고 닦을 수 있다는 것은 분명한 사실이다. 혹은 관심이 있는 프로젝트가 있다면 그 프로젝트에 참여하는 것도 좋다. 처음엔 초심자에게 알맞은 과제를 수행하고, 점차 자신의 영역을 넓혀 나가는 식으로 실력을 쌓자.

그렇다면 프로그래머로서 오픈 소스를 마주하는 태도는 어떠해야 할까? 오픈 소스는 소프트웨어 산업의 발전에 큰 기여를 했으며 이제는 개발자 채용에 있어서 중요한 요소가 되었다. 만약 당신이 프로그래머로서 회사에 취직하고 싶다면 오픈 소스 활동은 큰 도움이 될 것이다. 하지만 단순히 취업만을 생각하고 오픈 소스 활동을 시작한다면 그것은 참된 의미의 활동이 아니다. 오픈 소스의 목적은 공유를 통해 더욱 완성도 높은 소프트웨어를 만들고자 하는 것이지 취업을 위해 반강제적으로 행하는 오픈 소스 활동은 진정한 의미의 오픈 소스 활동이라고 할 수 없다. 오픈 소스의 역사는 진행중이며 앞으로도 진행되어야 한다. 역사의 흐름을 그대로 후세에 전해주기 위해서는 우리의 참된 노력이 필요하다. 더 나은 미래를 위해 노력한다는 자세로 오픈

소스 활동에 임하자.

오픈소스의 미래

“현재 사용 이유”, “사용해야 하는 이유”, “소통”

오픈소스 프로젝트를 공개하고 운영하는 것은 단순히 소스 코드만을 공개하는 것에 그치지 않고, 같은 문제로 고민하고 있는 외부 개발자들과 소통하고 더 나은 소프트웨어를 만들기 위해 노력하는 모든 활동을 의미한다.

개인이나 기업이 프로젝트를 오픈소스로 공개하고 운영하는 경우를 쉽게 찾아볼 수 있는데, 이엔 여러 가지 이유가 있다.

1. 원래 사용하던 오픈소스의 새로운 버전을 시작하거나 중단된 운영을 이어가기 위해

오픈소스를 사용하다가 필요한 기능을 직접 추가했을 때 보통 새로 만든 기능을 프로젝트에도 반영하고 싶어 한다. 하지만 추가한 내용이 유용해도 커뮤니티의 비전이나 프로젝트의 작업 우선순위에 따라 컨트리뷰션이 받아들여지지 않을 수 있다. 이런 경우에 원래의 오픈소스를 포크(fork)해서 직접 새로운 기능을 추가하고 별도의 프로젝트로 운영할 수 있다. 또는 사용하고 있던 오픈소스의 운영이 중단되어 해당 프로젝트를 포크한 다음 운영을 이어갈 수도 있다.

2. 오픈소스 라이선스 의무를 지키기 위해

GPL(GNU General Public License)이나 AGPL(Affero General Public License), MPL(Mozilla Public License) 등의 라이선스로 배포되는 오픈소스를 사용하면 라이선스 조항에 따라 산출물로 나온 소프트웨어의 소스 코드를 공개해야 할 때도 있다. Xiaomi kernel 프로젝트는 GPL 의무 조항을 준수하기 위해 공개한 Xiaomi의 커널 코드이다.

3. 더 나은 소프트웨어로 성장하기 위해

누구나 개발에 참여할 수 있도록 프로젝트를 공개하면 외부 개발자들과 함께 문제를 공유하고 해결할 수 있다. 소스 코드에 대한 컨트리뷰션 외에도 다양한 테스트 환경과 오류 신고 등으로 소프트웨어의 품질을 높일 수 있다.

4. 사회 공헌을 위해

일반적으로 서비스나 제품에 필요한 소프트웨어를 만들기 위해서는 오픈소스 기술이 반드시 필요하다. 이렇게 오픈소스 생태계에서 얻은 혜택을 다시 오픈소스 생태계에 돌려주기 위해 오픈소스를 운영하는 경우도 있다. 네이버도 이런 이유로 내부에서 개발한 다양한 도구를 오픈소스 프로젝트로 공개해서 운영하고 있다.

하지만 오픈소스를 사용하기 위해선 주의해야 할 점도 존재한다.

오픈소스를 사용할 때는 오픈소스의 출처와 저작권, 라이선스 정보를 남겨두어야 한다. 그리고 저작권과 라이선스 주석을 유지해야 한다. 많은 오픈소스는 소스 코드의 시작 부분에 저작권과 라이선스 정보를 주석으로 명시해 둔다. 그리고 대부분의 오픈소스가 저작권과 라이선스 정보 표기를 의무 조항으로 두고 있다. 하지만 오픈소스를 사용할 때 무심코 이 주석을 지우는 경우가 있다. 만약 이후 자신의 프로젝트를 오픈소스로 공개할 때를 대비해서

라도 프로젝트에 사용한 오픈소스의 출처를 밝히고 저작권과 라이선스 정보를 유지해야 한다. 소스 코드 유지 보수 및 관리를 위해서도 출처를 남겨 두는 습관은 중요하다.

파일 단위나 함수 단위의 오픈소스도 출처를 꼭 명시해야 한다. 오픈소스를 파일이나 모듈 단위로 일부만 사용할 때는 출처나 저작권, 라이선스 정보를 누락하기 쉽다. 만약 별도의 저작권과 라이선스 표기가 없더라도 자신이 작성한 소스 코드가 오픈소스로 공개될 때를 대비해 출처를 명시해 자신의 코드를 다른 사람의 코드와 구분할 수 있게 해야 한다.

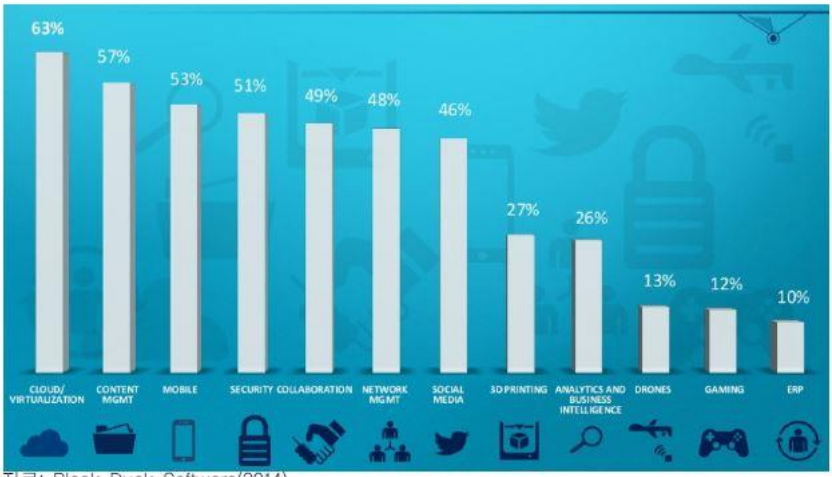
현재 오픈소스를 지향하는 기업들이 점점 증가하고 있다는 점에서 유추할 수 있듯 오픈소스의 전망은 상당히 밝다.

◇ 오픈소스 SW 시장 규모 및 전망 [자료=정보통신산업진흥원]

“
최근 3대 ICT TREND KEYWORD : Cloud - Openstack, Big Data - Hadoop, IoT - Linux
”

오픈소스는 클라우드·가상화, 콘텐츠 관리, 모바일, 보안, 협업, 네트워크 관리, 소셜미디어 등의 분야에서 중요성이 커지고 있는 것으로 나타난다. 특히, 클라우드, 빅데이터, 운영시스템, 연결제품·IoT 등의 분야에서 오픈소스 소프트웨어의 영향력이 높아질 전망이다.

[그림 4-2] 오픈소스가 기술 산업을 이끌고 있는 주요 분야



자료: Black Duck Software(2014)

IoT 환경이 확산되면서 수많은 기업들의 기기·네트워크·데이터 간 상호호환성과 연동성이 가장 중요한 과제이며, 이를 기술적으로 해결할 수 있는 방법은 오픈소스 소프트웨어가 유일한 대안이다.

소수의 독점적 기업이나 조직이 아닌 다수의 기업이 자유롭게 참여하여, 오픈소스 기반으로 소프트웨어·애플리케이션을 개발하고, 이를 표준화하는 방법으로 해결한다. 각 개별 기업별로 IoT 환경을 지원하는 소프트웨어나 기술을 개발하면 호환성을 보장할 수 없기 때문에 IoT 환경의 이용과 확산에 걸림돌로 작용한다.

즉, IoT 구현과 확산의 전제조건은 표준 플랫폼 확립과 호환성이며, 이를 실현하기 위해서는 누구나 쉽게 참여하여 소프트웨어와 애플리케이션을 개발할 수 있는 개방형 체제를 구축하는 것이 중요하다.

특히, IoT는 기기 상호 간의 연결성과 호환성도 중요하지만, 네트워크 기술, 데이터 처리 기술, 클라우드 기술 등이 필수적으로 융합되어야 하는 거대 플랫폼 환경이다. 따라서 소수의 기업이 자체적 혹은 M&A를 통해 추진하거나, 몇몇 기업의 제휴로는 해결할 수 없다. AllSeen Alliance, Thread Group, Open Interconncet Consortium, oneM2M, OCEAN 등과 같이 많은 기업체와 기관이 연합체와 컨소시엄을 구성하는 것은 IoT 산업에서의 개방성과 오픈소스 소프트웨어의 중요성을 말해주는 의미로 해석한다. AllSeen Alliance6)는 인터넷에 연결된 수많은 장리를 서로 이해할 수 있는 코드를 만들어 공유한다는 취지로 설립한다. Thread Group은 구글이 인수한 네스

트랩스가 주도하고 있으며, IoT 디바이스 간 연결을 가능하게 하는 프로토콜 개발에 집중한다.

Open Interconnect Consortium은 프로토콜보다 더 큰 개념인 플랫폼을 표준화하기 위한 연합전선이다. IoT와 밀접한 관계에 있는 클라우드와 빅데이터 부문에서도 오픈소스의 확장은 가속화되고 있는 추세다.

오픈스택은 대표적인 클라우드 오픈소스 플랫폼으로 도입 비용, 기술 접근성, 확장성을 무기로 영역을 넓혀가고 있으며, 이 외에도 유클립투스, 오픈네뷸라 등의 오픈소스 기반 클라우드 플랫폼이 있다.

빅데이터 부문에서는 빅데이터를 분석하기 위한 데이터 수집, 원본데이터 저장, 트랜젝션 데이터 저장, 배치 분석 플랫폼, 데이터 마이닝/통계, 데이터 클러스터 관리 및 모니터링, 데이터 직렬화 등에 오픈소스가 적용된다.

〈표 4-1〉 오픈소스 기반 빅데이터 필요 기술

빅데이터 필요기술	의미	오픈소스 기술
데이터 수집	• 데이터 발생원으로부터 안정적인 저장소로 저장하는 기능	Flume, Scibe, Chukwa
원본 데이터 저장	• 수집된 데이터를 안정적으로 저장하는 저장소, 주로 대용량 파일 저장소	Hadoop FileSystem, MogileFs
트랜잭션 데이터 저장	• 원본 데이터를 실시간으로 저장, 조회 처리를 위한 저장소 • 구조적 저장소 또는 검색 엔진 기술을 활용	NoSQL(Cloudata, HBase, Cassandra), Katta, ElasticSearch
배치 분석 플랫폼	• 전체 또는 부분 데이터에 대해 복잡하고 다양한 분석을 수행 • 대용량 처리를 위한 분반, 병렬처리가 필요 • 단순 텍스트 분석부터 그래프 분석까지 다양한 분석 모델 지원	Hadoop MapReduce(Hive, Pig), Giraph, GoldenOrb
데이터 마이닝 / 통계 도구	• Cluster, Classification 등과 같이 데이터 마이닝을 위한 기본 알고리즘, 라이브러리, 도구	R, Mahout
클러스터 관리 및 모니터링	• 대부분 분산 시스템으로 구성되기 때문에 전체 클러스터에 대한 관제 및 모니터링 수행	Zookeeper, HUE, Cloumon
데이터 직렬화	• 이기종 플랫폼 및 다양한 종류의 솔루션을 사용하기 때문에 데이터 전송 및 처리에 대한 표준 프레임워크 필요	Thrift, Avro, ProtoBuf

자료: http://blog.naver.com/lbjqudwn/220426153067 블로그 내용 정리

한편, 폭발적인 시장 성장과 다양한 제품들의 경쟁이 가속화되고 있는 드론 분야에서도 오픈소스 소프트웨어가 참여하여 본격적인 경쟁에 돌입한다. 리눅스재단은 3D로보틱스와 바이두, 박스, 드론디플로이, j드론스, 레이저 내비게이션, 스카이워크, 스콰드론 시스템, 월케라, 유닉, 인텔, 퀄컴 등 다국적

IT 기업들과 드론 전문기업들이 참여하는 '드론코드 포르젝트'를 발표한다. 전세계 6,000여 명의 개발자가 참여하여 OS와 드론 하드웨어를 개발하고 있는 커뮤니티 성격의 오픈파일럿 프로젝트도 가동한다.

3D 프린터 분야에서도 오픈소스 소프트웨어 및 플랫폼의 중요성에 대한 인식이 높아지고 있다. 현재 3D프린터 제조 기업들은 대부분 독자 SW를 사용하고 있으나, 가격 경쟁력 확보 차원에서 오픈소스 소프트웨어 및 플랫폼의 진출이 시작되고 있는 상황이다. 렙랩(www.reprap.org), 이벤트오봇(www.eventorbot.com), 탄틸러스(www.tantillus.org) 등이 오픈소스 소프트웨어와 플랫폼으로 3D 프린터 시장에서 경쟁한다. 오토데스크도 오픈소스 SW 플랫폼 '스파크'가 적용된 3D 프린터를 출시하였으며, 최근에는 마이크로소프트의 윈도10이 스파크 플랫폼에 대한 지원을 발표했다.

참고 자료

■ 오픈소스, 공유

http://verticalplatform.kr/archives/4038

https://ko.wikipedia.org/wiki/%EC%98%A4%ED%94%88_%EC%86%8C%EC%8A%A4

https://brunch.co.kr/@bumgeunsong/15

https://kocoafab.cc/tutorial/view/110

https://www.oshwa.org/definition/korean/

http://tronixstuff.com/2011/07/08/tutorial-arduino-and-a-thermal-printer/

http://terms.naver.com/entry.nhn?docId=1228317&cid=40942&categoryId=32837

https://brunch.co.kr/@bumgeunsong/15

https://en.wikipedia.org/wiki/History_of_free_and_open-source_software

■ 과거의 소프트웨어

https://en.wikipedia.org/wiki/History_of_free_and_open-source_software

■ 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_%EC%9A%B4%EB%8F%99

http://rakuraku.tistory.com/107

■ Android

https://www.android.com/