

Introduction

 [Edit on GitHub](#)

오픈소스의 소개와 역사

목차

[Edit on GitHub](#)

오픈소스의 소개

1. 오픈소스의 정의
 - 오픈소스란?
 - 자유 소프트웨어란?
 - 오픈소스와 자유소프트웨어의 차이점
2. 오픈소스의 라이선스
 - 오픈소스 라이선스란?
 - 오픈소스 라이선스 비교
3. 오픈소스의 장단점
4. 오픈소스의 사용추세 및 전망

오픈소스의 역사

1. 1980년대 이전의 역사
2. 1980년대의 역사
3. 1990년대의 역사
4. 현대
 - GIT
 - Android
 - Chrome OS
 - 주요인물


참고문헌

오픈소스의 소개

 [Edit on GitHub](#)

1. 오픈소스의 정의
 - 오픈소스란?
 - 자유 소프트웨어란?
 - 오픈소스와 자유소프트웨어의 차이점
2. 오픈소스의 라이선스
 - 오픈소스 라이선스란?
 - 오픈소스 라이선스 비교
3. 오픈소스의 장단점
4. 오픈소스의 사용추세 및 전망

오픈소스의 정의

 [Edit on GitHub](#)

오픈소스란?

[Edit on GitHub](#)

Open Source Software(OSS)란?



자유 소프트웨어의 문제점을 인식한 에릭 레이먼드, 부르스 페런스 등은 '오픈소스(Open Source)'라는 새로운 용어를 제안했습니다.

오픈 소스 소프트웨어란 소스 코드를 공개해 누구나 특별한 **제한 없이** 그 코드를 **보고 사용할 수 있는** 오픈 소스 **라이선스**를 만족하는 소프트웨어를 말합니다. 통상 간략하게 오픈소스 라고 말하기도 합니다. 소프트웨어 말고도 개발 과정이나 설계도가 공개되는 경우에 하드웨어에도 오픈소스 모델이 적용될 수 있으며, 글꼴과 같은 데이터에도 오픈소스 개발 모델이 적용되는 경우가 있습니다.

대부분의 오픈 소스 소프트웨어는 무료로 사용 가능 하기 때문에 프리웨어와 헷갈리는 경우가 많지만, **프리웨어는 무료로 사용 가능한 프로그램**이고, **오픈소스는 소스코드가 공개된 프로그램**이기 때문에 엄연히 다른 개념입니다.

소스가 공개되어 있고, 이를 마음껏 개조해 사용할 수 있다는 점에서 개발에 필요한 방향으로 최적화가 용이하기 때문에 일반 개인사용 자보개발자 사이에서 강세를 보였고, 서버 운영 체제에서 **리눅스**가 그 대표적인 예 입니다.

오픈 소스 소프트웨어의 종류로는 Linux, Apache, MySQL, PostgreSQL, HaqlDB, james, sendmail, JBoss, Tomcat, Geronimo 등등이 있습니다.

OSI(Open Source Initiative)로 시작한 오픈소스

에릭레이먼드, 브루스 페런스 등은 기업들이 소스코드 공개에 보다 많이 참여할 수 있도록 지원했습니다. 동시에 1998년 '오픈소스 이니셔티브'(Open Source Initiative, OSI)를 설립하고 오픈소스에 해당하는 라이선스의 최소한의 기준을 정의(Open Source Definition, OSD)해 놓았습니다. OSI는 이 정의에 따라 인증, 관리 및 촉진시키는 일을 진행하고 있습니다. 현재 업계에선 자유 소프트웨어와 오픈소스를 혼용해서 쓰는 경우가 많지만, 일반적으로 오픈소스 소프트웨어는 자유 소프트웨어를 포함한 넓은 의미로 사용됩니다.

자유소프트웨어란?

[Edit on GitHub](#)

Free Software란?



자유 소프트웨어는 복사와 사용, 연구, 수정, 배포 등의 **제한이 없는 소프트웨어** 혹은 그 통칭입니다. 소프트웨어의 수정 및 수정본의 재배포는 인간이 해독 가능한 프로그램의 소스코드가 있어야만 가능하며, 소스코드는 GPL 등의 라이선스를 통하거나, 혹은 드물게 퍼블릭 도메인으로 공개되기도 합니다. 자유 소프트웨어 운동은 초창기의 컴퓨터 사용자들이 이러한 자유를 누릴 수 있도록 하기 위해서 **1983**년에 시작 되었습니다.

자유 소프트웨어는 완전히 무료로 또는 최소한의 금액만을 받고 자유롭게 배포되어야 하며 자유 소프트웨어를 통한 비즈니스 모델들은 대개 고객 지원이나 커스터마이징 등을 통한 것들입니다. 자유 소프트웨어의 경제적 가능성은 **IBM**이나 **레드햇**, 썬 마이크로시스템즈등의 거대 회사들에 의해 인식되었습니다.

주력 산업이 IT 영역이 아닌 많은 회사들이 인터넷의 홍보 및 판매 사이트를 위해 비용이 적게 들고 애플리케이션을 쉽게 수정할 수 있다는 점에서 자유 소프트웨어를 선택했고, 또한 소프트웨어 이외의 산업에서도 그 연구와 개발을 위해서 자유 소프트웨어의 개발과 유사한 방법을 사용하기 시작했습니다.

자유소프트웨어의 종류로는 **7zip**(LGPL, 압축프로그램), **Emacs**(GPLv3), **FileZilla**(GPL), **FreeBSD**(BSD), **GIMP**(GPL, 그래픽 프로그램), **OpenTTD**(GPLv2), **ReactOS**, **Vim**(GPL + Charityware(우간다 기아 구제 프로젝트 참여 조건)), **VirtualBOX**(GPL), **삼바**(GPLv3), **송버드**, **썬더버드**, **오픈오피스**(LGPL, 오피스 프로그램), **오픈솔라리스**, **오픈스텝**, **파이어폭스**(LGPL, 웹브라우저), **하이쿠** 등이 있습니다.

GNU로 시작한 자유소프트웨어재단

1980년 PC및 소프트웨어 산업이 발전하고 관련 기업들이 생겨나며 소프트웨어에 특허나 독점에 관한 법률을 적용하는 경우가 늘어났습니다. 리처드 스톨만은 자신이 경험했던 소프트웨어 공유 문화를 되살리고 싶었습니다. 이를 위해 스톨만은 먼저 프로그램을 만들기로 했습니다. 그렇게 시작한 것이 'GNU'프로젝트입니다.

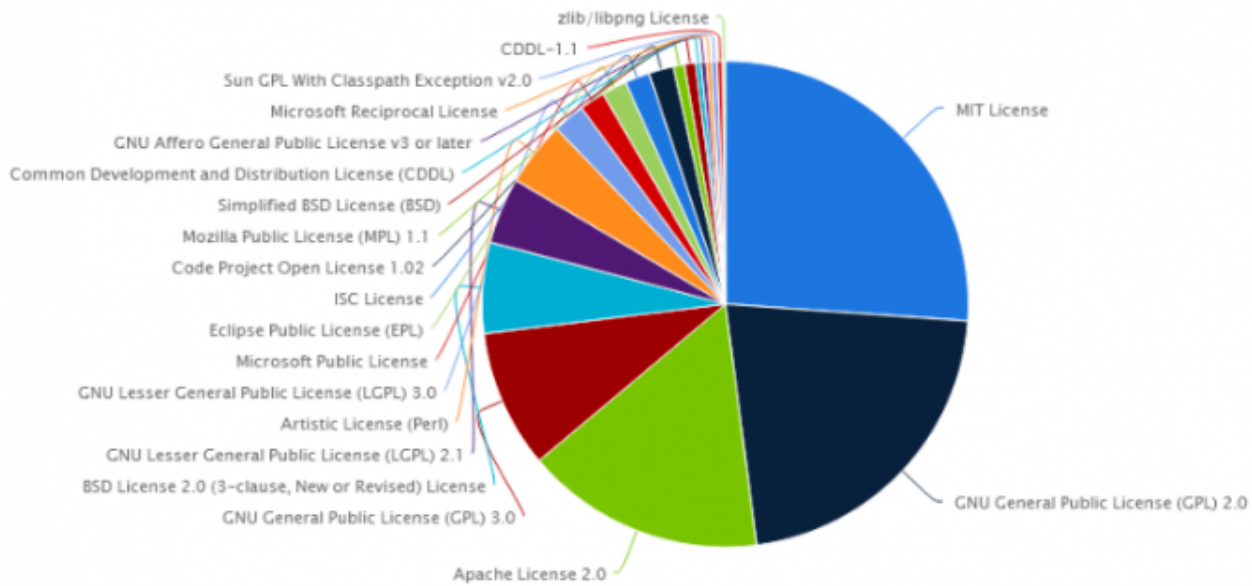
GNU 는 유닉스와 호환되는 운영체제입니다. 리처드 스톨만은 소프트웨어 공유 운동을 시작하기 위해 가장 먼저 필요한게 운영체제라고 생각했습니다.

리처드 스톨만이 '유닉스' 계열 운영체제를 만들기로 한 이유는 기존 유닉스 사용자들이 새로운 시스템에 쉽게 적응할 수 있으리라고 생각했기 때문이었습니다. GNU라는 이름은 'GNU는 유닉스가 아니다'라는 문장의 첫 글자를 따서 만든 약어입니다. GNU는 유닉스와 호환되도록 만들어진 운영체제이기는 하지만, 유닉스와는 다른 운영체제라는 의미를 내포하기 위해 만든 이름입니다.

자유소프트웨어재단

리처드 스톨만은 1985년 GNU 프로젝트를 지원하기 위한 자유소프트웨어재단을 설립합니다. 그러면서 운영체제를 만드는 것 외에 라이선스도 함께 만들어 공유 운동을 펼쳤습니다. 이렇게 만든 대표 라이선스가 'GNU GPL(GENERAL PUBLIC LICENSE, 일반 공중 사용 허가서)'입니다. 이를 줄여서 'GPL'이라고 표현하기도 합니다.

Top 20 Most Commonly Used Licenses in Open Source Projects



GPL는 지켜야 할 조항이 많은 라이선스였는데, 자유소프트웨어재단은 이를 조금씩 완화하면서 GPL 버전1, 버전3 식으로 새롭게 라이선스를 수정해 공개했습니다. 또한 누구나 GPL 위반사항을 발견하면 자유소프트웨어재단에 신고할 수 있게 했습니다. 자유소프트웨어재단은 이런 식으로 소프트웨어 공유 문화에 도움이 되도록 라이선스를 발전시키는 식으로 GPL 문화를 퍼뜨리는 운동을 진행하고 있습니다. 현재 GPL은 전체 오픈소스 가운데 2번째로 많이 사용되고 있습니다.

자유소프트웨어의 문제점

자유 소프트웨어가 등장하고 시간이 지나며 몇 가지 단점이 지적되기 시작했습니다. 자유 소프트웨어는 제약이 많은 GPL 조항 때문에 상용 기술 개발에서 활용할 수 없다는 지적이 대표적입니다. 그렇게 등장한 것이 오픈소스입니다.

오픈소스와 자유소프트웨어의 차이점

[Edit on GitHub](#)

오픈 소스 소프트웨어와 자유 소프트웨어 둘다 모두 자유를 추구하지만 둘 사이에는 자유에대한 철학의 차이가 있습니다.

오픈 소스 소프트웨어에서 추구하는 자유는 소프트웨어를 (재)사용하고 적응시키기위한 **실질적인 자유**를 원하며 자유 소프트웨어에서 추구하는 자유는 **지적 자유**를 원합니다.

오픈 소스 소프트웨어

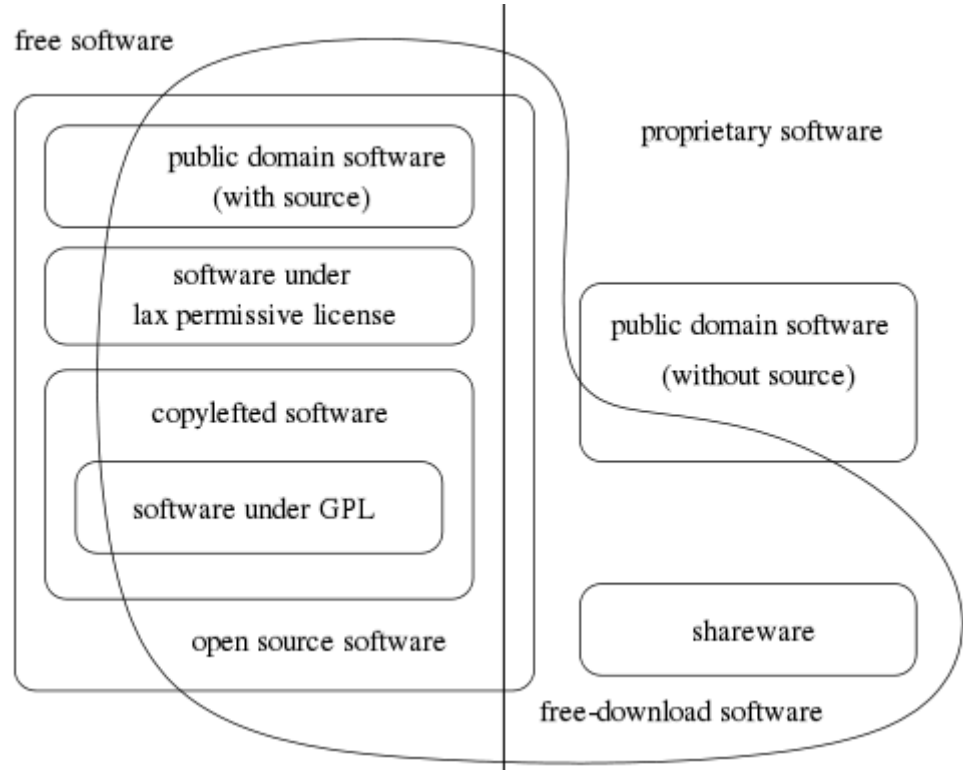
자유 소프트웨어

소스의 형태 자체를 중시

자유 소프트웨어 재단과 GNU 프로젝트와 관련된 소프트웨어에서 자유를 중시하는 의미에서 사용하는 말

자유소프트웨어재단은 오픈소스 소프트웨어라는 용어에는 '자유롭게 사용할 수 있는 권리'가 포함돼 있지 않다고 보고 자유 소프트웨어라는 용어를 사용하기를 권장합니다. 그렇다고 두 문화가 서로 적대적인 관계는 아닙니다.

GNU 프로젝트 공식 홈페이지에서는 "자유 소프트웨어 운동과 오픈소스 운동은 공동체에 있어서 두 개의 정당과도 같다"라며 "자유 소프트웨어 운동과 오픈소스 운동은 기본 원칙에 대해서 의견을 달리하지만, 모든 현실적인 방안에 대해서는 같은 생각을 갖고 있으며 세부적인 프로젝트에서 같이 협력하고 있다"라고 설명하고 있습니다. 특히 "자유 소프트웨어 운동에 있어서 우리는 오픈소스 운동을 적이라고 생각하지 않는다"라며 "우리의 적은 독점 소프트웨어"라고 강조합니다.



오픈소스 라이선스

[Edit on GitHub](#)

흔히 오픈 소스에는 저작권이 없을 것이라고 오해하는 경우가 있는데 독점적인 권리가 없을 뿐이지, 저작권은 존재합니다. 예로써, 오픈 소스 프로그램은 보통 일정한 라이선스로 배포합니다. 이 중 가장 강제성이 강한 것이 GPL인데, GPL은 오픈 소스인 만큼 소스 코드를 자유롭게 수정하고 배포할 수 있는 권리를 보장하지만, 재귀적 전염성 조항이라는 특이적인 성질이 부여됩니다. 이는 GPL 소스로부터 파생된 프로그램도 소스가 공개되어야 하고, 파생된 프로그램 또한 강제로 GPL 라이선스로 배포되어야 한다는 뜻입니다.

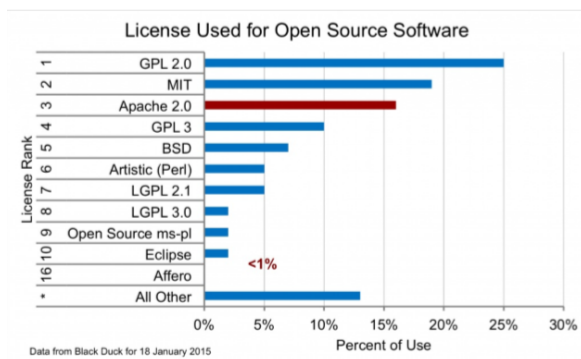
그래서 개념 없는 개발자가 생각 없이 GPL 라이선스를 따르는 오픈 소스 코드를 집어넣어서 프로그램을 만들면 그 프로그램 전체의 코드가 강제 공개 당합니다. 투하트 2, 스카이프 등이 소스 코드를 강제 공개 형벌을 받은 바 있고 이코는 영구 판매중지되었습니다. 뒷날에 플레이스테이션 3 용 게임으로 재발매 되었는데, 재발매 본에서는 GPL 소스를 전부 들어냈습니다. 소니 에서는 GPL 위반사항 때문에 DVD 4천만 장을 리콜 한 적이 있습니다. 컴파일된 프로그램인데 GPL 위반을 어떻게 아는가는 GPL 코드를 사용했는지 안 했는지 자유 소프트웨어 재단에서 이일을 하기 때문입니다.

오픈소스 라이선스란?

[Edit on GitHub](#)

오픈소스 소프트웨어 라이선스란 오픈소스 소프트웨어 개발자와 이용자간에 이용방법 및 조건의 범위를 명시한 계약입니다. 따라서, 오픈소스 소프트웨어를 이용하기 위해서는 개발자가 규정한 라이선스를 지켜야하며, 이를 위반할 경우에는 라이선스위반 및 저작권침해가 발생하고, 이에대한 책임을 지게됩니다. 이런 오픈소스 소프트웨어 라이선스는 기본적으로 이용자의 자유로운 사용을 보장하고 있습니다. 오픈소스 소프트웨어가 이와같은 라이선스를 만들어서 운영하는 이유는 오픈소스 소프트웨어를 이용하여 개발한 소프트웨어에 대해서도 법의 테두리안에서 소스코드를 공개하도록 하기 위한 것입니다. 2017년 05월 현재 오픈소스 소프트웨어 라이선스의 인증을 관장하고있는 OSI 에 따르면 78개가 있습니다. 하지만 실제로 많이 사용되는 라이선스의 개수는 한정되어 있습니다. 오픈소스 프로젝트 개발 포털사이트인 Freshmeat에 등록된 프로젝트 약 43,722개 중 약 72%가 GPL과 LGPL라이선스 입니다.

오픈소스 라이선스 사용률(2015년 기준)



오픈소스 라이선스 비교

[Edit on GitHub](#)

라이선스의 취지는 자신의 창작물을 공유하고 싶은데, 법적인 해석이 모호해지거나 의도된 바와 다르게 사용될 우려로 그러지 못하는 경우를 방지할 수 있도록 하기 위함입니다. 또한 저작권법의 철퇴를 맞을 우려없이 사람들이 공유된 저작물을 사용할 수 있도록 해주는 기능도 합니다. 간단히 말해서 원저작자가 설정한 조건만 지키면 저작권법에 걸릴일이 없다는 것입니다.

정의 규정 및 조건에 따라 사용 수정 및 또는 공유할 수 있도록 허용되는 물건들의 라이선스 종류입니다. 아래와 같은 종류가 있고 제약 정도에따라 사용되는 곳이 다릅니다.

라이선스 종류	제약 정도	비고	적용 사례
MIT	하	라이선스 및 저작권을 명시해야 합니다. 수정하더라도 공개 의무는 없습니다.	부트스트랩, Angular.js, Backbone.js, jQuery
BSD	하	라이선스 및 저작권을 명시해야 합니다. 수정하더라도 공개 의무는 없습니다.	Nginx
아파치 (Apache)	하	라이선스 및 저작권을 명시해야 합니다. 수정하더라도 공개 의무는 없습니다.	안드로이드(v2.0), 하둡(v2.0)
MPL(Mozilla Public License)	중	단순 사용 시 공개 의무는 없습니다. 다만 코드를 수정할 때 MPL로 공개해야 합니다.	모질라 파이어폭스(v1.1), 모질라 썬더버드(v1.1)
이클립스 (Eclipse)	중	단순 사용 시 공개 의무는 없습니다. 다만 코드를 수정할 때 Eclipse로 공개해야 합니다.	이클립스(v1.0)
GPL	상	사용하거나 수정을 해서 배포할 경우 반드시 소스 코드를 GPL로 공개해야 합니다.	리눅스 커널(v2.0), 깃(v2.0), 모질라 파이어폭스(v2.0), 마리아 DB(v2.0)
LGPL	중	단순히 사용만 할 경우 소스코드를 공개하지 않아도 됩니다. 하지만 배포할 때 함께 배포하면 안 됩니다. 따라서 사용자가 추가적으로 설치할 수 있도록 유도해야 합니다.	모질라 파이어폭스(v2.1)
AGPL	최상	네트워크로 사용하는 사용자가 있다고 해도 모든 소스코드를 AGPL로 공개해야 합니다.	몽고DB(v3.0)

오픈소스의 장단점

[Edit on GitHub](#)

오픈소스의 장점

• 비용절감

오픈소스 소프트웨어는 프리웨어 소프트웨어(만든이가 대가를 바라지 않거나 기타 까닭에 따라 무료로 쓰도록 제작한 소프트웨어)와 마찬가지로 무료 이용이 가능합니다. 여기서 더 나아가 소스코드가 공개 되어 직접 소프트웨어의 개선 또는 수정이 가능해지며, 이로 인해 개발 비용이 적게 드는 편입니다. 실제로 오픈소스는 무료 다운로드와 수정 / 재배포가 가능하여 초기 개발 비용이 새 소프트웨어를 개발하는 것의 절반 정도 되는 것으로 알려져 있습니다.

• 빠르고 유연한 개발

오픈소스 커뮤니티는 다양한 이용자들에게서 최신 기술 정보와 문제점의 해결책을 공유하여 운영되기 때문에 독점 프로그램에 비해 기술의 발전 속도가 빠른 편입니다. 특히 개발자와 사용자가 일치하는 경우 클로즈드 소스 프로그램(사유 소프트웨어 - 저작권 소유자의 예외적 법적 권한 하에 허가된 컴퓨터 소프트웨어입니다. 독점 소프트웨어라고도 합니다.) 보다 뛰어난 고품질의 오픈소스 소프트웨어가 개발되기도 합니다.

• 호환성 / 유연성

오픈소스는 주로 오픈포맷 또는 오픈프로토콜(개방형 표준)을 사용하기 때문에 서로 다른 소프트웨어간의 연동이 쉽습니다. 서로다른 플랫폼끼리의 상호연동 또한 가능합니다. 또한 특정 기기, 운영체제, 어플리케이션에 종속되지 않고 자유로운 변경이 가능합니다. 여러 기기들이 네트워크를 통해 하나로 연결되는 유비쿼터스 시대에 아주 적합한 장점이라고 할 수 있습니다.

• 신뢰성 / 안정성

전 세계의 수 많은 개발자들과 전문가들이 오픈소스의 개발에 참여하기 때문에 폐쇄적으로 개발되는 독점 프로그램에 비해 안정적으로 작동합니다. 단, 이는 많은 개발자들이 적극적으로 참여하는

프로그램의 경우에만 가능하므로 해당 오픈소스의 평판과 개발 과정을 주의 깊게 볼 필요가 있습니다.

오픈소스의 단점

- 강제 소스 공개와 전염성 조항이 있는 GPL라이선스에 해당되는 것인데, 유료 소프트웨어에 사용하기 어렵습니다.

다만 소프트웨어 시장 대부분이 SI인데, GPL 라이선스의 경우 사용자에게만 소스를 공개하면 되고 SI 소프트웨어를 제3자에게 제공하는 경우도 없기 때문에 의뢰인(甲)에게 소스를 제공하는 게 업계 관행인 SI의 특성상 이러한 GPL의 전염성이 거의 문제시되지 않습니다. GPL 소프트웨어를 수정해서 유료로 판매하는 것은 이론상으로 가능하나 유료로 판매한 GPL 소프트웨어를 구매한 사용자가 다시 이를 수정해서 무료로 배포하는 것을 막을 수 없기 때문에, 유료 소프트웨어가 이러한 오픈 소스를 채택하는 경우는 드뭅니다.

- 유료화의 한계는 오픈 소스 소프트웨어의 발전에 장애가 되는데, 유료화가 제한적이며 이는 개발 자금 부족으로 이어지는 경우가 많습니다.

부족한 자금 때문에 실력있는 핵심적인 개발자를 영입하기 어렵고 외부 기술의 라이선스 취득에서도 약점이 됩니다. 고급 상용화 소프트웨어들이 모든 기능을 자체적으로 개발하는 경우는 매우 드물며, 외부에서 라이선스를 사와 여기에 기능을 덧붙여서 구현하는 경우가 많습니다. 모든 기능 개발에 집중할 필요가 없으면서도 외부에서 전문적으로 개발한 고급 기술을 사용할 수 있으니 개발 효율이 높아지게 됩니다.

- 오픈 소스에는 원칙적으로 사후 지원의 제공 의무가 없습니다.

이것은 소스 코드를 제공하니 문제가 발생하면 사용자가 직접 고쳐 쓰라는 의미입니다. 제작자는 오픈소스 프로그램을 수정해 줄 의무가 없습니다. 물론 일반적인 용도로 많이 사용되는 오픈 소스 소프트웨어에 이런 사후지원이 없으면 일반 사용자는 쓰기 힘들거니 사후 지원을 따로 계약하는 경우도 많고, 이러한 사후 지원은 비용을 받는 것이 일반적입니다. 제 3자가 다시 만들어 배포해도 라이선스만 지키면 기술 지원만 못 받는 것일 뿐, 불법 복제가 아닙니다.

오픈소스 사용추세 및 전망

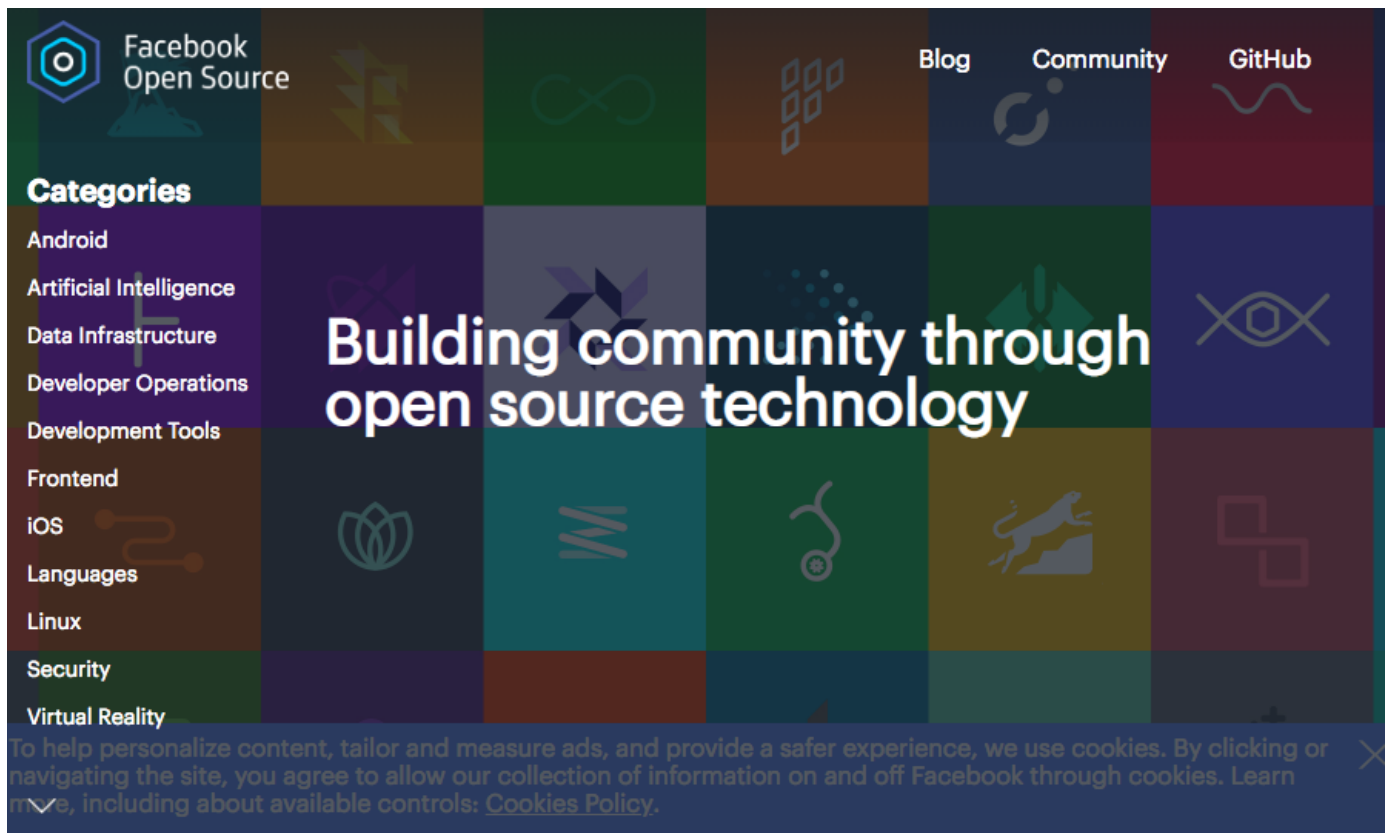
[Edit on GitHub](#)

오픈소스 사용추세

Deeply-positioned OpenSource in life

오픈 소스는 다양한 분야에서 사용되고 있습니다. 머신러닝 툴, 클라우드 컴퓨팅 소프트웨어, 네트워크 및 보안 소프트웨어 등 다양한 영역에서 오픈소스는 계속해서 무궁무진하게 진화하고 있습니다. 초기에는 리눅스재단의 후원으로 개발된 오픈 소스 컴퓨팅 프로젝트가 많았지만 최근 몇 년 동안 페이스북, 마이크로소프트, 구글 등 거대 기술 회사가 다양한 오픈 소스 프로젝트를 클라우드 소싱하면서 오픈 소스 시장에 뛰어들었습니다.

페이스북 - 파이토치(PyTorch)



페이스북은 페이스북 피드에 대한 정보를 정제하는 등 내부적으로 딥러닝을 사용하며 깃허브에서 호스팅 되는 파이토치(PyTorch) 딥러닝 프레임워크의 일부로 만든 모듈 중 일부를 오픈소스로 제공하고 있습니다.

넷플릭스- 카오스 멍키



주요 AWS 사용자인 넷플릭스는 클라우드에서 실행되는 애플리케이션의 복원력을 테스트하는 방법을 원했습니다. 카오스 멍키는 퍼블릭 클라우드 제공자가 호스팅하는 가상 머신으로 인위적으로 문제를 일으키기 위해 태어났습니다. 이 프로젝트는 시스템이 네트워크에서 임의의 실패에 대응할 수 있는지 테스트하기 위해 더 넓은 시미안 아미(Simian Army)에 흡수되었습니다.

넷플릭스는 현재 빅데이터 툴인 하둡, 하이브, 피그, 파케이(Parquet), 프레스토(Presto), 스파크 등 다수의 오픈소스 프로젝트에 기여하고 있습니다. 또한 넷플릭스에서 개발한 네볼라에는 수많은 그라들(Gradle) 플러그인이 오픈소스로 올라와 있습니다.

에어비엔비-에어플로우


[Open Source](#) [Events](#) [Blog](#) [Github](#) [Careers](#)

Open Source

Open source is at the heart of what we do at Airbnb

[All projects](#) [Data](#) [Backend](#) [Frontend](#) [Mobile](#) [Infrastructure](#) [Style guide](#) >


Airflow Use Apache Airflow (incubating) to author workflows as directed acyclic graphs (DAGs) of tasks

★ 8,613



Aerosolve A machine learning package built for humans

★ 4,385

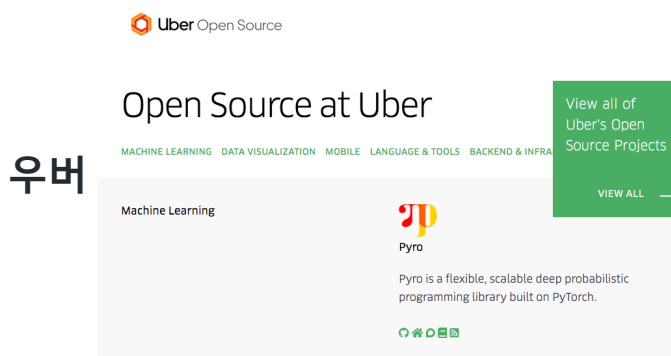


BinaryAlert Serverless real-time and retroactive malware detection

★ 705

에어플로우(Airflow)는 아파치 라이선스 아래서 사용할 수 있는 데이터 워크플로우 관리 프레임워크로 데이터 파이프 라인의 작성, 스케줄링, 모니터링을 지원합니다.

에어비앤비는 에어로솔브(Aerosolve) 머신러닝 툴도 오픈소스로 공개했습니다. 이 툴은 임대 속성을 위한 가격 권장 엔진 같은 기능을 지원하기 위해 내부적으로 사용됩니다.



차량공유 업체 우버에서 빠른 지형공간정보 빅데이터 처리를 위한 시각화툴을 오픈소스로 공개했습니다.

지형공간정보 빅데이터를 시각화 하는데 기존에 사용했던 방법은 오랜시간과 코딩이 필요했지만 시각화툴을 오픈소스로 공개해서 기존방식과는 별개로 우버의 서비스를 활용하면 웹기반으로 사용자가 원하는 시각화 결과를 손쉽게 얻을 수 있다고 합니다.

오픈소스의 전망

Open Source Perspectives

오픈소스의 가치는 계속 상승하는 추세입니다. IDC 와 정보통신산업진흥원(NIPA)의 '오픈소스 SW 시장 및 트렌드' 조사에 따르면 2016년 기준 전 세계 오픈소스 SW시장 규모는 약 600억 달러 (한화 65 조 2천억원) 규모를 형성했습니다.

기업이나 공공기관이 오픈소스 소프트웨어를 활용하는 것은 이제 전세계적인 추세입니다. 오픈소스 SW시장은 국내에서도 점차 무르익고 있습니다. 특히 4차산업혁명시대 진입에 따른 핵심기술로 다양한 분야에서 활용도가 더 높아지고 있다는 분석입니다.



실제 NIPA 공개 SW 포털에 따르면, 국내 오픈소스 SW시장은 클라우드와 빅데이터, 사물인터넷 (IoT), 인공지능 (AI)등 신산업 부문에서 활용이 늘면서 오는 2020년까지 연평균 15.2% 성장할 것으로 전망됩니다.

국내 오픈소스 SW시장은 2015년 1410억원, 2016년은 1602억원을 기록했고 2017년은 1834억원 규모로 전망됩니다. 이같은 추세라면 2년 후인 2020년에는 2862억원의 시장을 형성할 것으로 예상됩니다.

국내 전체 SW시장을 12~13조원으로 추정했을 때, 오픈소스SW 시장이 차지하는 비중은 미미합니다. 하지만 오픈소스 SW가 갖고있는 비용절감 효과가 최신기술 적용, 특별 벤더 종속 완화등의 가치는 관련 시장 확대의 원동력이 되고 있습니다.

정부의 오픈소스 SW 활성화 지원 사업도 계속해서 늘어나고 있는 실정입니다. 올해 들어선 전통, 융합산업별 요소기술 개발, 개방형 os 환경개발 등의 사업등을 추진하면서 활용 수요가 높은 공통 요소기술 개발에 대한 지원을 추진하고 있습니다.

앞으로 오픈소스는 무궁무진하게 발전해 나아갈 것이고 국내에서도 정부의 지원과 더불어 활용 수요가 상승하면서 오픈소스 SW가 국내시장에서도 활성화 될 것으로 보여집니다.

오픈소스의 역사

 [Edit on GitHub](#)

1. 1980년대 이전의 역사
2. 1980년대의 역사
3. 1990년대의 역사
4. 현대
 - GIT
 - Android
 - Chrome OS
 - 주요인물

참고문헌

1980년대 이전의 역사

[Edit on GitHub](#)

1960년대에는 하드웨어 판매가 중심을 이루고 소프트웨어는 그저 공유 문화 정도에 머물렀습니다.

1970년대에 와서는 소프트웨어의 비중이 점차 커지기 시작했는데요 벨 연구소에 의해 Unix 개발, AT&T를 비롯한 여러 회사, 버클리 등 비영리 단체에서 다양한 버전 개발, 1980년대 HP/UX, AIX, Solaris 등의 상용 버전 Unix 개발 및 상용화등이 일어나고있었습니다. 또한 오픈소스가 생기게 된 계기라고 할수있는 **가장 중요한 사건**이 발생하게 됩니다.

그것은 1976년 **빌게이츠**가 컴퓨터 애호가들에게 보내는 **"Open Letter to Hobbyists"**라는 공개편지입니다.

"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?"

(왜 그렇습니까? 애호가 분들의 대부분에 해당하긴 하지만, 당신들은 당신들의 소프트웨어를 훔친 것입니다. 하드웨어는 꼭 구입해야 하지만 소프트웨어는 그냥 공유하는 것입니까? 아무도 소프트웨어를 만들 사람이 보수를 받는지는 신경 안쓰는 것입니까? 그게 공평합니까?)

"Most directly, the thing you do is theft."

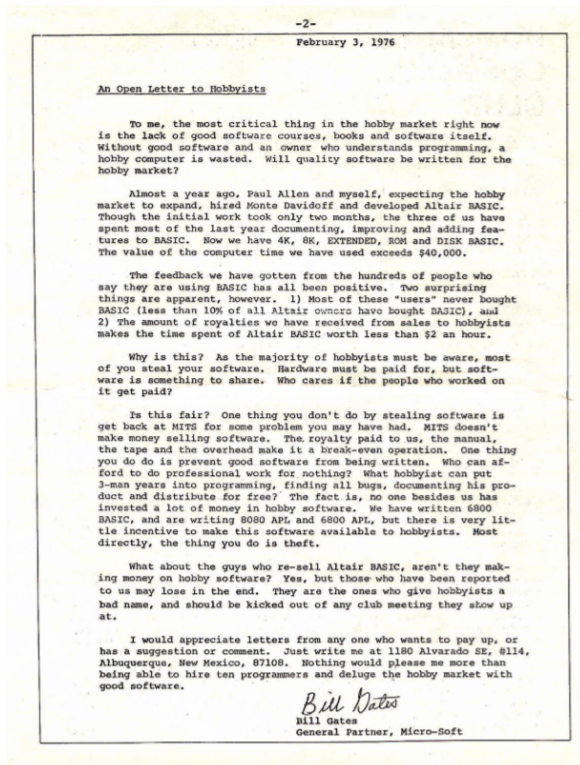
(정말 솔직하게 말해서, 당신들이 하고 있는건 절도 행위입니다.)

- 빌 게이츠(Bill Gates)의 Open Letter to Hobbyists 중

이 공개 편지가 큰 파장을 일으키며 **소프트웨어의 상업화**가 커지기 시작했습니다. 1976년 친구 폴 앨런과 MITS Altair 8800을 위한 베이식 인터프리터 제조 후 MITS를 통해 판매하며 소프트웨어 저작권에 대한 인지, 시작된 **상업용 소프트웨어 시대**가 찾아왔습니다.

결국 코드를 유로로 만들어 버리게 되는 계기인 **상용 소프트웨어가 등장하게 되었습니다.**

"빌 게이츠의 Open Letter to Hobbyists"



1980년대의 역사

[Edit on GitHub](#)

"누군가가 나의 등잔의 심지에서 불을 붙여가도 내 등잔의 불은 여전히 빛나고 있습니다"

- 토머스 제퍼슨(미국의 정치가)

이러한 명언은 이후 오픈소스의 소프트웨어 생성의 정신적 기초가 되었습니다.

하드웨어 중심체제로부터 소프트웨어의 비중이 높아지면서 소프트웨어 기업의 소프트웨어에 대한 이용, 배포, 복제, 수정 등에 일정한 제한을 가하려는 추세가 있자, 이러한 '독점' 체제에 반발해 '공유'를 주장하는 운동이 리처드 스톨만에 의해 일어난것이 오픈소스 소프트웨어의 기원입니다.

리처드 스톨만은 소프트웨어 상업화에 반대하고 소프트웨어 개발 초기의 상호 협력적인 문화로 돌아갈 것을 주장하며 **1984년 GNU(GNU is Not UNIX)프로젝트**를 주도했고, 이듬해인 **1985년 FSF(프리 소프트웨어 재단)**를 조직하면서, **'소프트웨어는 공유돼야 하며 프로그래머는 소프트웨어로 돈을 벌어서는 안 된다'**는 내용의 GNU 선언문을 제정하기도 했고, 이를 지원하기 위해 저작권(copyright)에 대응하는 **카피레프트 운동**도 주창했습니다.

마침내 **1989년에 최초의 오픈소스 라이선스 GNU GPL(General Public License)**를 배포했습니다.

프리 소프트웨어 재단(FSF)의 **Free**는 **'자유(Free)'**를 뜻하는 거지 **'무료(Free)'**를 뜻하는게 아닙니다.

FSF의 관점에서 자유의 의미는 크게 4가지로 볼 수 있습니다.

1. 프로그램을 어떠한 목적을 위해서도 실행할 수 있는 자유.
2. 프로그램의 작동 원리를 연구하고 이를 자신의 필요에 맞게 변경시킬수 있는 자유. 이러한 자유를 위해서는 소스코드에 대한 접근이 선행되어야 함.
3. 이웃을 돕기 위해서 프로그램을 복제하고 배포할 수 있는 자유.
4. 프로그램을 향상시키고 이를 공동체 전체의 이익을 위해서 다시 환원시킬 수 있는 자유. 이러한 자유를 위해서는 소스코드에 대한 접근이 선행되어야 함.



1990년대의 역사

[Edit on GitHub](#)

1990년대 들어서면서 인터넷의 보급과 더불어 **GNU GPL(General Public License)**로 배포된 리눅스의 보급으로 자유소프트웨어 운동이 확산됐고, 1991년에 핀란드 헬싱키 대학의 대학원생였던 **리누스 토발즈**가 취미로 개발한 커널인 **리눅스**가 탄생하였고 MINIX를 응용하여 **GNU GPL로 배포**하였습니다. 1998년에는 MS의 웹브라우저인 익스플로러에 밀려 어려움을 겪고 있던 **넷스케이프(Netscape)**사가 **웹브라우저 모질라의 소스코드를 공개하는 결정**을 하게 되었고, 공개형태를 결정하는 과정에서 오픈소스라는 이름이 생겼습니다.

이때 즈음 **자유소프트웨어**라는 용어에서 **오픈소스 소프트웨어**로 용어가 변경되는데, **자유(free)**란 용어 때문에 일반인들이 **무료**라고 인식하고 있다는 점, **GPL 조항의 엄격성** 때문에 소프트웨어 **개발이 용이하지 않다**는 점을 탈피하기 위해서였습니다.

이러한 경향들을 시작으로 이후 1998년 넷스케이프 소스코드 공개에 자극을 받아 오픈소스를 장려하기 위해 오픈소스 소프트웨어를 인증하는 **OSI(Open Source Initiative)**가 에릭레이몬드(Eric Raymond) 등에 의해 결성되면서 오픈소스 소프트웨어운동은 궤도에 오르게 되었습니다.

OSI 단체가 정한 오픈소스 소프트웨어의 기준을 OSD(Open Source Definition)이라 하는데, 이 기준을 만족해야만 오픈소스 소프트웨어로 인정받을수 있습니다

참고로, OSD 기준에 따르면 오픈소스 소프트웨어는 6가지 요건을 갖추어야 합니다.

1. 자유로운 재배포.(Free Redistribution)
2. 소스코드 공개.(Source Code)
3. 2차적 저작물 허용 .(Derived Works)
4. 저작자의 소스코드의 온전함.(Integrity of The Author's Source Code)
5. 차별금지. (No Discrimination Against Persons or Groups 및 No Discrimination Against Fields of Endeavor)
6. 라이선스의 배포.(Distribution of License).



OSI가 인증한 오픈소스SW 라이선스에는 OSI의 인증마크를 부여한다

위와 같은 역사를 거쳐 이제 오픈소스 소프트웨어는 소프트웨어의 한 축을 잡게 되었습니다.

현대

 [Edit on GitHub](#)

Git

[Edit on GitHub](#)

Git



Git이란?(2005~)

Linux kernel은 대규모 **Open Source Project**이다. 이 project의 버전관리를 위해 초기에는 **BitKeeper**라는 **상용DVCS**를 사용했었는데 2005년도에 이 BitKeeper의 **무료사용이 제고**되면서 리누스 토발즈의 주도로 Linux 개발 커뮤니티가 **자체 VCS를 개발**했는데 이게 바로 **Git**입니다. **2005**년부터 지금까지 주니오 하마노(Junio Hamano)가 소프트웨어의 유지보수를 맡고 있습니다.

Git은 GNU 일반 공중 사용 허가서 v2 하에 배포되는 자유 소프트웨어입니다.

원저자	리누스 토발즈
개발자	주니오 하마노, 리누스 토발즈
프로그래밍 언어	C, 셸, 펄, Tcl, 파이썬
운영체제	리눅스, POSIX, 윈도우, OS X
언어	영어
종류	버전관리
라이선스	GNU 일반 공중 사용 허가서 v2

Git의 특징

- 단순한 구조에서 오는 빠른 속도
- 완벽한 분산처리
- branch를 사용한 비선형적 개발 가능
- 속도나 크기면에서 대형 Project에 적합

Git의 기본

Git은 파일을 3가지 상태로 관리합니다.

- Committed: 파일을 수정한 후 해당 파일에 대해 commit명령을 실행해 파일을 로컬 데이터 베이스(로컬 Repository)에 안전하게 저장한 상태
- Modified: 파일을 수정한 후 아직 로컬 데이터 베이스에 commit하지 않은 상태를 의미
- Staged: 파일을 수정한 후 수정 할 파일을 곧 commit 할 것이라고 표시한 상태를 의미

Git은 파일상태 관리와 더불어 3가지 영역을 사용합니다

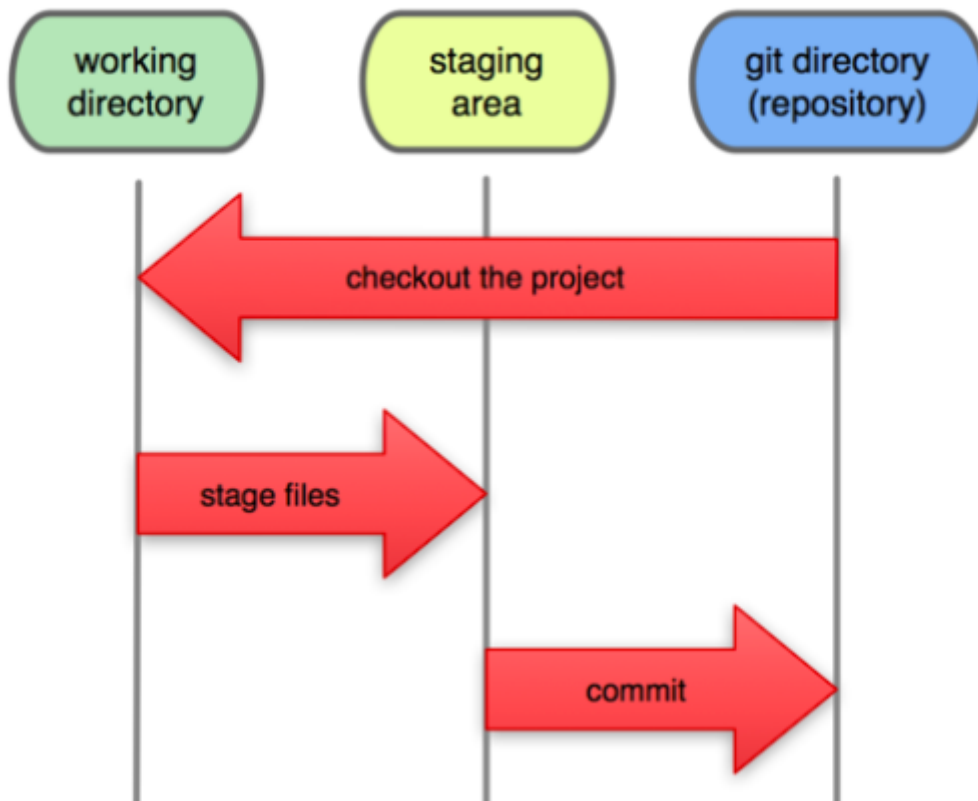
- Git directory : Git이 project의 메타 데이터와 객체 데이터베이스를 저장하는 곳을 의미합니다. 다른말로 Local Repository라고 하며 만약 특정 폴더를 Git directory 로 설정하려면 git init명령을 이용하면 됩니다. Repository로 설정되면 .git이라는 숨김폴더가 생성되고 이 안에 Git관리 정보들이 생성됩니다.
- Staging Area: Git directory에 존재하며 단순한 파일입니다. 곧 commit할 파일에 대한 정보를 가지고 있게 됩니다.

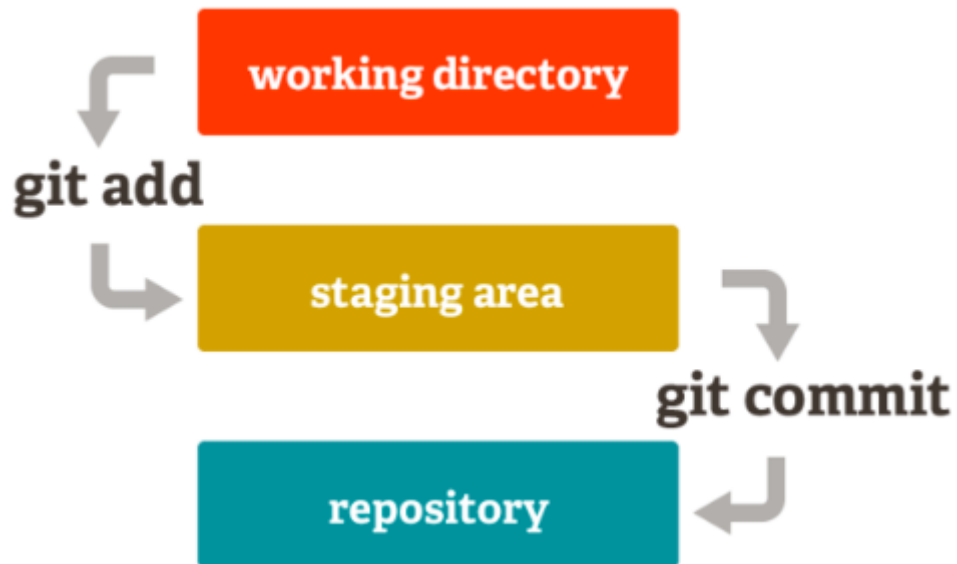
- Working directory: project의 특정 branch를 checkout한 내용이 들어있는 폴더

Git으로 하는 작업의 기본 순서

- Working directory에서 파일 수정
- Staging Area에 수정한 파일을 Stage해서 commit할 Snapshot 생성(`git add`)
- Staging Area에 있는 수정된 파일을 commit해서 Git directory에 영구적인 Snapshot으로 저장(`git commit`)

Local Operations





GitHub



GitHub란?(2008~)

깃허브는 분산 버전 관리 툴인 깃을 사용하는 프로젝트를 지원하는 웹호스팅 서비스어 입니다. GitHub는 영리적인 서비스와 오픈소스를 위한 무상서비스를 모두 제공합니다. 2009년의 Git사용자 조사에 따르면 GitHub는 가장 인기있는 Git호스팅 사이트입니다. 또한 2011년의 조사에서는 가장 인기있는 오픈 소스코드저장소로 꼽혔습니다.

깃이 텍스트 명령어 입력방식인데 비해, 깃허브는 화려한 그래픽 유저 인터페이스(GUI)를 제공합니다. 깃허브는 페이스북과 유사한 서비스인 Gits와 위키를 각 저장소마다 운영하고 있으며, 깃 저장소를 통해 고칠 수 있습니다.

깃허브 회사는 2008년 톰 프레스턴워너(Tom Preston-Werner), 크리스 완스트래스(Chris Wanstrath), 피제이 하이엣(PJ Hyett)이 공동 설립했습니다. 본사는 미국 캘리포니아 주 샌프란시스코에 있으며 2018년 6월 4일 마이크로소프트는 7,500,000,000 달러에 깃허브를 인수할 것이라 발표하였습니다.

GitHub에서 가능한 것들

- 깃허브는 대부분 코드를 위해 사용됩니다.
- 다양한 마크다운 식의 파일 포맷의 자동으로 렌더링 되는 README파일을 포함한 문서화
- 레이블, 마일스톤, assignee, 검색엔진을 갖춘 issue tracker(기능 요청 포함)
- 코드 검토 및 댓글을 지원하는 Pull Request
- Commit history
- 그래프
- Integration Directory(연동 디렉터리)
- diff 통합 및 분석
- 이메일 알림
- 파일 내의 중첩 작업 목록
- 지형공간 분석 데이터 시각화
- PDF 문서 뷰어
- 각기 다른 패키지에서 공통 취약점 및 노출로 알려진 보안 경보

Android

[Edit on GitHub](#)

Android란?(2008~)



안드로이드는 휴대전화를 비롯한 휴대용 장치를 위한 운영체제와 미들웨어, 사용자 인터페이스 그리고 표준 응용프로그램 (웹 브라우저, 이메일 클라이언트, 단문 메시지 서비스(SMS), 멀티미디어 메시지 서비스(MMS)등) 을 포함하고 있는 소프트웨어 스택이자 모바일 운영체제입니다. 안드로이드는 개발자들이 자바와 코틀린 언어로 응용프로그램을 작성할 수 있게 하였으며, 컴파일 된 바이트코드를 구성할 수 있는 런타임 라이브러리를 제공합니다. 또한 안드로이드 소프트웨어 개발 키트를 통해 응용 프로그램을 개발하는 데 필요한 각종 도구와 응용프로그램 인터페이스(API)를 제공합니다.

회사/개발자	구글, 오픈 핸드셋 얼라이언스(OHA)
OS 계열	유닉스 계열, 리눅스
소스 형태	오픈 소스 소프트웨어
마케팅 대상	스마트폰, 태플릿 PC, 스마트워치, TV , 셋톱박스
사용가능한 언어	100개 이상
프로그래밍 언어	C, C++, 자바, 코틀린
커널 형태	리눅스 커널
기본 UI	그래픽 사용자 인터페이스
라이선스	아파치 2.0, GPL v2

구글은 안드로이드의 모든 소스 코드를 오픈소스 라이선스인 아파치 v2라이선스로 배포하고있어 기업이나 사용자는 각자 안드로이드 프로그램을 독자적으로 개발을 해서 탑재할 수 있습니다. 또한 등록한 개발자들이 소비자에게 응용프로그램을 판매할 수 있는 구글 플레이 스토어를 제공하고 있으며, 이와 별도로 각 제조사 혹은 통신사별 응용 프로그램 마켓이 함께 운영되고 있습니다.

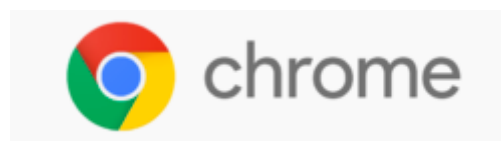
Android의 특징

- 오픈소스
- 스마트폰을 위한 완벽한 컴포넌트 제공
- 누구나 만들 수 있는 무료 플랫폼
- 자바 개발 언어
- 커널이 공개 운영체제인 리눅스에 기반
- 검증된 많은 라이브러리를 대거 포함
- 플랫폼에 내장된 빌트인 프로그램과 사용자가 많은 프로그램이 동일한 API사용
- 어플리케이션 프레임 워크

Chrome OS

[Edit on GitHub](#)

Chrome OS란?(2009~)



크롬 os는 구글이 설계한 차기 오픈소스 운영체제이며 웹 애플리케이션과만 동작합니다. 2009년 7월 7일, 크롬 os는 2010년에 공식적으로 안정된 버전이 출시될 예정이라고 발표하였습니다. 이 운영체제는 리눅스에 기반을 두고 있으며 지정된 하드웨어에서만 동작된다. 사용자 인터페이스는 크롬 웹 브라우저의 것과 비슷합니다.

회사/개발자	구글
os계열	크로미움 os
프로그래밍 언어	C,C++
지원되는 플랫폼	x86(32비트), 32비트 ARM(ARMv7)
커널 형태	모놀리딕(리눅스)
기본 UI	구글 크롬 브라우저 기반 그래픽 인터페이스
라이선스	구글 크롬 os서비스 조항

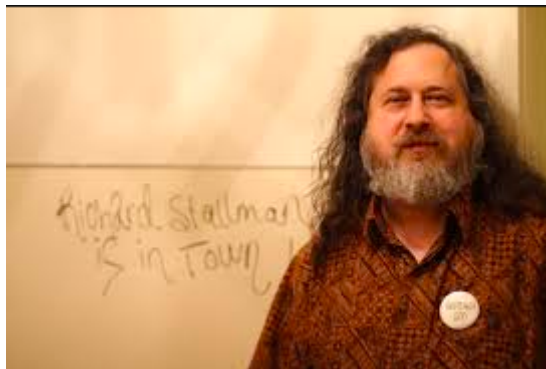
Chrome Os의 특징

- 크롬만의 멀티프로세스 아키텍처로 충돌이 없습니다. (어느 한 곳에서 문제가 발생해도 다른 브라우저에까지 그 여파가 미치지 않습니다.)
- 멀티프로세스 아키텍처로 브라우저들 사이에 속도로 영향이 없기때문에 빠릅니다.
- 인터페이스가 단순합니다.
- 옴니박스(브라우저 상단에 위치한 다목적 바(bar))에 검색어와 URL 동시에 넣을 수 있어서 검색이 보다 쉽습니다.
- 탭 조절이 더 간편해졌습니다.
- 자주가는 홈페이지에 더 쉽게 접근할 수 있도록 배려합니다.(스냅샷 형태로 노출)
- 사생활 보호가 뛰어나다(인코그니토(Incognito))라는 이름의 보안 브라우징 옵션을 제공합니다. (이 기능이 제공하는 특별한 창을 이용할 경우 이 때 했던 모든 작업들에 대한 기록이 남지 않습니다.)

주요 인물

[Edit on GitHub](#)

Richard Matthew Stallman(1953~)



살아온 길

1953년 3월 16일 맨해튼에서 출생

1960년대 처음으로 개인용 컴퓨터를 접함. IBM 뉴욕 과학센터에서 일하면서 IBM 7064를 위한 전처리기를 PL/I프로그래밍 언어로 작성

1971년 하버드 대학의 신입생으로 MIT인공지능 실험실의 해커가 됨

1980년대 심볼릭스의 결과물들과 똑같은 기능의 프로그램을 작성하여 그들의 독점을 막는일을 계속함. 그러나 그는 그의 세대 중에서 마지막 해커였으며 그의 원칙인 다른 이들과의 공유나 이웃을 돕는 것에 위배되는 작업들을 수행할 것을 요구받음

1985년 스톨만은 비영리 재단인 FSF(자유 소프트웨어 재단)설립, GNU(Gnu is Not Unix)선언문을 발표

1989년 일반공중사용허가서 (GPL)내에 카피레프트의 개념 적용. 허드 커널을 제외하고 대부분의 GNU시스템이 거의 동시에 완성

1991년 리누스 토발즈는 GPL로 리눅스 커널을 발표

1998년 전자 개척 재단의 선구자상을 리투스 토발즈와 공동 수상

=====

그는 문서 편집기인 Emacs, GNU 컴파일러 모음 컴파일러, GDB 디버거 등 많은 프로그램을 만들었으며 이들 모두를 GNU프로젝트의 일부로 만들었습니다. 그는 자유 소프트웨어 운동의 도덕적, 정치적, 법적인 기초를 세우는데 본질적인 영향을 준 인물이며, 이는 독점 소프트웨어 개발과 공급에 대한 대안이 되었습니다.

Linus Benedict Torvalds(1969~)



살아온 길

1969년 12월 28일 핀란드 헬싱키에서 출생

1989년 헬싱키 대학교 입학

1996년 컴퓨터 과학 석사로 졸업

1990년 최초로 DEC MicroVAX에서 운영하는 ULTRIX의 형태로 유닉스를 만나게 됨

1991년 인텔 80386 기반의 IBM PC구입

2000년 헬싱키 대학교는 리누스 토발즈에게 명예박사학위 수여

2010 년 미국 시민권 취득

=====

=====

리누스 토발즈 스웨덴계 핀란드인으로서 소프트웨어 개발자이자 리눅스 커널과 깃을 최초로 개발한 사람으로 잘 알려져 있습니다. 후에 그는 리눅스 커널 개발 최고 설계자가 되었고, 현재 프로젝트 코디네이터로 활동하고 있습니다. 그는 커널의 플랫폼 독립적인 부분과 인텔 IA-32 아키텍처로 구체화되는 핵심 커널의 컴포넌트들을 관리합니다. 저명한 오픈소스 소프트웨어 개발리더들에게 부여되는 명예 타이틀직인 자비로운 종신독재자(BDFL, en:Benevolent Dictator for Life) 중의 한 사람이기도 합니다.

Eric Steven Raymond(1957~)



살아온 길

1957년 미국 매사추세츠주 보스턴에서 출생

펜실베이니아 대학 입학

1970년후반 아르파넷 시절부터 인터넷과 해커문화에 매료돼 참여하고 관찰하며 해커로 활동

1997년 리눅스회의에서 성당과 시장 처음공개

1999년 같은 이름의 책에 포함되어 출판

2015 termcap의 관리자로 프로젝트 수행 중

=====

=====

그의 연구는 리눅스와 인터넷의 발전을 통해 효과적으로 증명된 분산화된 오픈소스 소프트웨어 개발 모델을 설명하는데 기여했습니다. 직접 만든 여러 오픈소스 프로젝트 중에서 가장 널리 알려진 것은 페치메일(fetch mail)입니다. 페치메일은 인터넷에서 가장 많이 사용되는 이메일 전송 프로그램으로 모든 주요 리눅스 배포판에 포함돼 있습니다. 이외에도 GNU Emacs와 ncurses에도 크게 공헌하였습니다.

에릭레이먼드는 오픈소스 개발 과정이 어떤 식으로 운영되고 반복되는지 설명한 그의 초기문서인 "성당과 시장"의 저자이기도 합니다. 해커들이 쓰는 용어를 설명한 자곤파일(《The New Hacker's Dictionary》)을 편집했으며, 넷스케이프의 소스코드 공개, 모질라의 설립에도 큰 영향을 끼침으로써 오픈소스의 대표 인물이라고 말 할 수 있습니다.

참고문헌

[Edit on GitHub](#)

- OSI - <https://opensource.org/>
- 오픈소스(네이버 지식백과, 위키백과)
- 오픈소스 교육 자료 - <https://www.slideshare.net/JerryJeong2/ss-58804386>
- 오픈소스 라이선스 종류 - <https://olis.or.kr/license/licenseClassification.do>
- Top10 오픈소스 프로젝트 사례 - <https://opensource.com/article/16/12/yearbook-top-10-open-source-projects>
- 오픈소스 - <http://sketchit.tistory.com/18>
- 오픈소스 사용추세 및 현황 - <http://www.gartner.com>
- 오픈소스 소프트웨어의 역사 - <http://www.ddaily.co.kr/news/article.html?no=115248>
- 빌게이츠 편지 - https://en.wikipedia.org/wiki/Open_Letter_to_Hobbyists
- GNU - http://www.econote.co.kr/main/view_post.asp?post_seq_no=23725
- GNU 프로젝트 - <https://ko.wikipedia.org/wiki/GNU>
- FSF - https://en.wikipedia.org/wiki/Free_Software_Foundation
- FSF의 자유 의미 - <https://www.gnu.org/philosophy/free-sw.html>
- Linux - <https://ko.wikipedia.org/wiki/리눅스>
- Git - <https://en.wikipedia.org/wiki/Git>, <https://git-scm.com>
- Android - [https://ko.wikipedia.org/wiki/안드로이드_\(운영_체제\)](https://ko.wikipedia.org/wiki/안드로이드_(운영_체제))
- Chrome OS - https://en.wikipedia.org/wiki/Chrome_OS
- 리처드 스톨만 - https://ko.wikipedia.org/wiki/리처드_스톨먼
- 리누스 토르발스 - https://en.wikipedia.org/wiki/Linus_Torvalds
- 에릭 S 레이몬드 - https://en.wikipedia.org/wiki/Eric_S._Raymond
- 오픈 소스 - Eric Raymond 저 / 송창훈 역 / 한빛미디어
- 스마트 혁명과 오픈 소스 리더십 - 레슬리 가드맨, 캐리 쿠퍼 저 / 이미숙 역 / 멘토르
- 리눅스와 오픈소스의 비즈니스와 경제학 - Martin Fink 저 / 조광제 역 / 영진닷컴