Table of Contents

Open-Source Pamphlet

ABOUT.

오픈소스 소개

들어가기 전..

프로그래밍 세계에는 독특한 문화가 있다.

바로 자신의 소스코드를 웹에 공개하여 다른 개발자들과 함께 더 나은 코드를 만드는 것이다.

오늘날, 많은 대기업 및 스타트업 회사들이 소스코드를 공개하는 '오픈소스 프로젝트'를 진행하고 있는데,

도대체 오픈소스는 언제, 어떻게, 왜 시작되었으며 소스코드를 공개하는 것이 무슨 이득이 있기에 많은 사람들이 참여하는 것일까?

여기 오픈소스 소책자에서는 오픈소스 소개와 오픈소스의 역사 그리고 오늘날 오픈소스를 어떻게 활용하는지를 다루고 있다.

이 내용을 통해 오픈소스를 이해하고, 적극적으로 참여할 수 있었으면 한다.

1. 오픈소스란 무엇인가?

오픈소스 공식 홈페이지에서 정의하는 '오픈소스'는 소스 코드가 공개되어 있고 자유로운 이용과 개발이 인정되고 있다는 것을 의미한다. 따라서 오픈소스 소프트웨어라하면 오픈소스 형식을 택한 소프트웨어를 말한다.

1) 다른 소프트웨어와 오픈소스 소프트웨어의 차이점은 무엇일까?

어떤 소프트웨어는 한정된 개발자들만 만들 수 있는 소스코드를 가진다. 이런 종류의 소프트웨어를 '독점소프트웨어' 혹은 '클로즈드 소프트웨어(closed software)"라 부른다. 이런 소프트웨어를 사용하려면 사용자가 소프트웨어를 마음대로 사용하지 않겠다는 동의를 해야만 한다. 대표적으로 'Microsoft Ofiice'나 'Adobe Photoshop'을 예시로 들 수 있다. 하지만 오픈소스 소프트웨어는 다르다. 사용자가 소스코드를 볼 수 있고 변형하거나 배포할 수도 있다. 대표적으로 'GNU Manipulation Program'을 들 수 있다. 이런 소프트웨어를 사용하려면 오픈 소스 라이선스를 받아들여야만 한다.

2) OSI 의 '오픈소스의 정의'

오픈소스의 정의는 오픈 소스의 범위에 포함되는 다양한 종류의 사용 허가서들이 지켜야 할 최소한의 기준을 정의해 놓은 것이다. 오픈소스의 정의에 관한 내용은 아래와 같다.

- 자유로운 재배포
- 원시 코드
- 파생 저작물 허용
- 저작자의 원시 코드 원형 유지
- 개인 및 단체에 대한 차별 금지
- 사용 분야에 대한 차별 금지
- 사용 허가의 배포
- 특정 제품에만 유용한 사용 허가의 금지
- 다른 소프트웨어를 제한하는 사용 허가의 금지

3) 그럼, 오픈소스 소프트웨어는 무료인가?

오픈소스를 설명한 바에 따르면, 오픈소스 소프트웨어는 소스코드를 공개하는 조건을 따라야 하는 것 같다. 그럼 오픈소스 소프트웨어는 '무료'와 같은 의미인가? 절대 아니다. 만약 오픈 소스 소프트웨어가 유료 사용이 조건인 라이선스를 적용한다면 , 유료가 될 수 있다. 다만, 많은 경우에서 오픈 소스 개발자들은 오픈소스 라이선스를 따르고 오픈소스 라이선스가 소스코드를 공개하도록 요구하기 때문에 무료로 사용이 가능한 것이다. 따라서, 오픈 소스 소프트웨어 자체는 '무료'와 같은 의미가 될 수 없다.

2. 자유소프트웨어

오픈소스를 알려면 자유소프트웨어에 대한 설명을 빼놓을 수 없다. 그 이유에 대해서는 역사 부분에서 자세히 다루겠지만, 그 전에 맛보기로 자유소프트웨어를 알아보도록 하자.

1) 자유소프트웨어란?

자유소프트웨어는 사용자의 자유와 공동체를 보장하는 소프트웨어를 말한다. 간단히 말하자면, 사용자가 소프트웨어를 작동시키고, 복사하고, 배포하고, 변형시키고, 발전시킬 수 있는 자유를 가진다는 것을 의미한다. 자유소프트웨어의 정의에 대해 설명하고 있는 GNU Operating System홈페이지에 따르면

> Thus, “free software” is a matter of liberty, not price. To understand the concept, you should think of “free” as in “free speech,” not as in “free beer”.
>
> (그러므로, '프리소프트웨어'는 자유의 문제이지 가격의 문제는 아닙니다. 이 개념을 이해하려면 당신은 '자유'를 '공짜맥주'가 아닌, '언론의 자유'와 같은 예를 생각해보면 됩니다.)

라고 설명을 한다. 자유소프트웨어 즉, 영어로 '프리소프트웨어'에서의 'Free'가 소스코드에서의 자유를 의미하지, 공짜로 배포하는 '프리웨어'를 설명하지 않는다고 정의한다. 그럼 자유소프트웨어에서 정의하는 자유란 무엇인가? 다음의 내용은 자유소프트웨어에서 말하는 4가지 종류의 자유이다.

1) 자유소프트웨어의 4가지 '자유'

만약 프로그램의 사용자가 다음의 4가지 필수 자유를 가지면, 그 프로그램은 자유소프트웨어이다.

자유 0) 프로그램을 어떠한 목적을 위해서도 실행할 수 있는 자유.

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

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

자유 3) 프로그램을 향상시키고 이를 공동체 전체의 이익을 위해서 다시 환원시킬 수 있는 자유. 이러한 자유를 위해서는 소스 코드에 대한 접근이 선행되어야 한다.

2) 자유소프트웨어와 오픈소스의 차이점

오픈소스 공식 홈페이지에 따르면, 자유소프트웨어와 오픈소스가 같은 자유를 언급하고 있어도 둘은 다른 가치를 내포하고 있다는 입장이다. 자유소프트웨어의 개념은 1980년대부터 시작되었으며 소프트웨어를 변형하고 복사하고 분배하고 실행할 자유 그 자체에 집중을 한다. 하지만, 오픈소스의 개념은 1997년도 부터 시작되었으며 라이선스에 의한 실용적인 결과에 집중한다. 자유소프트웨어 운동을 이끌었던 리처드 스톨만 역시, 오픈소스는 개발 방법론인데 반해, 자유소프트웨어는 사회적인 운동이라고 말한 바가 있다.

2. 오픈소스 라이선스

오픈소스 소프트웨어 또한 오픈소스 소프트웨어를 사용가능하다는 라이선스가 존재한다. OSI의 정의에 따른 모든 오픈소스 라이선스는 소스코드의 배포를 허용하며 오픈소스 소프트웨어를 받은 누구라도 코드를 보고 수정할 권리가 주어진다.

1) 오픈소스 라이선스 종류

- **GNU** 일반 공중 사용 허가서**(GNU GPL** 혹은 **GPL)**

GPL은 가장 엄격한 제약이 있는 라이선스로 카피레프트의 성격이 강하다. GPL 라이선스를 가진 프로그램을 사용하여 새로운 프로그램을 만들게 되면 파생된 프로그램 역시 같은 카피레프트를 가져야 한다. 이러한 철학에서 GPL은 컴퓨터 프로그램을 이용하는 사람에게 자유 소프트웨어의 권한을 주고 카피레프트를 사용함으로써 그 자유를 보전해주며 이전 작업 내용을 수정하거나 다른 내용을 추가하는 것도 허용한다.

- **LGPL** 라이선스

기존의 GPL의 높은 제약을 완화하기 위해 만들어졌으며 소프트웨어 라이브러리를 염두한다. 만약 LGPL로 작성된 소스코드를 라이브러리로만 사용하는 경우엔 소스코드를 공개하지 않아도 되며 그 이외 사항은 GPL과 동일하다.

- 아파치**(Apache)** 라이선스

아파치 라이선스는 아파치 소프트웨어 재단에서 만든 소프트웨어 라이선스이다. 아파치 라이선스 2.0은 GPL과는 달리 소스 코드 공개의 의무가 존재하지 않고, 2차 라이선스와 변형물의 특허 출원이 가능하다. 라이선스 적용 시 아파치 재단의 이름과 라이선스의 내용을 명시해야 하며, 아파치 라이선스 2.0이 적용된 소스 코드를 수정했을 경우 그 사실을 밝혀야 한다.

- 모질라**(Mozilla)** 퍼블릭 라이선스**(MPL)**

모질라 어플리케이션 스위트, 모질라 파이어 폭스 등 모질라 소프트웨어에 적용된다.

- **BSD license**

버클리의 캘리포니아 대학에서 배포하는 공개 소프트웨어의 라이선스. GPL보다 훨씬 개방적인 4개항의 간단한 문구로 되어 있으며 BSD의 운영체제인 FreeBSD, Django 또한 BSD 라이선스를 따르고 있다.

2) 오픈소스 라이선스 비교

	무료이용 가능	배포 가능	소스코드 수정 가능	파생 저작물 재공개 의무	독점 **SW**와 결합 가능
GPL	O	O	O	O	X
LGPL	O	O	O	O	O
MPL	O	O	O	O	O
BSD 라이선스	O	O	O	X	O
Apache 라이선스	O	O	O	X	O

(출처 : 컴퓨터프로그램 보호 위원회)

3. 오픈소스 프로젝트

대부분 오픈소스 프로젝트 구성원은 다음과 같다.

컨트리뷰터: 컨트리뷰션을 하는 모든 사람들.

커미터:컨트리뷰션의 내용을 리뷰하고 프로젝트에 반영할지 결정하는 사람

메인테이너:프로젝트 방향 설정, 관리하는 사람.

저작자:프로젝트 만든 사람 또는 조직을 말한다.

1) 오픈소스 프로젝트 활동

(1) 컨트리뷰션

컨트리뷰션은 오픈소스 프로젝트에 참여하고 기여하는 모든 활동을 말한다. 컨트리뷰선이 일반적으로 소스코드를 작성하는 것만 해당한다고 생각하는 경향이 있는데, 소스코드 뿐만 아니라 코드 테스트, 오타 수정, 번역, 가이드 문서 작성, 디자인 작업, 의견 제시 또한 컨트리뷰션이다.

(2) 개발 도구

1) 버그 트래커

버그 트래커(Bug Tracker) 혹은 이슈 트래커는 오픈 소스 개발에서 없어서는 안될 중요한 소프트웨어이다. 이는 제품의 문제점을 발견하고 해결하기 위한 과정을 시스템화 한 것으로 문제 발견, 원인 규명 ,재구현,문제 해결의 과정을 거친다.

2) 버전 컨트롤

버전 관리(version control, revision control) 란 동일한 정보에 대한 여러 버전을 관리하는 것을 말한다. "버전"을 통해서 시간적으로 변경 사항과 그 변경 사항을 작성한 작업자를 추적할 수 있다. 대표적인 오픈소스 버전 관리 시스템으로 'Git'이 있다.

- Git

git은 대표적인 '분산 버전 관리 시스템'이다. 분산 버전 관리 시스템이란 각 개발자가 중앙서버에 접속하지 않은 상태에서도 코드 작업을 할 수 있는 버전 관리 시스템을 말한다. 즉, 로컬 저장소에서 원격 저장소로 코드 작업을 할 수 있다는 것이다. 이는 다른 원격저장소나 로컬저장소와는 다른 가장 큰 특징이다.

5. 왜 오픈소스를 사용하는가**?**

오픈소스는 어떤 장점이 있기에 많은 사람들이 사용하는 것일까**?** 여기 오픈소스의 **4**가지 장점을 소개한다.

① 빠르고 유연한 개발을 할 수 있다**.**

소스코드를 공개적으로 작업하기 때문에 최신 기술 정보와 오류의 해결책을 쉽게 공유할 수 있다. 또한, 독점 소프트웨어에 비해 기술의 발전 속도가 빠르다.

② 소유권에 대한 전체적인 비용이 적다**.**

오픈소스 소프트웨어를 사용하는 것은 독점 소프트웨어와 비교해 보았을 때, 소유권에 대한 전반적인 비용이 적다. 오픈소스 소프트웨어를 채택하는 것은 일반적으로 선행비용이 적다. 왜냐하면 소프트웨어가 보통 무료이거나 값이 싸기 때문이다. 또한, 오픈소스 소프트웨어는 라이선스에 관한 비용을 청구하지 않지만, 독점 소프트웨어는 비용을 청구한다.

COST	OPEN SOURCE	PROPRIETARY
Licensing	No	Yes
Implementation	Yes	Yes
Maintenance	Yes	Yes
Support	Yes	Yes

③ 호환성이 뛰어나다**.**

오픈소스는 오픈 포맷이나 오픈 프로토콜을 사용하기 때문에 서로 다른 소프트웨어와의 연동이 쉽다. 또한 특정 운영체제나 어플리케이션 등에 종속되지 않고 자유롭게 변경할 수 있다.

④ 신뢰성과 안정성이 확보된다**.**

전 세계의 많은 개발자와 전문가들이 오픈소스 프로젝트에 참여하기 때문에 독점 소프트웨어에 비해 안정적이다.

오픈소스 역사

2018년, 올해로 20주년을 맞이한 오픈소스 소프트웨어는 현재 IT업계에서 큰 영향력을 끼치고 있다.

'오픈소스'는 어떤 과정을 거쳐 탄생하였으며 오늘날 오픈소스는 어떻게 활용되고 있는지 알아보자.

1. 컴퓨터 소프트웨어의 시작

1950 ~

컴퓨터 프로그램의 초기 사용시기로써 모든 프로그램은 공개적으로 개발되고 소스코드와 함께 버그 수정 등 새로운 기능을 추가할 수 있는 도구도 제공되었다. 소프트웨어와 하드웨어가 밀접하게 결합되어 있어서 사용자가 소프트웨어를 수정해주지 않으면 다른 기종에서 작동하지 않았기 때문이다. 이 시기의 개발자 및 사용자들은 소스코드를 자유롭게 고치며 유용한 프로그램을 만들거나 공유하기도 하였다.

1960 ~

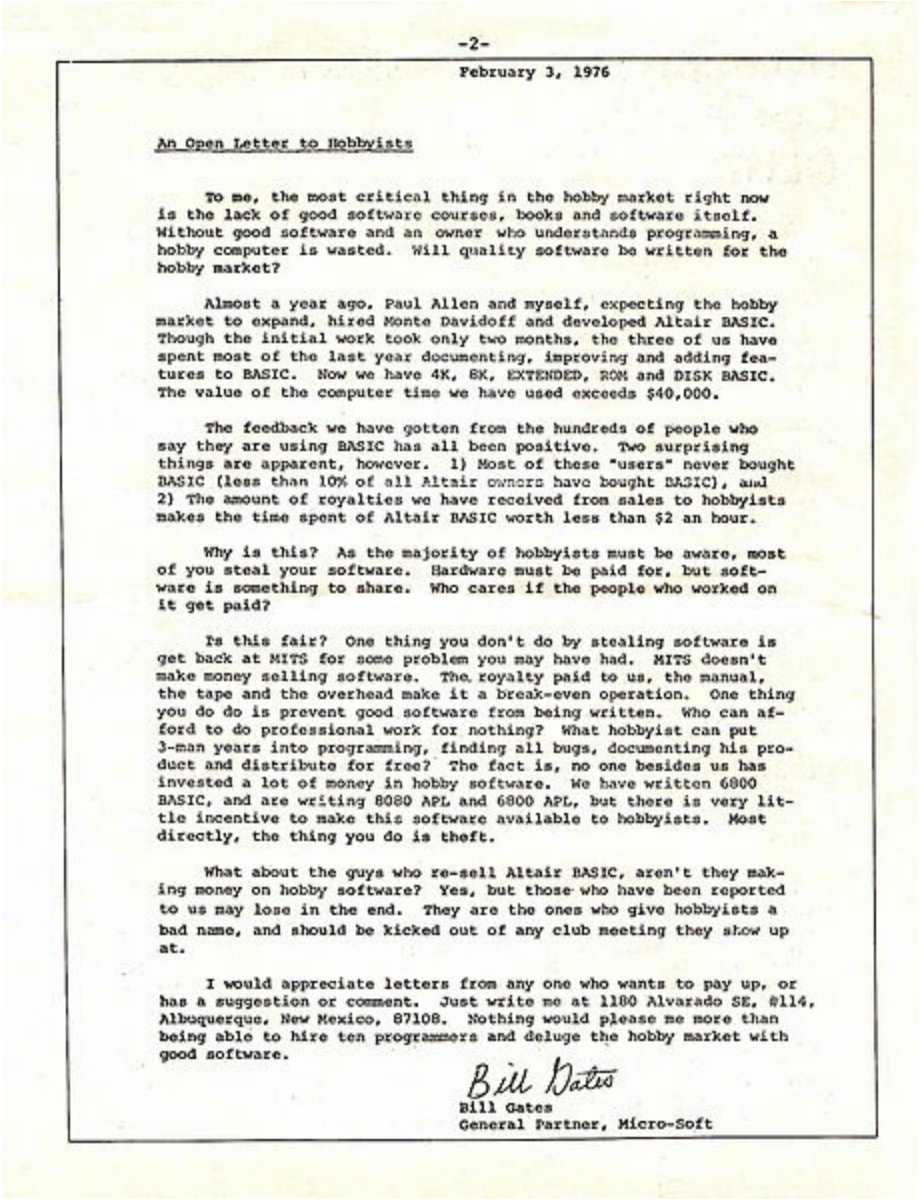
1960년대에 들어서 소스코드와 하드웨어를 함께 제공해온 IBM의 메인프레임이 시장을 점유하기 시작하자, 1969년 IBM은 컴퓨터 시장을 독점한다는 혐의로 미국정부와 싸워 패소하게 된다. IBM은 패소에 대한 방안으로 1970년 1월부터 소프트웨어와 하드웨어 각각 가격을 매기겠다고 선언을 하게 되는데, 이는 소스코드가 기업의 일급기밀이 되는 동시에 소프트웨어가 하나의 중요한 산업이 되는 시작이나 다름없었다.

한편, 다른 한쪽에서는 소스코드를 공개하는 흐름이 생기기 시작했다. 바로 1969년 AT&T의 벨 연구소에 근무하고 있던 켄 톰슨이 UNIX 운영체제를 만들어 소스코드를 배포한 것이다.

2. 소프트웨어가 상품으로 전락하다**.**

1970 ~

UNIX 소스코드는 많은 대학과 연구기관에서 자유로이 사용하고 수정할 수 있었다. 하지만 1976년도에 빌게이츠가 [2] '컴퓨터 애호가들에게 보내는 공개 서한'을 발표하면서 소프트웨어 저작권에 대한 인식이 강해졌다. 또한, UNIX의 시장성을 인지한 기업들이 UNIX 기종을 만들기 시작하면서 1984년 독점금지법과 관련하여 AT&T 분할이 이루어졌다. 그로 인해 UNIX는 상품화로 입장을 바꾸어, 무료로 배포하던 소스코드를 공개하지 않았고 가격을 올리기 시작했다. 공개되어왔던 소스코드가 기업의 소유가 되어버린 것이다.



-2-

February 3, 1976

An Open Letter to Hobbyists

To me, the most critical thing in the hobby market right now is the lack of good software courses, books and software itself. Without good software and an owner who understands programming, a hobby computer is wasted. Will quality software be written for the hobby market?

Almost a year ago, Paul Allen and myself, expecting the hobby market to expand, hired Monte Davidoff and developed Altair BASIC. Though the initial work took only two months, the three of us have spent most of the last year documenting, improving and adding features to BASIC. Now we have 4K, 8K, EXTENDED, ROM and DISK BASIC. The value of the computer time we have used exceeds $40,000.

The feedback we have gotten from the hundreds of people who say they are using BASIC has all been positive. Two surprising things are apparent, however. 1) Most of these "users" never bought BASIC (less than 10% of all Altair owners have bought BASIC), and 2) The amount of royalties we have received from sales to hobbyists makes the time spent of Altair BASIC worth less than $2 an hour.

Why is this? As the majority of hobbyists must be aware, most of you steal your software. Hardware must be paid for, but software is something to share. Who cares if the people who worked on it get paid?

Is this fair? One thing you don't do by stealing software is get back at MITS for some problem you may have had. MITS doesn't make money selling software. The royalty paid to us, the manual, the tape and the overhead make it a break-even operation. One thing you do do is prevent good software from being written. Who can afford to do professional work for nothing? What hobbyist can put 3-man years into programming, finding all bugs, documenting his product and distribute for free? The fact is, no one besides us has invested a lot of money in hobby software. We have written 6800 BASIC, and are writing 8080 APL and 6800 APL, but there is very little incentive to make this software available to hobbyists. Most directly, the thing you do is theft.

What about the guys who re-sell Altair BASIC, aren't they making money on hobby software? Yes, but those who have been reported to us may lose in the end. They are the ones who give hobbyists a bad name, and should be kicked out of any club meeting they show up at.

I would appreciate letters from any one who wants to pay up, or has a suggestion or comment. Just write me at 1180 Alvarado SE, #114, Albuquerque, New Mexico, 87108. Nothing would please me more than being able to hire ten programmers and deluge the hobby market with good software.

Bill Gates

Bill Gates
General Partner, Micro-Soft

3. 자유 소프트웨어 그리고 오픈소스

1980~

- 소프트웨어는 공유되어야 한다**.**

공유되어 왔던 소프트웨어 소스코드가 제한되고 기업의 수익을 위한 상품으로 전락하자, MIT의 리처드 스톨만(Richard Stallman)은 딜레마를 느꼈다. 소프트웨어를 사용하려면 다른 사람과 공유하지 말아야 한다는 의미가 협동적인 공동체 사회로부터 단절되어야 한다는 의미로 다가왔기 때문이다. 그래서 리처드 스톨만은 "소프트웨어는 공유되어야 한다."라는 소신을 가지고 1984년도에 GNU라는 프로젝트를 시작한다. [3] GNU프로젝트란 UNIX와 완벽하게 호환되는 자유소프트웨어 운영체제를 개발하기 위한 프로젝트다.

이를 본격적으로 촉진하기 위해서 1985년에 자유소프트웨어재단**(FSF)**를 설립하고 자유소프트웨어 운동을 펼쳐나간다. 운영체제를 만들기 위해 text editor, compiler, debugger 등을 개발하고 [3] 라이선스인 GNU General Public License를 배포하였지만 운영체제의 핵심기능인 커널을 개발하기에는 현실적으로 한계가 있음을 깨닫게 된다.

- 리눅스 커널이 탄생하다**.**

그리고 1991년, GNU 프로젝트에 가장 적합한 리눅스라는 커널이 탄생한다. 리눅스를 만든 사람은 오늘날 위대한 프로그래머로 손꼽히는 리누스 토발즈이다. 그가 핀란드 헬싱키 대학의 2학년이었던 시절, 컴퓨터를 가지고 놀면서 교육용 운영체제인 MINIX를 좀 더 잘 써보려고 만든 것이 리눅스이다. 그는 리눅스 커널 개발 후에 GNU GPL을 부여하여 GNU프로젝트에 의해 개발될 수 있도록 소스코드를 공개하였다. 마침내 LINUX(GNU/LINUX) 운영체제가 탄생되었고 LINUX 운영체제가 인정을 받으면서 배포와 기술 지원을 전문으로 하는 '레드햇'같은 회사들이 등장하게 된다. 그리고 이는 LINUX가 크게 성장할 수 있는 요건이 된다.

- '오픈소스' 이름이 탄생하다**.**

1997년 에릭 레이먼드가 '성당과 시장'이라는 논문을 발표했다. 그는 이 책에서 리눅스 커널, GCC 등을 사용하는 자유 소프트웨어 개발 방법론의 이점을 보여주었고 대형 기업인 넷스케이프가 소스코드를 공개하기로 결정한 데에 영향을 미쳤다. 자유 소프트웨어가 IT업계에 바람을 일으키자, 몇몇 사람들은 자유 소프트웨어를 사업 영역으로 가져가길 바랐다. 그들이 당면한 문제는 자유 소프트웨어에서 나타나는 '자유(free)'가 상업적인 면에서는 '공짜'로 인식될 수 있다는 점이었다. 그들은 1998년 자유소프트웨어 리더와 함께 넷스케이프 소스코드를 어떤 형태로 공개할 것인지를 정하는 전략회의에서 '오픈소스'라는 용어를 새롭게 만들었고 기업들이 소스코드 공개에 많이 참여할 수 있도록 지원했다.

> "기존에 있던 자유 소프트웨어 개념을 기업에 홍보하고 또 라이선스를 인증하기 위해 '오픈소스'가 이 운동에 적합한 이름이다" - 브루스 페렌스

오픈소스가 주목받기 시작한 초기에는 자유소프트웨어의 개념과 명확한 차이가 존재하지 않았다. 사람들마다 해석의 차이가 존재할 뿐만 아니라, 기업과 자유소프트웨어 추종자사이에 이해관계를 조절할 필요성이 생기자, 1998년 2월, 에릭 레이먼드와 브루스 페렌스가 **OSI(Open Source Initiative)**를 설립하게 된다.

(OSI의 로고에서 O는 개방성을, 열쇠구멍은 소스코드의 잠금 해제를 의미한다.)

그리고 OSI는 OSD라는 오픈소스 정의를 발표한다. 오픈소스의 정의는 데비안 자유소프트웨어 지침(Debian Free Software Guidelines)에 기반을 두고 있다.

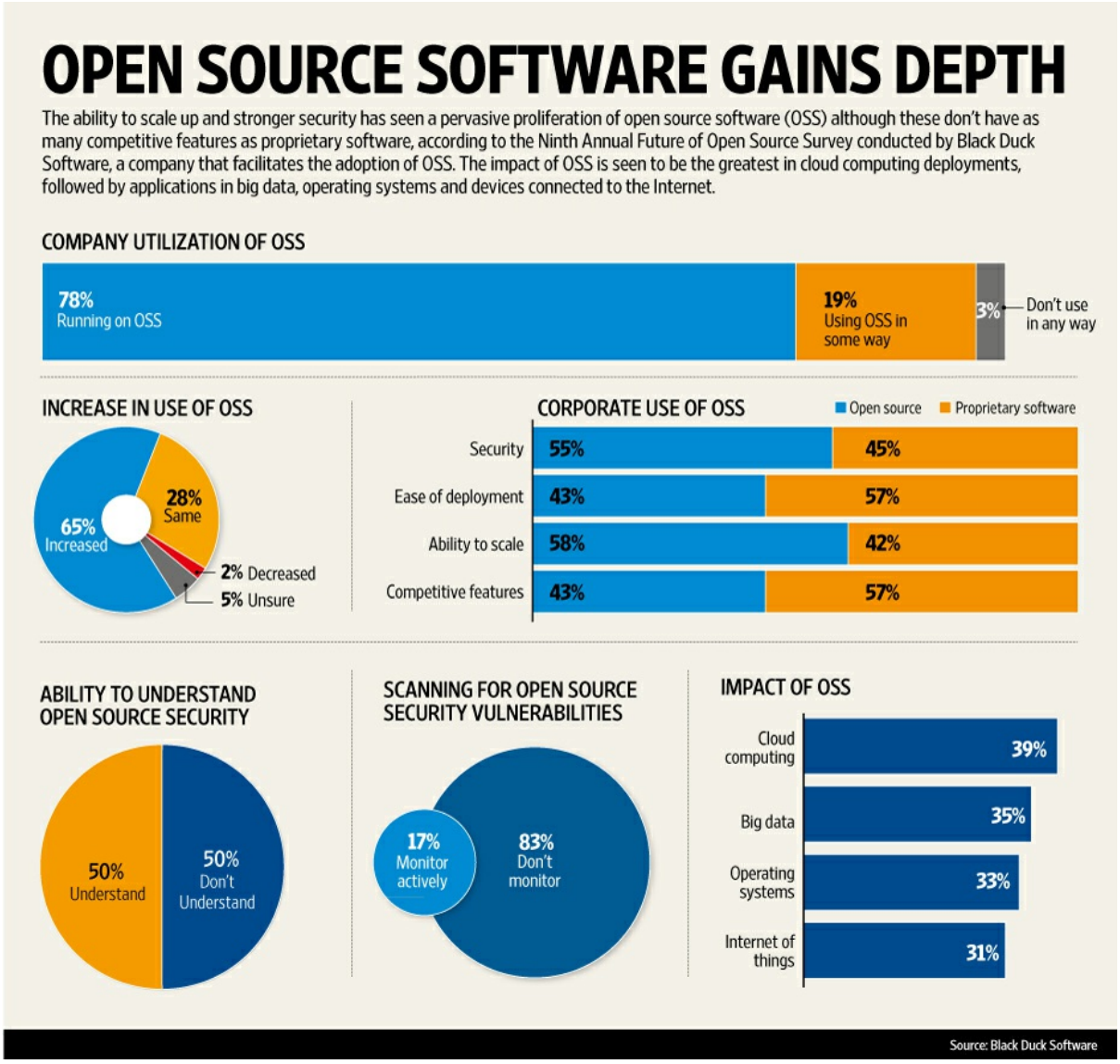
3. 그리고 **2000**년대 오픈소스

수년간 OSI는 수백개의 라이선스를 받았고, 대략 60개 정도의 라이선스를 승인 받았다. 라이선스 선택의 폭이 폭발적으로 증가한 이유는 오픈소스의 관심이 높아짐에 따라 사람들이 그들의 오픈소스 소프트웨어를 만들고 관리하기를 원하기 때문이었다. 하지만, 모든 라이선스가 소스코드를 읽고 수정하고 공유하는 것을 허용했기 때문에 라이선스의 무분별한 확산이 문제가 되기 시작했다. 결국 OSI는 2004년~2006년 기간에 이 문제를 처리하기 위해 공공 의견 수렴 과정(Report of License Proliferation Committee and draft FAQ)을 진행했고 쓸모 없는 라이선스를 정리하여 더 많이 쓰이는 라이선스를 위한 공간을 제공하기로 했다.

'리눅스는 지적 재산권 개념에 달라 붙는 암적인 존재다.' 라며 오픈소스에 대해 부정적인 평가를 내렸던 마이크로소프트도 더 이상 오픈소스를 피할 수 없게 되었다. 마이크로소프트 오픈 테크놀로지 담당 이사인 지아누고 라벨리노는 오픈소스에 대해 이렇게 말한다.

> "시장이 변했다. 2002년의 시장은 지금의 모습과 완전히 달랐다. 생물도 환경에 적응하듯, 우리도 그러했을 뿐이다."

2000년대 후반, IT시장의 주인공인 클라우드가 들어서면서 오픈소스와 클라우드는 떨어질 수 없는 관계가 되었다. 마이크로소프트도 시장의 변화에 따라 클라우드 솔루션 구축을 할 수 밖에 없었고 더 이상 오픈소스에 대한 공개적인 적대감을 드러낼 수 없었다.



(출처 : Black Dock Software)

다른 상용 소프트웨어 회사에서도 오픈소스를 사용하기 시작했다. 특히 웹 기반 서비스에서 오픈소스의 활용이 폭발적으로 증가하였는데, Github, sourceforge, Chrom등의 인터넷 브라우저의 출시로 오픈 소스 활동이 전 세계적으로 활성화 되었다. 또한 구글, 아마존, 넷플릭스, 인텔, 오라클 등 오늘날 많은 기업들이 오픈소스를 사용하고 있다.

오픈소스의 활용

이제 오픈소스 프로젝트는 클라우드 컴퓨팅, 다양한 웹서비스, 빅데이터 등의 분야로 나아가고 있다.

이번에는 어떤 회사들이 오픈소스 프로젝트에 참여하고, 어떤 방식으로 오픈소스 프로젝트를 진행하고 있는지 알아보자.

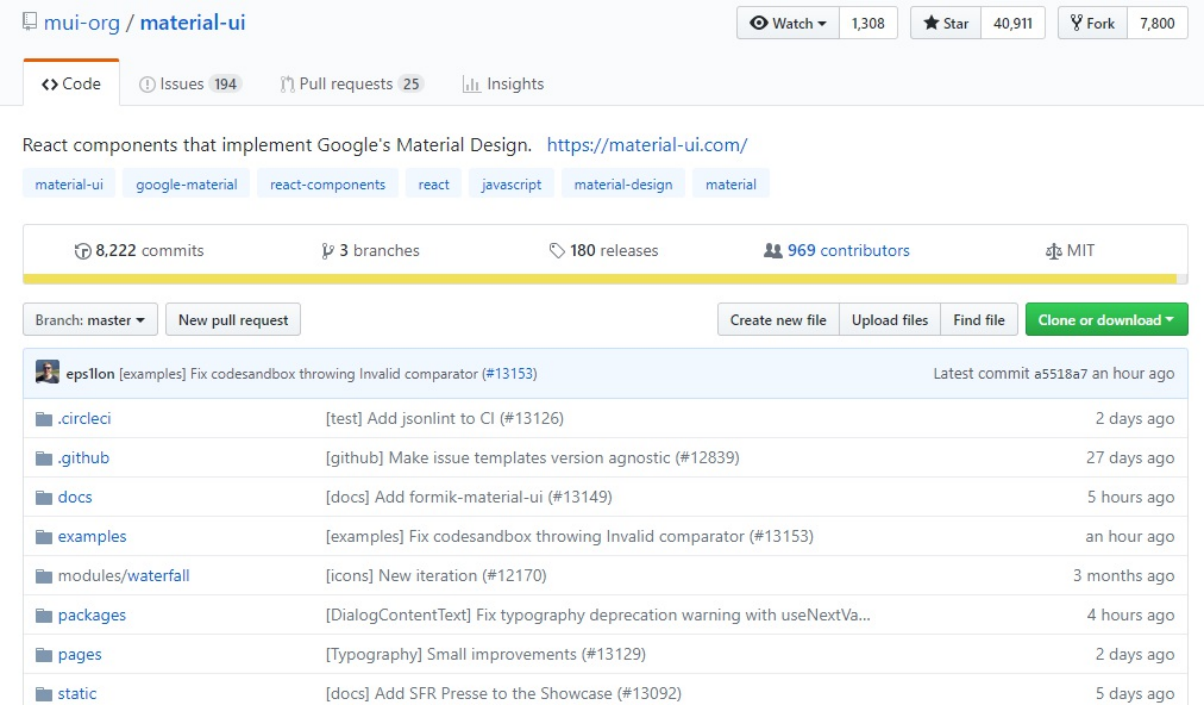
1) Netflix(넷플릭스)

Netflix는 TV 프로그램, 영화, 다큐멘터리 등 다양한 콘텐츠를 언제 어디서나 광고 없이 즐길 수 있는 스트리밍 서비스이다. 한국 콘텐츠 시장에 큰 영향력을 끼치고 있는 이 Netflix는 몇 년간 Github에 자사의 오픈 소스 센터 홈페이지를 열어서 코드를 공개하고 있다. 콘텐츠 인코딩 기술, 보안 기술, 백업·복구 기술 등이 포함되어 있으며, 최근에는 Titus라는 컨테이너 관리 플랫폼을 공개하기도 했다. Titus는 스트리밍, 추천기능, 콘텐츠 인코딩 시스템 등 Netflix에서 핵심적인 부분들을 실행시키는 기술이다.

Netflix는 Titus를 공개함으로써 자사와 생각이 비슷한 회사 및 팀들을 도울 수 있을 것이고 Netflix가 배운 교훈들을 컨테이너 관리 공동체에게 전달할 수 있을 것이라 생각하여 오픈소스를 하기로 결정했다고 한다.

2) Google (구글)

2018년 5월 기준 전 세계 검색량의 90%를 점유한 최대 검색 서비스 기업 구글은 오픈 소스 소프트웨어 업계의 가장 큰 기여자라고 할 수 있다. 안드로이드 운영체제에서부터 인공지능 기술, 웹브라우저, 프로그래밍 언어등 다양한 분야에서 오픈소스 기술에 관심을 가지고 열심히 활동하고 있다. 구글 깃허브 홈페이지에 접속하면 2000여개의 오픈소스 프로젝트를 진행하고 있는 것을 확인할 수 있다. 그중에서 개발자들에게 가장 많은 관심을 받고 있는 프로젝트가 Material -UI 이다.



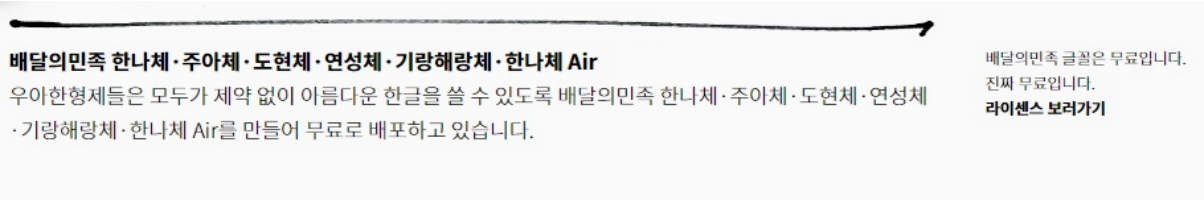
Material design은 지난 2014년에 발표한 구글의 첫 디자인 시스템이다. 종이와 잉크에서 영감을 받아 그것을 구성하는 카드, 그림자, 포인트 컬러 등을 스크린 안에 구현하였으며, 수준 미달 디자인으로 평가되던 구글 제품들과 안드로이드 애플리케이션의 디자인 수준을 높였다.

(구글이 선보인 Meterial Design이다. 출처: CODECONDO)

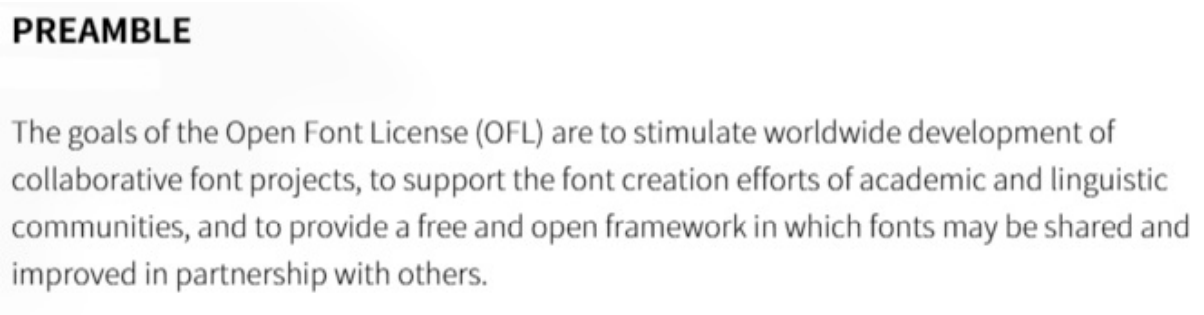
그리고, 올해 구글은 새로운 Material design과 Material design UI를 안드로이드와 크롬 OS, 웹에 적용시켰다. Material design UI는 사용자들이 Material design을 쉽게 이용할 수 있도록 하는 프로그램으로 현재 Github에서 오픈소스 프로젝트를 활발히 진행하고 있다.

3)배달의 민족 글꼴

오픈소스는 SW뿐만 아니라 다양한 분야에서도 활용된다. 대표적으로 오픈소스 글꼴이 있다. '배달의 민족'이라는 배달 서비스 앱을 개발한 우아한 형제들은 '한나는 11살체'를 비롯한 네 개의 오픈소스 글꼴을 배포하고 있다.



이 글꼴들은 **OFL** 라이선스 **(Open Font License)**하에서 개인 혹은 단체 누구든 어떤 목적으로 사용할 수 있다.

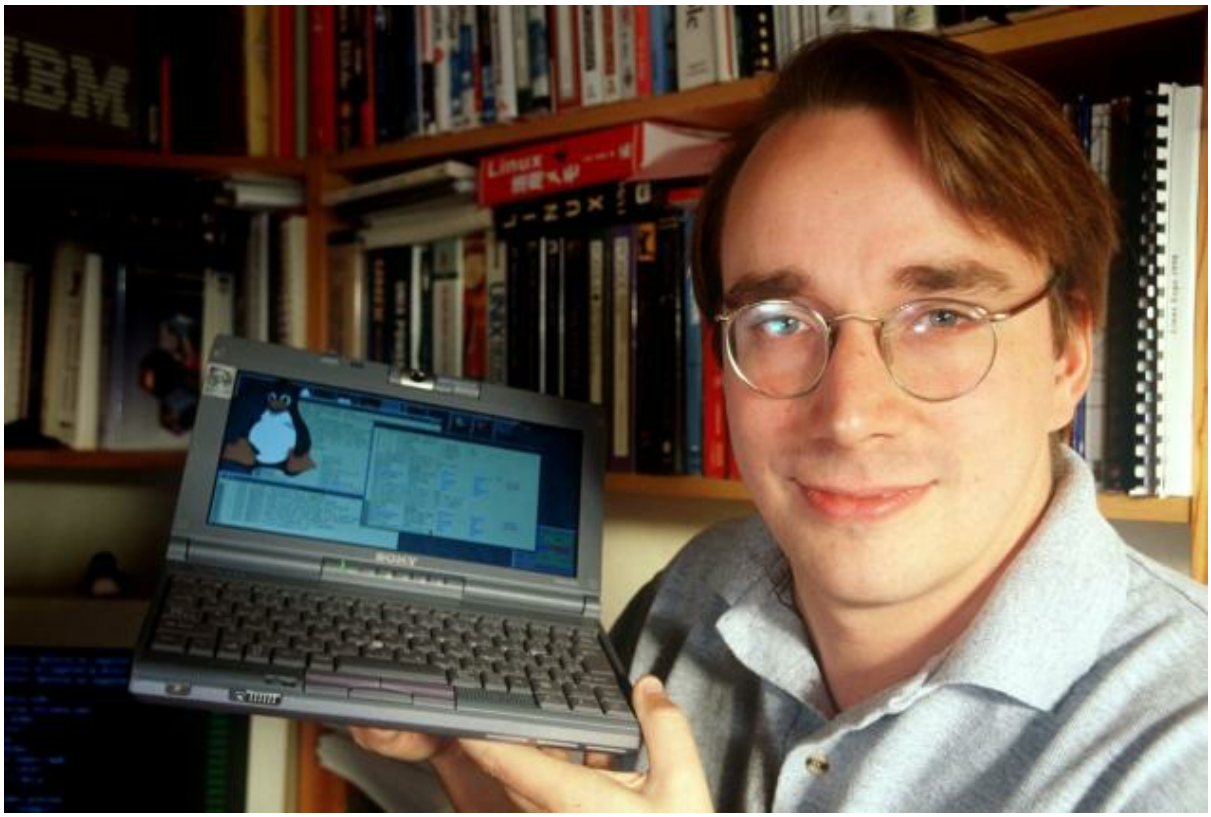


오픈소스와 자유소프트웨어의 얼굴들

자유소프트웨어 및 오픈소스에 기여한 대표 인물들을 만나볼 차례다.

그들의 업적과 근황 그리고 약간의 일화를 소개한다.

1.리누스 토발즈(Linus Benedict Torvalds)



업적

리누스 토발즈는 리눅스 이외에도 많은 업적을 남겼는데, 대표적으로 GIT이 있다.

리누스 토발즈는 초기에 리눅스 커널 소스 코드를 관리하면서 기존의 [4] 버전 관리 시스템(VCS)이 마음에 들지 않는다는 이유로 아예 사용하지 않았다. 그러다, 버전 관리에 어려움을 겪자, BitKeeper라는 프로그램을 사용하기로 결정했다. BitKeeper는 상용 소프트웨어였지만, 분산처리기능과 빠른속도가 다른 버전관리 시스템보다 괜찮았기 때문에 사용했다고 한다. 토발즈의 이런 선택은 자유 소프트웨어 단체로부터 비난을 받았으나 자유 소프트웨어 진영에서도 마땅한 대안을 제시하지는 못했다. 그러다가, BitKeeper 쪽에서 리버스 엔지니어링 문제로 리눅스에 관한 지원을 끊었고 결국, 토발즈는 자신이 직접 버전 관리 시스템을 만들기로 했다. 그렇게 만든 게 바로 Git이었다. Git은 기존의 버전관리 시스템과 다르게 저장소와 히스토리를 더불어 복제하기 때문에 서버에 문제가 생기면 이 복제물로 다시 작업을 시작할 수도 있다. 또한, 엄청나게 빨라서 대형 프로젝트에 사용하기도 좋다. 그 외에도 많은 강력한 장점들을 가진 Git은 오늘날 많은 개발자들이 빠르고 동시다발적으로 작업을 하는 데 많은 도움을 주고 있다.

근황

최근 리누스 토발즈에 관한 소식은 실로 흥미롭다. 토발즈가 이제까지 자신의 무례한 행동에 관해 사과를 한 것이다. 토발즈는 사람을 코드로 평가하기로 유명했다. 범죄자라도 코드만 좋으면 충분히 프로젝트에 합류시킬 수 있으며 만약 코드가 별로라면 인신공격을 서슴치 않았다. 그런 그가 이제까지 자신의 행동은 프로답지 못했다며 사과를 하고, 충분한 휴식기를 가지면서 프로다운 모습을 보이도록 노력하겠다고 말한 것이다. 그리고 리눅스는 새로운 CoC(Code of Conduct, 행동강령)를 받아들였는데,

index : kernel/git/torvalds/linux.git

Linux kernel source tree

about summary refs log tree commit diff stats

path: root/Documentation/process/code-of-conduct.rst

blob: ab7c24b5478c6b30adad49ad2f857ee358435173 (plain)

```
Contributor Covenant Code of Conduct
++++++++++++++++++++++++++++++++++++

Our Pledge
========

In the interest of fostering an open and welcoming environment, we as
contributors and maintainers pledge to making participation in our project and
our community a harassment-free experience for everyone, regardless of age, body
size, disability, ethnicity, sex characteristics, gender identity and
expression, level of experience, education, socio-economic status, nationality,
personal appearance, race, religion, or sexual identity and orientation.

Our Standards
==========

Examples of behavior that contributes to creating a positive environment
include:

* Using welcoming and inclusive language
* Being respectful of differing viewpoints and experiences
* Gracefully accepting constructive criticism
* Focusing on what is best for the community
* Showing empathy towards other community members


Examples of unacceptable behavior by participants include:

* The use of sexualized language or imagery and unwelcome sexual attention or
  advances
* Trolling, insulting/derogatory comments, and personal or political attacks
* Public or private harassment
* Publishing others' private information, such as a physical or electronic
```

현재 몇몇 리눅스 개발자들은 이 행동강령을 보고 코드로만 평가를 받아야할 개발자들이 코드 이외의 태도에 관해서도 신경을 써야하는 것이냐며 반발을 하고 있다.

2.리차드 스톨만**(Richard Matthew Stallman)**

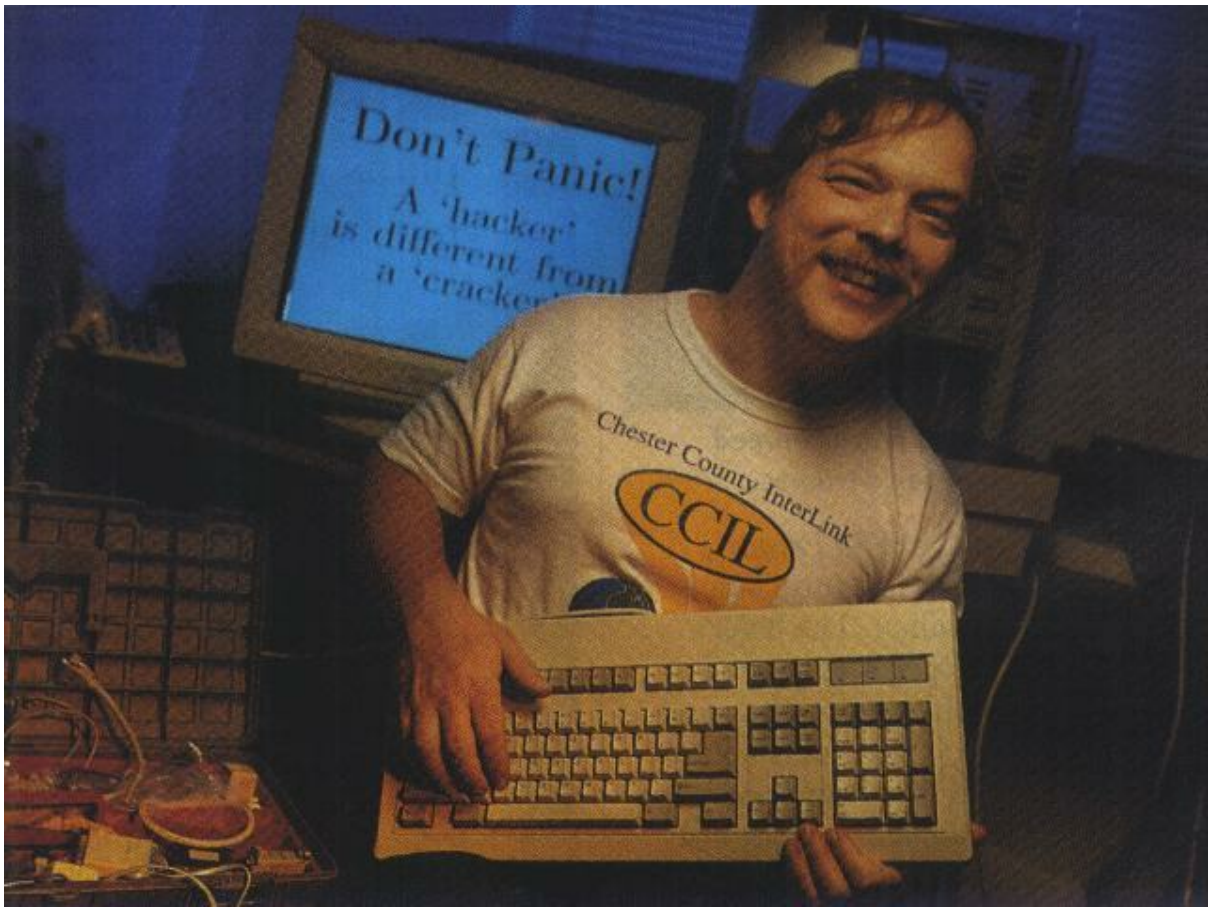
업적

자유소프트웨어 운동의 선구자이며 프로그래머이기도 하다. GNU 프로젝트와 자유 소프트웨어 재단을 설립하였고 GNU프로젝트를 진행하는 동안에는 [5] GNU 컴파일러 모음(GCC), GNU Emacs등을 개발했고 '카피레프트'의 개념과 GNU 일반 공중 허가서를 도입했다.

일화

리처드 스톨만이 MIT 공대의 인공지능 실험실의 해커로 활동하고 있었을 때였다. 실험실 동료들은 프린터기의 소스코드를 마음대로 사용해서 프린터기를 종종 고장내곤 했다. 동료 중 누군가 프린터기가 오류 메세지를 보내도록 프로그램을 수정해 놓으면, 또다른 누군가가 와서 그 오류를 고치는 일종의 장난질을 했던 것이다. 하지만, 이 장난질은 오래가지 못했는데, 실험실에 Xerox라는 새로운 프린터기가 설치되었기 때문이다. Xerox 프린터기의 소스코드는 패키지 안에 포함되지 않았고 그들이 예전 프린터기에서 사용했던 유지 관리와 같은 것을 설치할 수 없었다. 후에 리처드 스톨만은 다른 실험실의 누군가가 Xerox의 소스코드를 가지고 있다는 것을 듣고 소스코드를 얻기 위해 노력한다. 마침내, 그가 소스코드를 얻었을 때, 어쩌면 그의 인생을 바꿔버리는 결정적인 한마디를 듣게 된다. 프린터기의 소스코드를 받았음을 비밀로 해야 하며 다른 사람에게 줘서는 안된다고 한 것이다. 스톨만은 이런 이기적인 행동에 화가 났고 독점 시스템의 위험성에 대해 깨닫기 시작한다. 그리고 곧 자유소프트웨어 운동을 하기로 결심하게 된다.

3.에릭 레이먼드(Eric Steven Raymond)



업적

에릭 레이먼드는 인류학자이자 오픈 소스 운동의 대표 인물이다. 그는 오픈 소스 개발과정이 어떤 방식인지, 오픈 소스 방식을 취하면서 어떻게 고품질의 소프트웨어를 만들 수 있는지를 철학적인 관점에서 고찰하여 '성당과 시장'이라는 논문을 발표하였다.

성당과 시장

에릭 레이먼드는 두 가지 방식의 자유 소프트웨어 개발 모델을 대조한다.

성당 모델: 출시 때에만 소스 코드를 공개하고 그 사이에는 제한된 개발자들만 소스코드에 접근 할 수 있는 폐쇄적 개발 스타일로, 소규모의 프로젝트 그룹이 체계적이고 권위적인 방법으로 운영된다. 릴리즈 간격은 매우 길다.

시장 모델: 소스 코드가 인터넷으로 일반에 공개된 상태로 개발된다. 프로젝트 외부의 사람들로부터 끊임없이 피드백을 요구하며 상당히 집중적인 상호 검토 과정을 거친다. 릴리즈 간격은 매우 짧다.

코드가 공개되어 있으면 많은 사람들이 테스트하고 훑어볼 수 있어서 버그는 빨리 잡힐 것이고 이에 반해, 성당 모델에서는 소스코드를 관련 개발자들만 볼 수 있으므로 버그를 잡는데 시간이 걸릴 것임을 주장하고 있다. (리누스의 법칙)

4.이언 머독 **(Ian Murdock)**

업적

이언 머독은 리눅스 배포판 중 하나인 [6]Debian GNU/LINUX(데비안)를 창시한 개발자이며 자유소프트웨어의 가장 충직한 [7] 지지자였다. 데비안의 가장 큰 장점이라 하면, '안정성'이다. 국제 우주 정거장에 쓰이는 수많은 윈도우XP 노트북을 대체하기위해 데비안을 선택할 정도로 데비안의 안정성은 크게 알려져있다. 데비안 deb 패키지 저장소는 Experimental(실험)- Unstable(불안정) - Testing(테스팅) - Stable(안정)의 단계를 가지며 testing버전은 차기 배포판을 만들기 위한 준비 공간으로 testing버전의 버그 수가 일정한 수 이하가 되면 stable 버전으로 올라간다. 이런 배포판 세부구조의 엄격한 관리는 데비안 배포판의 안정성을 뒷받침하게 된다. 또한 데비안은 Debian Social Contract(DSC)를 만들어 데비안 자유소프트웨어 지침(DFSG)을 만들었는데, 이는 자유 소프트웨어 공동체가 오픈소스의 정의의 기초로 채택한 것이다.

각주

내용주

[1] 여기서 빌게이츠는 소프트웨어도 하드웨어처럼 값을 지불해서 사용해야한다는 '소프트웨어 단독 제품' 모델을 제시한다.

[2] GNU라는 이름은 "GNU's Not Unix"라는 의미를 가지고 있다. 당시 리차드 스톨만은 유닉스 운영체제와 비슷한 시스템을 개발하고 있었는데 완전히 유닉스 운영체제는 아니었기 때문에 이런 이름을 붙였다고 한다.

[3] GNU라이선스는 자유 소프트웨어 재단에서 만든 자유 소프트웨어 라이선스로 GPL혹은 GNU GPL로 불린다.

[4]파일의 변화를 시간에 따라 기록하여 과거 특정 시점의 버전을 불러올 수 있는 시스템을 말한다.

[5] 컴파일러란 특정 프로그래밍 언어로 쓰여 있는 문서를 다른 프로그래밍 언어로 옮기는 프로그램을 부른다.

[6] Debian이라는 이름은 이안(Ian)머독과 그의 부인인 데브라(Debra)의 이름을 합쳐 만든 이름이다.

[7] 이안 머독은 2015년 12월 26일 밤과 다음날 새벽에 폭력 행위로 경찰에 연행된 후, 보석금을 지불하고 석방되었다. 그리고 28일, 자살을 예고하는 글을 남긴 후 숨진 채 발견되었다.

참조주

[1] 'Doesn't "open source" just mean something is free of charge?',

URL :https://opensource.com/resources/what-open-source

[2] '"무료 비용"은 오픈소스의 전반적인 가치의 부산물입니다.'

URL :https://opensource.guide/ko/starting-a-project/

[3] 'What's the difference between open source software and free software?'

URL : https://opensource.com/article/17/11/open-source-or-free-software

[4] IBM이 메인 프레임을 장악하다.

한국소프트웨어진흥원,오픈소스 소프트웨어 연구 보고서(2002.12),3쪽

[5]'마이크로소프트가 오픈소스를 받아들인 진짜 이유',Paul Rubens, URL :http://www.ciokorea.com/news/22074#csidx7dc8123d1a901a3b91c6efdf13dcf61

[6] '구글이 2018년 기준으로 검색량의 90%를 점유하였다. '

URL : http://gs.statcounter.com/search-engine-market-share

참고 문헌

참고 문헌

오픈소스 개발방법론 (편저):윤석찬

소프트웨어와 리걸 프레임, 10가지 이슈 (공)저: 김윤명, 이민영

소프트웨어 문화개론 (공)저: 최정호

참고 외부링크

[1] https://opensource.org/

[2] https://git-scm.com/

[3] http://www.unterstein.net/su/docs/CathBaz.pdf

[4]https://youtu.be/4ZHloJVhcRY