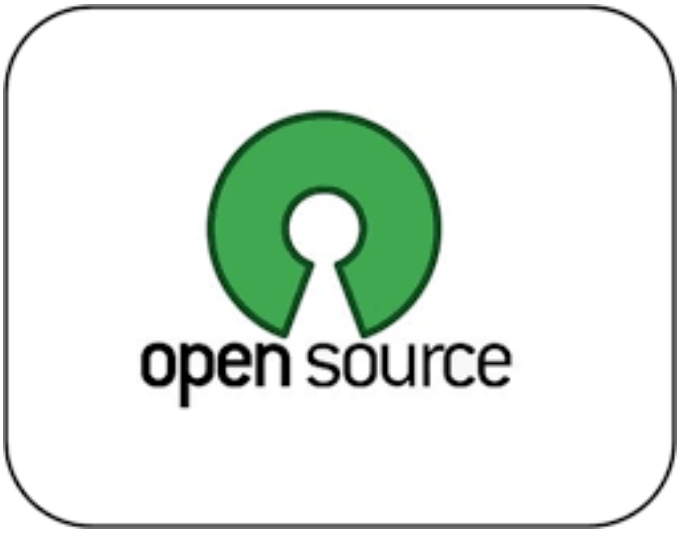
목차

1. 소개

1.1 머리말



우리는 더 이상 컴퓨터 없이는 살아갈 수 없을 정도로 그와 밀접한 관련을 맺고 있다. 지금 이 글을 쓰는 매체도 또한 컴퓨터이고, 당신이 이 글을 읽는 매체도 또한 컴퓨터일지 모른다. 잠시 이 글에서 눈을 떼고 주위를 둘러보라. 무엇이 보이는가? 당신의 주변에는 온통 컴퓨터로 가득할 것이다.

당신은 또한 컴퓨터 한 대를 언제나 들고 다닐 것이다. 아니라고? 당신의 손을 내려다보라. 자그마한 검은 화면이 당신을 반기지 않는 가. 그것도 또한 컴퓨터이며, 경우에 따라 오히려 더 뛰어난 성능을 낼 수도 있는 종류이다. 지금도 수많은 전문가들이 컴퓨터 프로그램과 밤낮없이 싸우고 있을 것이다. 새로운 것을 만들기 위해서일지도 모르고, 더 나은 것을 만들기 위해서일지도 모른다. 다만 이제는 혼자만의 싸움이 아니게 되었다. 혼자서 모든 것을 떠안고 가는 시대는 지났다.

기술의 분야는 넓어졌고, 삶의 질도 높아졌다. 이용자가 필요로 하는 것은 점점 진화하고, 고급스러워졌다. 그들의 수요를 맞추기 위해, 더는 혼자가 아니라, 공동으로 작업을 하게 되었다. 오픈 소스, 이것이 바로 그 공동 작업의 가장 큰 울타리이자, 미래를 잇는 계보가 될 것이다. 지금 우리의 삶을 흐르는 조류. 그것은 바로 오픈소스이다.

1.2 오픈소스란?

오픈소스란 소프트웨어나 하드웨어 제작자의 권리를 지키면서 누구나 자유롭게 소스 코드를 열람할 수 있도록 한 소프트웨어, 혹은 오픈소스 라이선스에 준하는 모든 것을 일컫는다.

오픈소스라는 말은 1998 년 2 월 3 일 미국 캘리포니아 마운틴 뷰의 VA 리눅스 시스템즈 사무실에서 열린 전략회의에서 처음 만들어졌다. 용어 자체는 포어인사이트 인스티튜드의 사장인 크리스틴 페터슨이 만들었다. 이는 자유(Free)라는 단어의 또 다른 뜻인 공짜(Free)라는 인식을 바꾸기 위해 오픈소스라는 표현을 사용한 것이다.

오픈소스는 특정 라이선스에 따라 일반적으로 자유롭게 열람하여 사용할 수 있도록 소프트웨어의 소스 코드가 공개되어 있다. 사용자는 소스 코드를 이용하여 복사, 수정, 개작, 재배포 등을 할 수 있다. 이는 소프트웨어 뿐만 아니라 하드웨어에도 적용되며, 사용 시, 원 제작자의 권리를 지키는 것도 중요하다.

오픈소스의 대표적인 예로는 리눅스(Linux)가 있는데, 리눅스는 누구나 무료로 이용할 수 있고 공개된 코드를 활용해 프로그램을 마음껏 변형할 수 있다.

이 리눅스를 기반으로 페도라(Fedora), 우분투(Ubuntu), 센트 OS(CentOS) 등 다양한 배포판이 만들어 졌으며, 모바일 운영체제인 안드로이드(Android) 역시도 리눅스 커널을 기반으로 만들어졌다. 그리고 크로미움(Chromium)이라는

구글 크롬(Google Chrome)의 기반이 되는 오픈 소스도 존재한다.
이처럼 오픈소스는 벌써 널리 퍼져 있고, 조금만 관심을 기울이면 금방 찾을 수 있으며, 앞으로의 개발 행보에 지대한 영향을 미칠 것이다.
그렇다면 오픈소스를 마음껏 가져다 쓰기만 하면 될까? 답은 '그렇지 않다'이다. 상술했듯이, 원 제작자의 권리를 지키는 것도 중요하기에 '반드시' 출처와 저작권, 라이선스 정보를 남겨두어야 한다. 많은 오픈소스가 소스 코드의 시작부에 저작권과 라이선스 정보를 주석으로 명시하고, 대부분의 오픈소스가 저작권과 라이선스 정보 표기를 의무 조항으로 두고 있다. 하지만 우리는 무심코 이런 주석을 지우는 경우가 잦은데, 이후 자신이 오픈소스를 배포할 때를 대비해서라도 주의를 기울이는 노력이 필요하다.

1.3 오픈소스 라이선스

오픈소스 라이선스란 소스코드의 저작자가 자신의 저작물을 활용할 때 지켜야 할 규칙을 법적으로 명시해 놓은 것이다. 때문에 이 라이선스를 제대로 지키지 않을 경우 법적으로 처벌을 받을 수 있음을 염두에 두어야 한다. 위에서 이야기했듯이, 무심코 소스코드 상단에 있는 주석을 지울 경우, 법적으로 처벌받을 수 있음을 명심하자.

흔히들 오픈소스의 라이선스가 모든 소프트웨어에 공통된다고 오해하는 사람들이 많은데, 오픈소스의 라이선스도 종류가 다양하고, 그 종류별로 조금씩 다른 규칙을 가지고 있다. 누군가의 필요로 인해 규칙을 바꾼 경우도 있고, 혹은 이해관계로 인해 그 규칙을 바꾼 것도 있다.
아래의 표는 그 몇몇 라이선스들의 규칙을 간단하게 보여주고 있다.

라이선스	무료 이용	배포 허용	소스코드 취득 가능	소스코드 수정 가능	2 차 저작물 공개 의무	독점 SW 와 결합 가능
GPL	**O**	**O**	**O**	**O**	**O**	**X**
LGPL	**O**	**O**	**O**	**O**	**O**	**O**
MPL	**O**	**O**	**O**	**O**	**O**	**O**
BSD	**O**	**O**	**O**	**O**	**X**	**O**

라이선스	무료 이용	배포 허용	소스코드 취득 가능	소스코드 수정 가능	2 차 저작물 공개 의무	독점 SW 와 결합 가능
APACHE	**O**	**O**	**O**	**O**	**X**	**O**

GPL : General Public License 의 준말. 저작권은 개발자에게 있지만, 소프트웨어의 수정, 변경, 복사와 배포의 자유를 제 3 자에게 허용하는 라이선스이다.

LGPL : GNU Lesser General Public License 의 준말. GPL 을 변형해 더욱 넓은 분야를 허가한 형태로서, 소프트웨어 라이브러리를 염두에 둔 것이라고 한다.

MPL : Mozilla Public License 의 준말. 모질라 썬더버드, 모질라 파이어폭스 등 모질라 계열 소프트웨어들에 적용되는 라이선스이다.

BSD : 버클리의 캘리포니아 대학에서 배포하는 공개 소프트웨어의 라이선스. GPL 보다 훨씬 개방적인 4 개항의 간단한 문구로 되어 있는게 특징이다.

APACHE : Apache 소프트웨어 재단에서 자체적으로 만든 라이선스 규정. 누구나 해당 소프트웨어에서 파생된 프로그램을 제작할 수 있으며, 저작권을 양도, 전송할 수 있는 것이 특징이다.

1.4 오픈소스의 조건

오픈소스에도 조건이 있다. 자유 소프트웨어에 무슨 조건이냐고 따지지 말라. 자유에는 원래 책임이 따르는 법이다. 아래는 그 오픈소스의 조건들이다.

1. 자유로운 재배포 라이선스는 여러 다른 출처의 프로그램이 포함된 집합적 소프트웨어 배포의 구성 요소로써 소프트웨어를 판매하거나 양도하지 못하도록 제한하지 않는다. 라이선스는 그러한 판매에 대해 로열티 또는 기타 수수료를 요구하지 않는다.

2. 소스 코드 포함 프로그램은 소스 코드를 포함해야 하며 컴파일 된 양식 뿐만 아니라 소스 코드로도 배포 할 수 있어야한다. 어떤 형태의 제품이 소스 코드와 함께 배포되지 않는 경우 합리적인 복제 비용 이상으로 소스 코드를 얻고, 바람직하게는 인터넷을 통해 무료로 다운로드하는 방법이 있어야한다. 소스 코드는 프로그래머가 프로그램을 수정하는 기본 형식이어야 한다. 의도적으로 난독화 된 소스 코드는 허용되지 않는다. 프리프로세서(preprocessor)의 아웃풋(output) 또는 트랜슬레이터(translater) 같은 중간 형태도 역시 사용될 수 없다.

3. 파생 작업 라이선스는 수정 및 파생된 저작물을 허용해야 하며 원본 소프트웨어의 라이선스와 동일한 조건으로 배포할 수 있어야한다.

4. 소스 코드의 무결성 라이선스는 빌드 타임에 프로그램을 수정하기 위한 목적으로 소스 코드와 함께 "패치 파일"의 배포가 허용되는 경우 에만 소스 코드가 수정된 형태로 배포되는 것을 제한할 수 있다. 라이선스는 수정된 소스 코드로 빌드 된 소프트웨어의 배포를 명시 적으로 허용해야 한다. 라이선스는 파생된 저작물이 원래 소프트웨어와 다른 이름 또는 버전 번호를 포함하도록 요구할 수 있다.

5. 개인 또는 그룹의 평등 라이선스는 어떤 개인이나 집단을 차별해서는 안 된다.

6. 분야에 대한 평등 라이선스는 특정 분야에서 프로그램을 사용할 수 있도록 해야 하고, 그것을 규제해서는 안 된다.
   예를 들어, 프로그램이 비즈니스에서 사용되거나 유전 연구에 사용되는 것을 제한하지 않을 수 있다.

7. 라이선스 배포 프로그램에 대한 권리는 프로그램이 재배포 된 모든 사람에게 해당 당사자가 추가 라이선스를 발급할 필요 없이 적용해야 한다.

8. 제품 스펙에 따른 라이선스 프로그램에 대한 권한은 프로그램이 특정 소프트웨어 배포의 일부가 되어서는 안 된다.
   프로그램이 해당 배포에서 추출되어 프로그램 라이선스 조건에 따라 사용되거나 배포되는 경우 프로그램을 재배포하는 모든 당사자는 원래 소프트웨어 배포와 함께 부여된 것과 동일한 권한을 가져야한다.

9. 라이선스는 다른 소프트웨어를 제한하지 않음 라이선스는 라이선스를 받은 소프트웨어와 함께 배포된 다른 소프트웨어에 제약 사항을 두어서는 안 된다.
   예를 들어, 라이선스는 동일한 매체에 배포된 다른 모든 프로그램이 오픈소스 소프트웨어여야 한다고 주장해서는 안 된다.

10. 라이선스는 기술 중립적이어야 함 라이선스는 기술이나 인터페이스 스타일을 한정해서는 안 된다.

1.5 오픈소스의 참여 모델

오픈소스의 참여 모델은 위와 같은 구조를 하고 있다. 먼저 개발자들이 최고 상위에 위치해 있고, 커뮤니티가 그 다음이며, 그 뒤를 이어 사용자, 잠재적 사용자 순으로 모델이 나타난다. 위 그림을 보면 알 수 있듯, 아직 오픈소스의 시장은 더욱 커질 가능성이 매우 높다.

2. 역사

2.1 1980년대 이전

1950년대와 1960년대에 거의 모든 소프트웨어는 공동 연구로 일하는 학자 및 기업 연구원에 의해 제작 및 공유가 이루어지고 있었다. 그래서 소프트웨어는 학계의 분야에서 오랫동안 확립된 개방과 협력의 원칙에 따라 일반적으로 배포되었고, 그 당시 사용자들은 버그를 수정하거나 새로운 기능을 추가하기 위해 소프트웨어 자체를 자주 수정했기 때문에 인간이 읽을 수 있는 형태의 소프트웨어인 소스코드는 일반적으로 소프트웨어 기계코드와 함께 배포되었다. 또한 일부 대학에서는 컴퓨터실의 컴퓨터에 설치된 모든 프로그램에 게시된 소스코드 파일이 있어야 한다는 정책조차 있었다.

2.2 자유소프트웨어의 하락

하지만 1960년대 후반에는 운영체제 및 프로그래밍 언어 컴파일러가 발전하면서 소프트웨어 생산 비용이 하드웨어에 비해 크게 증가했다. 후에 일부 소프트웨어는 무료로 계속 제공되었지만 제한적인 라이선스 하에서만 판매되는 소프트웨어가 점차 증가했다.

비록 1970년대 초 AT&T는 유닉스의 초기 버전을 정부 및 학술 연구자에게 무료로 배포했지만 이 버전은 수정된 버전을 재배포하거나 배포할 수 있는 권한이 없었기 때문에 자유 소프트웨어가 아니었다. 1970년대 후반에 들어서는 소프트웨어 전용 회사는 소프트웨어 라이선스를 통상적으로 요금을 청구하기 시작했다.

하지만 사람들이 라이선스 비용을 지불하지 않자 1976년 빌 게이츠는 '애용자들에게 공개 서한'이라는 제목의 에세이를 써서 사용자가 라이선스 비용을 지불하지 않고 마이크로소프트의 제품인 Altair BASIC을 광범위하게 공유하는 것을 보고 "소프트웨어를 만든 사람들에게 대가를 주지 않는다는 것은 도둑질이다." 라고 낙담하며 밝히기도 했다. 이때부터 수익을 높이기 위해 소스코드를 배포하지 않고 소스코드에서 컴파일 된 실행파일인 기계코드만 배포하는 관행이 시작되었다. 이 새로운 관행에 특히나 고통받은 사람은 '리처드 스톨만'이었다.

'스톨만'은 처음에 다른 사람들이 작성한 프로그램을 더 이상 연구하거나 수정할 수 없다고 우려했고 이 관행이 윤리적으로 잘못된 것으로 보았다. 그래서 스톨만은 1983년에 GNU 프로젝트를 설립하여 사람들이 자유 소프트웨어만을 사용하여 컴퓨터를 사용할 수 있게 했다.

2.3 1980년대 이후

이때도 여전히 소스코드를 다른 사람들과 무료로 공유하고 싶어하는 사람들이 있었으며 이들을 '애용자' 및 '해커' 라고 불렀다. 이들은 인터넷이 도입되고 광범위하게 사용되기 이전, 컴퓨터 잡지나 컴퓨터 프로그래밍 책에 소스코드를 전부 집어넣어서 소스코드를 공유하는 방법을 사용했다.
1955년에 설립된 SHARE 사용자 그룹은 무료 소프트웨어를 수집하고 주로 자기 테이프를 통해 배포하기도 하고 어떤 사람들은 DECUS 테이프를 통해 배포하기도 했다. 그러다 1980년대에는 자유 소프트웨어 운동과 나란히 소스코드가 있는 소프트웨어가 BBS 네트워크에서 공유되었다.

사용자가 소스코드를 모으고 수정을 논의하기 위해 구체적으로 보드를 설정하기 시작했고, 이것은 사실상 오픈소스 시스템이었다.
가장 확실한 사례 중 하나는 가장 많이 사용된 BBS 시스템 및 네트워크 중 하나인 WWIV입니다. 이때 자신의 소프트웨어를 수정하고 수정본을 배포하는 문화는 매우 성장해서 소스코드가 가입 유저들에게 계속 배포되었다.

2.4 자유소프트웨어 운동의 시작

'리차드 스톨만'은 1983년 GNU 프로젝트를 시작하여 소스코드의 사용에 제약이 없는 완전한 운영체제를 작성했다. GNU 프로젝트는 모든 소프트웨어를 아무런 제한 없이 무료로 만들고 무료로 배포한다는 기본 이념에서 시작했다. 특히 이 프로젝트는 모든 소스 코드를 사람들에게 공개해 소프트웨어 자유롭게 개발하도록 보장해 주었다. 또한 '스톨만'은 GNU 프로젝트의 목적을 설명하고 자유 소프트웨어의 중요성을 설명하기 위해 1985년 'GNU 선언문'을 발표했으며, 1986년에는 자유 소프트웨어의 생산과 보급을 장려하기 위해 자유 소프트웨어 제단을 설립했다.

여기서 그치지 않고 '스톨만'은 1989년, 'GNU 일반 공중 규약(GPL)'의 첫번째 판을 출판했고 1991년 두번째 판을 출판했다. 하지만 '스톨만'은 1990년에 'GNU 일반 공중 규약(GPL)'에 따른 'GPL하의 완전한 운영체제'인 'GNU Hurd'를 개발했지만 커널까지는 개발해내지 못했다.

2.5 리눅스

'리누스 토발즈'에 의해 시작된 리눅스 커널은 1991년 자유롭게 수정가능한 소스코드로서 공개되었다. 리눅스의 라이선스는 자유 소프트웨어 라이선스가 아니었지만 곧바로 1992년 GPL하에 다시 라이선스를 제공했다. 또한 이때까지 커널이 없었기 때문에 완전한 자유 소프트웨어 운영체제가 존재하지 않았던 GNU 운영체제는 리눅스 커널과 결합하면서 최초의 완전한 자유 소프트웨어 운영체제가 되었다. 인터넷의 보급과 더불어 배포된 리눅스의 보급으로 자유소프트웨어 운동이 확산된 것이다.

토발즈가 리눅스를 처음 공개했을 때는 누구도 리눅스가 역대 최대 규모의 소프트웨어 프로젝트가 될 거라고 생각하지 않았다. 그 때만해도 오픈소스는 소수의 해커들 만이 참여하는 마이너한 문화였기 때문이다. 이전에는 핵심 개발자 집단이 독점하고, 완성도가 있다고 판단된 후에 소스를 공개하는 방식을 사용했다면 '토발즈'는 누구나 코드를 수정할 수 있도록 했고, 수정한 코드가 받아 들여 지면 그 사람은 기여자가 되어 다음 버전에 이름이 등록되었다. 이 시스템이 과시욕을 불러 일으켜 개발자들은 무료로 커널 수정과 기능 추가에 매진했다. 결국 점점 더 많은 사람들이 개발에 참여하게 되었고, 그들은 리눅스를 점점 더 발전시켰다.

2.6 오픈소스 출시

이때즘부터 일반인들은 자유 소프트웨어의 자유란 용어를 무료로 인식하고 있고, GPL 조항의 엄격성은 소프트웨어 개발을 막고 있었다.
1997 년 '에릭 레이먼드'가 해커 커뮤니티와 저유 소프트웨어 원칙의 분석을 반영한 '성당과 시장'을 발표했다. 이 문서는 1998 년 초에 주목을 받았으며 Netscape 가 인터넷 제품군을 자유 소프트웨어로 출시하도록 유도하는 역할을 해냈다. 이에 대한 반응으로 캘리포니아 Palo Alto 에서 열린 전략 회의에서 자유 소프트웨어 운동의 일부 사람들은 '오픈소스'라는 단어를 채택했다.
그 후, '레이먼드'와 다른 사람들은 '오픈소스'라는 단어를 퍼뜨리는 일을 했고, '레이먼드'는 자유 소프트웨어 커뮤니티에 새로운 용어를 채택하도록 첫 번째 공식적인 전화를 했다.
Open Source Initiative(OSI)는 이 직후 형성되었고, OSI 에 따르면 '스톨만'은 처음에 '오픈소스'라는 용어를 채택하는 아이디어에 솔깃했지만 굉장히 성공적인 '오픈소스'라는 용어가 '스톨만'의 자유 소프트웨어라는 용어와 그의 메시지를 묻어버렸기 때문에 후에 '스톨만'은 OSI 의 접근방식과 용어선택에 강력히 이의를 제기했다고 한다. 결국 1990 년대 말, '오픈소스'라는 용어는 대중매체에서 큰 인기를 얻었으며, 소프트웨어 산업에 인정을 받았다.
참고로 다음은 OSI 에서 정한 오픈소스 소프트웨어의 10 가지 정의(조건)이다.

- 자유롭게 재배포가 가능해야 한다.
- 소스코드를 공개해야 한다.
- 프로그램을 수정하고 재배포가 가능해야 한다.
- 원저자의 소스코드를 수정하기를 요청하는 제안이 가능해야 한다.
- 사람이나 그룹에 차별이 있어서는 안된다.
- 사용 분야에 제한을 가하면 안된다.
- 라이선스는 배포가 가능해야 한다.

- 라이선스는 어떤 시스템에도 적용 가능해야 한다.
- 라이선스가 다른 라이선스의 소프트웨어를 제한해선 안된다.
- 라이센스의 조항은 개별 기술 또는 인터페이스 스타일에 근거할 수 없다.

2.7 2000 년대

이제는 거의 모든 주요 소프트웨어들이 오픈소스로 개발되고 있다.
Facebook, Twitter, Netflix, Naver 등 오픈소스 문화 기반의 회사들이 늘어나고 있으며, 이 외에도 클라우드, 데이터베이스, 빅데이터 관리 등 거의 모든 분야에서 오픈소스 기술을 사용한다.
즉, 소스코드만으로는 아무것도 할 수 없는 시대가 되어버렸다.
심지어는 마이크로소프트가 최근 리눅스를 지원하기 시작했고, 자사 소프트웨어의 일부를 오픈소스로 교체했다.
최근에 들어서는 오픈소스의 양이 더 이상 무시하기 어려울 정도로 방대하게 증가했다.
이젠 사실상 오픈소스의 세상이 되어버린 것이다.

3. 오픈소스와 관련된 사람들

3.1 리누스 토르발스

"말은 쉽지. 코드를 보여줘."

"Talk is cheap. Show me the code."

Linus Benedict Torvalds(리누스 베네딕트 토르발스)는 1969년 12월 28일에 태어난 스웨덴계 핀란드인 소프트웨어 개발자이다. 리눅스의 아버지로 유명하며, 분산 버전 관리 시스템인 Git 등을 만들었다.

리누스는 1988년 헬싱키 대학교에 입학해 다녔고, 1996년 컴퓨터 과학 석사로 졸업했다. 석사 논문 제목은 '리눅스: 이식 가능한 운영 체제'이다. 리누스는 대학교 1학년을 끝낸 후 포병 관측 장교로 핀란드 군에 입대하여 소위로서 11개월간 복무하여 병역을 마쳤다. 1990년에 복학한 후 최초로 DEC MicroVAX에서 운영하는 ULTRIX의 형태로 유닉스를 만나게 되었다.

리누스의 컴퓨터에 대한 관심은 코모도어 VIC-20와 함께 시작했다. 이후 싱클레어 QL를 구입하고 그 운영체제를 변형시키며 OS를 변형시키고 어셈블리어 프로그램과 텍스트 에디터를 만들거나 몇 가지 게임을 프로그램 하기도 하였다. 1991년 2월 2일, 그는 인텔 80386 기반의 IBM PC를 구입하였다. 한 달 정도 페르시아의 왕자 등의 게임을 하면서 MINIX의 사본을 기다렸는데, MINIX가 도착한 직후 그는 리눅스 커널을 만드는 일을 시작하였다. 2000년 6월에 헬싱키 대학교는 리누스 토르발스에게 명예 박사학위를 수여했다.

3.2 리처드 스톨먼

리차드 매튜 스톨만(영어: Richard Matthew Stallman, 1953년 3월 16일 ~)은 자유 소프트웨어 운동의 중심인물이며, GNU 프로젝트와 자유 소프트웨어 재단의 설립자이다. 그는 이 운동을 지원하기 위해 카피레프트의 개념을 만들었으며, 현재 널리 쓰이고 있는 일반 공중 사용 허가서(GPL) 소프트웨어 라이선스의 개념을 도입했다. 그는 또한 탁월한 프로그래머이기도 하다. 그는 문서 편집기인 Emacs, GNU 컴파일러 모음 컴파일러, 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 개발과 공급에 대한 대안이 되었다.

1960년대 그의 고등학교 저학년 시절에 처음으로 개인용 컴퓨터를 접해볼 수 있는 기회를 얻었다. 그리고 지금은 사라진 시내에 있던 연구소인 IBM 뉴욕 과학센터에서 일하면서, 스톨만은 거기에서 그의 첫 번째 프로그램인 IBM 7064를 위한 전처리기(pre-processor)를 PL/I 프로그래밍 언어로 작성했다.

1971년에 하버드 대학의 신입생으로 스톨만은 MIT 인공지능 실험실의 해커가 되었다. 1980년대, 스톨만의 삶이었던 해커 공동체가 소프트웨어 산업의 상업화로 인해 점차 사라질 위기에 처하기 시작했다. 특히, 실험실 내의 다른 해커들은 심볼릭스(Symbolics)라는 회사를 차리고는, 기존의 자유 소프트웨어를 그들만의 독점 소프트웨어로 바꾸는 작업을 적극적으로 시도했다. 1983년에서 1985

년 사이의 2년 동안, 스톨만은 연구실 내에서 혼자 힘으로 심볼릭스의 결과물들과 똑같은 기능의 프로그램을 작성하여 그들의 독점을 막는 일을 계속했다. 그러나 그 당시 이미 그는 그의 세대 중에서 마지막 해커였다. 그는 비밀 유지 합의서에 사인하기를 요구 받았으며, 그의 원칙인 다른 이들과의 공유나 이웃을 돕는 것에 위배되는 작업들을 수행할 것을 요구받았다.

1985년, 스톨만은 GNU 선언문을 발표했다. 이는 유닉스에 대항하여 자유로운 대안을 만들기 위한 그의 의지와 동기를 역설한 것이었다. 그리고 얼마 안 있어 그는 비영리 기관인 자유 소프트웨어 재단을 설립했다. 그리고 그는 1989년 일반 공중 사용 허가서(GPL) 내에 카피레프트의 개념을 적용하였다. 허드(Hurd) 커널을 제외하고, 대부분의 GNU 시스템이 거의 동시에 완성되었다.

1991년, 리누스 토르발스는 GPL로 리눅스 커널을 발표했다. 이를 통해 완벽하게 기능하는 GNU 시스템인 GNU/리눅스 운영 체제가 탄생하게 되었다.

3.3 에릭 레이먼드

에릭 레이먼드(Eric Steven Raymond, ESR , 1957년 12월 4일 ~)는 미국 매사추세츠주 보스턴에서 태어났다. 인류학자이며 오픈 소스 개발 과정이 어떤 식으로 운영되고 반복되는지 설명한 그의 초기 문서인 성당과 시장의 저자로 잘 알려진 오픈 소스 운동의 대표 인물이다. 해커들이 쓰는 용어를 설명한 자곤 파일("신 해커사전")을 편집했으며, 넷스케이프의 소스 코드 공개, 모질라의 설립에도 큰 영향을 끼쳤다.

4. 오픈소스 장단점

4.1 장점

융통성

- 라이선스 비용이나 예산에 제한을 받지 않고, 다양한 오픈소스들을 테스트한 뒤 최선의 것을 선택할 수 있다.

기술 지원

- 신속한 문제 해결, 빨라진 성능 개선 프로세스, 기술의 공동 습득이 가능하다.

기술 혁신

- 유료일 경우 사용치 않았을 기술의 실험 적용이 가능하다.

재활용

- 소스 코드 접근이 가능함으로 재활용이 증가한다.

빠르고 유연한 개발

- 오픈소스 커뮤니티는 보통 최신 기술 정보 및 문제점과 해결책을 공유하는 형태로 자유롭게 운영되기 때문에 독점 프로그램에 비해 기술 발전 속도가 빠르다. 그리고 이미 검증된 소스를 사용함에 따라 개발이 빠르고 유연하다.

호환성

- 오픈소스는 주로 오픈 포멧 또는 프로토콜을 사용하기 때문에 서로 다른 소프트웨어간 상호 연동성이 보장된다. 여러 기기들이 각기 다른 네트워크를 통해 하나로 연결되는 시대에 필요한 필수적 요소이다.

낮은 진입 비용

- 오픈소스는 소스코드가 공개되어 무료 다운로드 및 소스코드의 개선 또는 수정/재배포가 가능하므로 일반적으로 초기 개발비용이 새로 개발하는 것에 비해 1/2 정도인 것으로 알려져 있다.

신뢰성과 안전성

- 오픈소스의 개발 과정을 보면 전 세계에 있는 수많은 우수한 개발자들이 직접 개발과 디버깅 과정에 참여하기 때문에 In-house 에서 폐쇄적으로 개발되는 독점 프로그램에 비해 비교적 안정적으로 동작한다. 단, 이는 많은 개발자들이 적극적으로 참여하는 프로그램일 경우에만 가능하므로 해당 소스의 개발 과정과 평판을 주의 깊게 볼 필요성이 있다.

4.2 단점

어플리케이션의 부족

- 대부분의 이용자들이 MS 윈도우 기반의 GUI 에 익숙한 반면, 오픈소스는 GUI 가 일반적이지 않다. 또 이미 표준적으로 사용되는 소프트웨어가 있는 경우 호환성 문제가 발생할 수 있다. 오픈소스는 리눅스 기반으로 개발된 애플리케이션이 많기 때문에 윈도우 기반 애플리케이션과 호환되지 않는 문제점도 있다.

빈약한 문서

- 오픈소스를 수정하여 원하는 애플리케이션을 제작하고자 할 경우 문서화가 중요한데, 상용 프로그램에 비해 오픈소스는 문서화가 제대로 되어있지 않은 경우가 많다. 경우에 따라서는 개발과정을 지체 시키는 원인이 되기도 한다.

불확실한 로드맵

- 오픈소스는 영리를 목적으로 하는 회사에서 개발되는 것이 아니라 개인의 자발적인 참여를 통해 개발되는 경우가 많기 때문에 독점 프로그램에서 볼 수 있는 로드맵을 기대하기 어렵다. 어느 날 갑자기 단종되고, 업그레이드가 중단되는 경우도 있다.

지적재산권

- 일반적으로 오픈소스를 수정한 프로그램은 사용료 없이 배포할 것을 요구하고 있다. 따라서 기업이 보유한 특허를 소스코드에 포함시켜 재배포하려는 경우 반드시 명확한 입장을 밝히고 오픈소스 저작권자의 정책을

고려해야 한다.

비숙련 사용자(초보자)들은 사용이 어렵다.

핵심 프로젝트 보다 주변에 집중해야 하는 경우가 더러 있다.

고객지원이 불리하다.

잠재적인 비용이 존재한다.

나중에 오픈소스 소프트웨어를 다루게 될 때, 이에 대한 장단점을 알게 된다면 좀 더 효과적이고 효율적인 개발이 가능할 것이다.