



®

open source  
initiative

---

# Table of Contents

## 1.0 오픈소스 소개

Introduction	1.1
1.1 오픈소스란 무엇인가?	1.2
1.2 오픈소스의 장점과 단점	1.3
1.3 오픈소스의 저작권과 라이선스	1.4
1.4 성공한 오픈소스 프로젝트	1.5

## 2.0 오픈소스의 역사

2.1 1950년대 ~ 1960년대	2.1
2.2 1970년대	2.2
2.3 1980년대 GNU의 설립	2.3
2.4 오픈소스라는 단어의 사용	2.4
2.5 성당과 시장	2.5
2.6 1990년대	2.6
2.7 오픈소스의 황금기	2.7

## 3.0 오픈소스의 문화

3.1 intro	3.1
3.2 오픈소스 프로그래밍 언어	3.2
3.3 프레임워크 & 라이브러리	3.3
3.4 일반 소프트웨어	3.4
3.5 머신러닝	3.5
3.6 CMS	3.6
3.7 OS	3.7
3.8 디자인 요소	3.8
3.9 하드웨어 요소	3.9

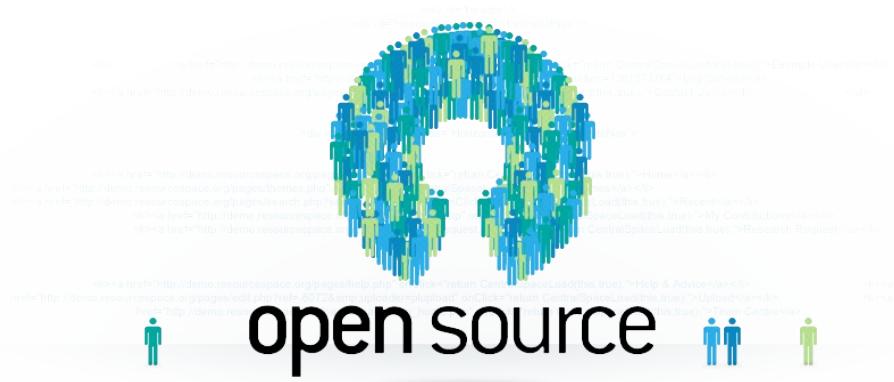
## 참고문헌

참고문헌	4.1
------	-----



## 1. 오픈소스 소개

### 1.1 오픈소스란 무엇인가?



오픈소스(Open Source), 또는 공개 소프트웨어는 저작권자가 소스코드를 공개하여 누구나 특별한 제한 없이 자유롭게 사용, 복제, 배포, 수정할 수 있는 소프트웨어다. 오픈 소스는 소프트웨어 혹은 하드웨어의 제작자의 권리를 지키면서 소스 코드를 누구나 열람할 수 있도록 한 소프트웨어 혹은 오픈소스 라이선스에 준하는 모든 통칙을 일컫는다. 어원에 대해 History of the OSI 자료에 따르면, 1998년 2월 3일에 넷스케이프 브라우저의 소스 코드에 대해 어떠한 형태로 공개할까하는 전략회의에서 붙여진 새로운 용어라고 설명되어 있다.

소스 코드를 공개한다고 해서 모두 오픈소스인 것은 아니다. 예를 들어, 마이크로소프트는 전체 비율로 보았을 때 극소수의 고객(주로 정부나 거대 다국적 기업 또는 대학교 및 연구소)들에게 마이크로소프트 윈도우의 소스를 공개했다. 오로지 보안 유지를 위해서만 소스를 직접 수정할 수 있으며, 그 수정본을 재배포하는 것은 금지되어 있다. 이것은 오픈소스의 의의에 어긋나므로 이러한 경우는 오픈소스라 부르지 않는다.

오픈소스는 원래에는 컴퓨터 소프트웨어 개발에서 유래했지만 현재에는 오픈 소스 방식이라고 칭하는 가치들인 공동참여, 오픈교환, 투명성 등을 두루 일컫는 개념들뿐만 아니라 이러한 방식을 일부분으로 포함함으로써 지속 가능한 목표로 표방하는 행위까지도 넓혀져 가고 있다. 오픈소스는 소스를 공개한다는 면에서 Freeware, Shareware와 다르며, 오픈소스 이니셔티브(OSI)라는 단체에서 오픈소스에 대한 라이선스와 가이드를 제시하고 있다.

#### 소스코드를 공개하는 이유

이미 있는 오픈소스를 포크하여 새 프로젝트가 만들어지는 경우, 라이센스 때문에 공개해야 한다. 소프트웨어를 더 좋게 만들기 위해서 이기도 하다. 외부 개발자들이 참여하도록 하면 더 다양한 환경에서 소프트웨어를 시험해보고 품질을 높일 수 있다. 사회 공헌을 위해서 이기도 하고 프로그램의 신뢰성을 보장하는 방법이 될 수 있기 때문이기도 하다.

#### 컨트리뷰션(contribusion)

오픈소스 프로젝트에 도움이 되는 모든 활동을 컨트리뷰션(contribusion)이라고 한다. 버그 수정, 기능 추가, 소스 코드 수정, 코드 테스트와 같은 중요한 활동 이외에도, 오타 수정, 번역, 가이드 문서 작성, 디자인 작업, 의견 제시와 같은 사소한 활동도 컨트리뷰션에 해당한다.

#### 컨트리뷰션을 하는 이유

- 사용하던 오픈소스를 개선한 경우 자신만 사용하면 버전업이 될 때마다 추가로 패치해야하는 불편함이 있다. 이 불편함을 해소하기 위해 컨트리뷰션을 한다.
- 개인의 개발 실력 향상과 영어능력 향상을 위해서 하는 경우도 있다. 오픈소스 커뮤니티로부터 피드백을 받을 수 있어 특히 개발 실력 향상에 도움이 된다.

- 오픈소스 활동은 공개 기록으로 남기 때문에, 구직 활동 시 자신의 이력으로 사용할 수 있다. 이 외에도 명성을 높이려고 기여하기도 하는 등 다양한 이유로 사람들은 기여를 한다.

## 1.2 오픈소스의 장점과 단점

### 오픈소스의 장점

- 낮은 진입비용 : 오픈소스는 무료로 다운로드 및 소스코드의 수정/재배포가 가능하므로 일반적으로 초기 개발비용이 새로 개발하는 것에 비해 1/2 정도인 것으로 알려져 있다.
- 빠르고 유연한 개발 : 오픈소스 커뮤니티는 보통 최신 기술 정보 및 문제점과 해결책을 공유하는 형태로 자유롭게 운영되기 때문에 독점 프로그램에 비해 기술 발전 속도가 빠르다.
- 호환성 : 오픈소스는 주로 오픈포맷 또는 프로토콜을 사용하기 때문에 서로 다른 소프트웨어간 상호연동성이 보장된다. 모든 기기들이 서로 다른 네트워크를 통해 하나로 연결되는 유비쿼터스 시대에 필수적인 요소이다.
- 신뢰성과 안정성 : 오픈소스의 개발 과정을 보면 전 세계에 있는 수많은 우수한 개발자들이 직접 개발과 디버깅 과정에 참여하기 때문에 In-house에서 폐쇄적으로 개발되는 독점 프로그램에 비해 비교적 안정적으로 동작한다. 하지만 신뢰성과 안정성은 많은 개발자들이 적극적으로 참여할 때에만 가능하므로, 사용하고자 하는 오픈소스의 개발과정, 평판 등을 주의깊게 살펴보아야 한다. 실제로 잡다한 수많은 오픈소스들이 있기 때문에 쓸만한 오픈소스를 가려내는 것이 중요하다.

그 외에도 원하는 대로 변형 가능하다는 점이나 보안 취약점이 쉽게 발견된다는 점, (그러나 오픈소스가 보안 측면에서 더 우수하다는 것을 의미하지는 않는다.) 누구나 버그를 고칠 수 있다는 점, 특정 벤더에 의존하지 않아도 된다는 점 등이 있다.

### 오픈소스의 단점

- 어플리케이션의 부족 : 대부분의 이용자들이 MS 윈도우즈기반의 GUI에 익숙한 반면, 오픈소스는 GUI가 일반적이지 않다. 또 오픈소스는 리눅스 기반으로 개발된 애플리케이션이 많기 때문에 윈도우즈 기반 애플리케이션과 호환되지 않는 문제점도 있다.
- 빈약한 문서 : 오픈소스를 수정하여 원하는 애플리케이션을 제작하고자 할 경우 문서화가 중요한데, 상용 프로그램에 비해 오픈소스는 체계적인 문서를 갖지 않은 경우가 많다. 경우에 따라서는 개발과정을 지체시키는 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 저작자의 정책을 고려해야 한다.

그 외에도 비숙련 사용자들은 사용이 어렵다는 점, 고객지원이 불리하다는 점 등이 있다.

## 1.3 오픈소스의 저작권과 라이선스

자유로운 배포가 가능한 오픈소스라고 해도 저작권에 대한 개념이 없는 것은 아니다. 저작권법에 의한 소프트웨어의 창작물은 다른 사람이 쓸 수 없도록 보호하는데 목적이 있는데에 반해 오픈소스는 더 나은 오픈소스가 될 수 있는 발전 방법과 지켜야 하는 규칙이 있다. 이렇게 지켜야 하는 규칙과 방법을 명확히 알려주기 위해 오픈소스 라이선스를 몇 가지 종류로 구분을 하였다. 다음 표가 오픈소스의 종류별로 규칙과 방법이 뭐가 있는지를 나타낸 것이다. 무료 이용가능 여부, 배포 허용가능 여부, 소스코드 취득가능 여부, 소스코드 수정 가능 여부, 2차 저작물 재공개 의무 여부, 독점 소프트웨어에 사용 가능 여부 등으로 구분되어 있다.

	GPL	LGPL	MPL	BSD	Apache
무료 이용가능	○	○	○	○	○
배포 허용가능	○	○	○	○	○
소스코드 취득가능	○	○	○	○	○
소스코드 수정가능	○	○	○	○	○
2차 저작물 재공개 의무	○	○	○	×	×
독점 SW와 결합가능	×	○	○	○	○

- GPL : General Public License. 저작권은 개발자에게 귀속되지만 소프트웨어의 복사, 수정 및 변경, 배포의 자유를 제 3자에게 허용.
- LGPL : GNU Lesser General Public License. GPL을 변형해 더 허가된 형태로서, 소프트웨어 라이브러리를 염두에 둔 것.
- MPL : 모질라 공용 허가서(Mozilla Public License). 모질라 애플리케이션 스위트, 모질라 파이어폭스, 모질라 선더버드 및 그 외의 모질라 소프트웨어들에 적용.
- BSD : 유닉스(Unix)의 양대 뿌리 중 하나인 버클리의 캘리포니아 대학에서 배포하는 공개 소프트웨어의 라이선스. GPL보다 훨씬 개방적인 4개항의 간단한 문구로 되어 있음.
- Apache : 아파치 소프트웨어 재단에서 자체적으로 만든 소프트웨어에 대한 라이선스 규정. 누구나 해당 소프트웨어에서 파생된 프로그램을 제작할 수 있으며 저작권을 양도, 전송할 수 있는 라이선스 규정을 의미.

## 1.4 성공한 오픈소스 프로젝트

### LINUX



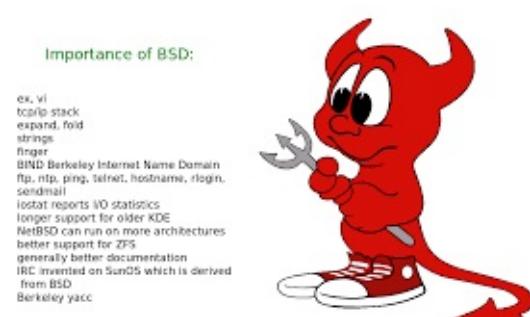
리눅스는 커널의 일종이다. 또한 그 커널을 사용하는 운영체제를 가리키는 말이기도 하다. GNU쪽 사람들은 리눅스는 커널일 뿐이고, 이 커널을 가져다가 GNU 프로그램들을 올려 만든 운영 체제는 GNU/Linux라고 이야기하며 이런 명칭에 민감하게 반응하는 경우도 있다. 소스 코드가 공개되어 있는 대표적인 오픈소스 소프트웨어다. 컴퓨터 역사상 가장 많은 사람이 들어간 오픈 소스 프로젝트다.

### UBUNTU



우분투는 리눅스 커널을 기반으로 한 리눅스 배포판 가운데 하나이다.

### BSD



BSD는 빌 조이(Bill Joy)를 주축으로 캘리포니아 대학교 버클리 캠퍼스의 CSRG(Computer Systems Research Group)에서 개발한 유닉스 계열의 운영 체제이다. BSD는 Berkeley Software Distribution의 약자이다. BSD 라이선스로 배포되며 GPL과는 달리 자기가 소스코드를 수정 및 추가해도 소스 코드 공개 의무가 없기 때문에 macOS, iOS, 솔라리스, TmacOS 등의 상업적인 운영 체제에도 많이 사용된다. 커널만 개발하는 리눅스 프로젝트와는 다르게 윈도우즈나 macOS처럼 데스크탑 환경과 응용 소프트웨어를 모두 포함하여 개발 및 배포한다.

### MySQL



MySQL은 1955년 발표된 오픈소스 DBMS이다.

### APACHE



Apache HTTP Server(약칭 "httpd")는 오픈 소스 소프트웨어 그룹인 아파치 소프트웨어 재단에서 만드는 웹 서버 프로그램이다. 팀 버너스 리가 만든 최초의 웹 서버 프로그램인 "NCSA HTTPd"를 기반으로 만들어졌다. 팀 버너스 리의 NCSA HTTPd는 유닉스 기반으로 만들어졌기에, 아파치 HTTP 서버는 NCSA HTTPd를 리눅스에서도 돌리는 것을 목표로 만들어진 프로그램이다. 그 이후 리눅스와 함께 퍼져나갔고, 리눅스가 서버 OS의 최다 점유율을 차지하자 아파치도 자연스럽게 최다 점유율을 차지하게 되었다. 현재 거의 모든 리눅스 배포판이 이 아파치를 지원한다.

### FIREFOX



파이어폭스는 모질라 재단에서 데이브 하야트(Dave Hyatt), 조 휴이트(Joe Hewitt), 블레이크 로스(Black Ross) 주도로 개발한 게코 엔진 기반의 오픈 소스 웹 브라우저이다. 원래 만들어졌을 당시 이름은 Phoenix였다. 0.6, 0.6.1, 0.7, 0.7.1 버전에서는 Firebird라고 불리었고, 0.8 버전부터 지금까지 Firefox라는 이름으로 불리우고 있다.

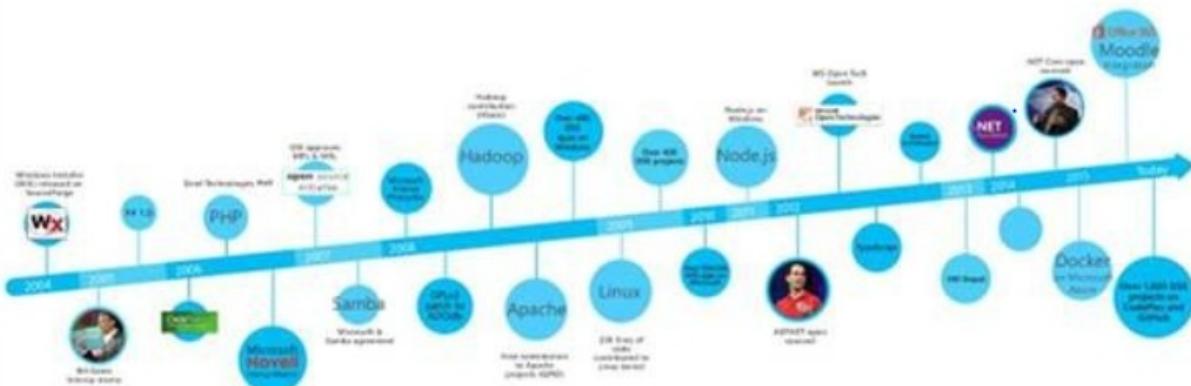
### WORDPRESS



워드프레스는 오픈소스 설치형 블로그이자 CMS, PHP 기반이며 라이센스는 GNU GPL v20이다.

## 2. 오픈소스의 역사

### 10+ Years of Open Source Involvement



## 2.1 1950년대 ~ 1960년대

### • 1950년대

컴퓨터 프로그램 초기 사용시기에는 연구와 학술분야를 기반으로 모든 프로그램들을 공개적으로 개발하였다. 1950년대 후반부터는 일부 판매가 시작되었다.

### • 1960년대

1960년대는 IBM360으로 대표되는 메인프레임이 지배하던 시대였다. 이 당시에는 소프트웨어와 하드웨어가 밀접하게 결합되어 있었기 때문에 어떤 특정 기종을 위해 만들어진 소프트웨어는 다른 기종에서 작동하지 않았다. 이때 IBM은 소프트웨어의 소스코드를 폐쇄하지 않았다. 그러나 IBM이 반독점법 패소에 대한 대응방안의 하나로 소프트웨어를 하드웨어와 분리하면서 소프트웨어 자체가 중요한 산업이 되고 기업들간의 경쟁이 심화되면서 소프트웨어의 소스코드는 기업의 비밀이 되고 모든 기업들에게 소스코드는 경쟁우위를 보장해주는 기술자원으로 변하였다. 이러한 흐름의 다른 한쪽에서는 소스코드를 공개하는 새로운 흐름이 형성되고 있었다.



1969년 AT&T의 벨연구소에서 근무하고 있던 켄 톰슨이 개인적인 연구를 수행하기 위해 Unix라는 운영체제를 만들었다. 이 운영체제는 C라는 새로운 고급언어로 작성되었는데 이는 하드웨어에 종속되지 않는 특징을 지니고 있었다. 이로 인해 벨 연구소가 아닌 곳에서도 유닉스를 사용할 수 있게 되었다. 이렇게 개발된 유닉스의 소스코드는 주로 연구기관을 중심으로 배포되었다.

1960년 후반부터는 소프트웨어로서 정식 상품화되면서 소스코드를 공개하지 않게 되었다. UNIX등 공개 소프트웨어는 소스 코드를 공개했다.

## 2.2. 1970년대의 오픈소스

### • Intro

1970년대에는 1960년도와는 분위기가 달라지기 시작한다.

소프트웨어 기술 발전과 더불어 개발 비용이 들어가면서 소프트웨어 상품 시장이 등장했기 때문이다.

IBM이 하드웨어에서 소프트웨어를 분리하는 Unbundling 정책도 한몫하였다.

### • 1970년대 초반

1970년대 초반, 벨 연구소 직원인 켄 톰슨, 데니스 리치, 더글拉斯 매클로린 등이 유닉스는 1960년대 말의 분위기를 이어받아 탄생하였다. 이때의 유닉스는 특정한 하드웨어가 종속되지 않는 하나의 운영체계로 행해졌으며, 저렴한 컴퓨터에 맞게 만들어졌다.

소스코드도 함께 배포되었으며, 많은 사람들이 고치고 수정해서 기능을 향상시켜 사용했다.

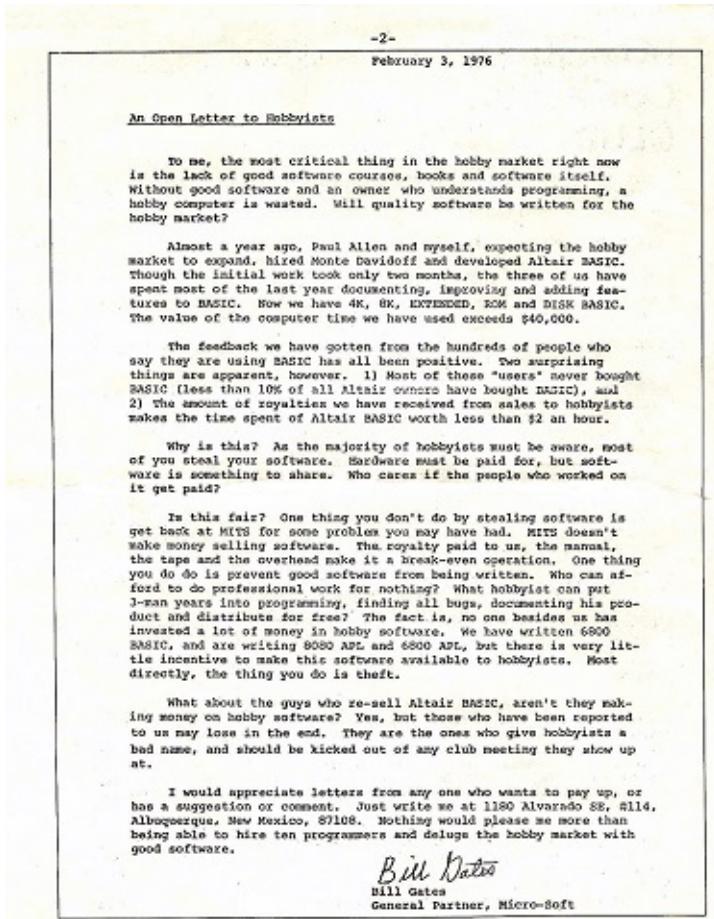
그 결과 AT&T를 비롯한 여러 회사들과 버클리대학등 비영리 단체들이 유닉스를 수정해서 BSD를 배포하였다.

### • 1970년대 중반

1970년대 중반, 해커들과 컴퓨터를 취미로 하던 사람들이 실리콘밸리에 모여 홈브루 컴퓨터클럽을 결성하였다.

이때까지만 해도, 소프트웨어에 소유권에 대해서는 별 생각없이 마음대로 사용이 가능했다.

그러나 1976년 1월 31일, 빌게이츠가 이 회원들에게 독점 소프트웨어라는 새로운 개념을 편지로 보내 설명하면서 뒤바뀌기 시작했다. 편지의 이름은 'Open Letter to Hobbyists'이다.



"취미시장에서 나에게 중요한건 좋은 소프트웨어 교육과 책과 소프트웨어이며, 좋은 소프트웨어와 프로그래밍을 이해하는 소유자가 없다면 취미컴퓨터는 소모적이 될것이다."

양질의 소프트웨어가 취미시장을 위해 쓰여지지 않는다. BASIC을 사용해 본 사용자들은 긍정적인 반응이었으나 실제 사용자들 대부분은 BASIC을 구매하지 않았고, 취미가들에게 얻은 로열티의 합은 시간당 2달러의 가치도 없었다.

이는 취미가들이 소프트웨어를 훔치고 있는 것이다. 하드웨어는 돈을 주고 사는데 소프트웨어는 나눠쓰고, 이 소프트웨어를 만든 사람이 댓가를 받는지 신경도 안쓰는 상황이 공정한가?

취미가들은 좋은 소프트웨어가 쓰여지는 것을 막고 있다. 댓가없이 전문적인 일을 누가 하겠는가? 우리를 제외하고 아무도 취미 소프트웨어에 큰 돈을 투자하지 않는다."

## • 1970년대 후반

결국 1970년대 후반에 이 편지로 인해 소프트웨어를 감추기 시작하며 소스코드를 함부로 변경하지 못하게 되었다.

개인용 컴퓨터 발전과 보급이 소프트웨어 상품 시장이 폭발적으로 성장하면서 소프트웨어는 더 이상 자유롭지 못하게 만든 것이다.

일례로 유닉스는 1979년에 유료가 되었다.

## 2.3 GNU의 설립

리눅스 탄생 이전, 리처드 스톤만이라는 사람과 자유소프트웨어 운동이 먼저 존재했다.

리처드 스톤만은 자유소프트웨어 운동의 창시자이다. 그는 GNU운영체제를 만들려는 과정에서 자유소프트웨어 운동의 토대를 구축했다. 그는 리눅스와 오픈소스가 있게 해준 인물이다.

(인터뷰: BRUCE PERENS - Open Source Definition의 저자)

오픈소스는 컴퓨터의 시작과 함께 시작했습니다. 맨 초기의 소프트웨어는 사람들 사이에 돌아다녔기 때문입니다.

1970년대 후반에서 80년 초반이 되어서야 소프트웨어를 감추며 소스코드를 변경하지 못하게 했습니다. 그 원인 중에 하나는 독점적인 소프트웨어모델을 만든 선구자인 마이크로소프트 때문입니다.

(인터뷰: ERIC RAYMOND - 성당과 시장의 저자)

1970년대 초반과 1980년대 후반동안 리처드 스톤만은 독점 소프트웨어가 가질 수 있는 부정적인 요소는 다 경험했습니다. 예를 들어, 그가 일하고 싶었던 회사들은 문을 열어주지 않았고, 코드를 가진 회사는 리처드 스톤만이 코드를 고치는 것이 이익임에도 불구하고 허락하지 않았습니다.

(인터뷰: Richard Stallman - GNU Project의 설립자)

그 상황은 저에게 도덕적 딜레마였습니다. 1980년대 초반 최신 컴퓨터를 구한다는 것은, 독점적인 운영체제를 구해야 한다는 것을 의미합니다. 그 운영체제의 개발자들은 나누려 하지 않고 사용자들을 통제하고 지배하고 제한합니다. 시스템을 얻으려면 공유하지 말아야 한다는 약속에 서명해야 했습니다. 저에게 그것은 나쁜 사람이 되기를 강요받는, 세계의 일부분을 속이고 저를 협동적인 공동체 사회에서 단절되라는 약속이었습니다. 그리고 그것은 연구소에 안 좋은 영향을 미쳤습니다. 예전처럼 유용한 일을 못하게 되었지요. 따라서 저는 그것에 반대했습니다.

(인터뷰: ERIC RAYMOND - 성당과 시장의 저자)

결국 리처드 스톤만은 지적재산을 가진 소프트웨어에 관한 적의를 키워 '자유소프트웨어재단'을 설립하였습니다.

(인터뷰: Richard Stallman - GNU Project의 설립자)

저는 제가 운영체제 개발자임을 깨달았습니다. 제가 다른 운영체제를 개발할 것이라면, 개발자로써 모든 사람이 공유할 수 있도록 장려할 것입니다. 다른 사람들로부터 널리 퍼져 나간다면, 도덕적 딜레마로부터 벗어날 수 있으며, 이것이 제가 하고자 했던 일임을 깨달았습니다. 1984년 1월에 GNU운영체제 개발 프로젝트를 시작했습니다. GNU는 GNU's Not Unix의 줄임말이며, 이 뜻은 제가 Unix운영체제와 비슷한 시스템을 개발하고 있었으나 명백히 다른 시스템임을 의미합니다. 유닉스는 독점적이므로 우리에게 쓸모가 없어 대체품을 작성한 것입니다.

1980년대, 리처드 스톤만이 GNU프로젝트를 진행하는 동안 캘리포니아 대학 버클리 분교 컴퓨터 과학자들은, '버클리 유닉스' 또는 'BSD'라고 알려진 AT&T로부터 라이센스 받은 유닉스 커널에 기초한 자유로운 운영체제를 개발하고 있었습니다. 그러나 AT&T와 법적문제, 소스코드의 분열로 채택 받는 것이 늦어졌습니다.

(인터뷰: Richard Stallman - GNU Project의 설립자)

유닉스는 많은 수의 서로 분리되어 통신을 주고받는 프로그램들로 구성됩니다. 저는 이런 프로그램들을 모두 교체해야 했습니다. 대체 프로그램을 계속 작성했습니다. 그때 제가 프로그램을 작성하는 걸 도와달라고 요청한 걸 사람들이 보고 와서 제 작업에 동참하기 시작했습니다. 그리고 1991년도에 모든 대체 작업이 끝났습니다. 완성한 프로그램들의 예로는, 다른 프로그램들이 사용하는 자원을 할당하는 커널이라는 프로그램이 있으며, 사람이 읽을 수 있는 숫자로 된 소스코드로부터 컴퓨터가 실제로 실행시킬 수 있는 미스테리한 숫자들로 된 프로그램으로 번역해주는 프로그램인 컴파일러가 있습니다. 컴파일러와 함께 이런 일을 할 수 있는 다른 프로그램들이 필요했습니다. 디버거나, 텍스트편집기, 텍스트 포매터, 메일 발송 프로그램 등 말입니다. 유닉스와 비슷한 운영체제에는 수많은 프로그램들이 있습니다.

(인터뷰: MICHAEL TIEMANN, Cygnus Solutions의 공동 설립자)

리차드 스톤먼과 1987년 2월에 만났습니다. 5일짜리 Emacs 강연을 하러 왔습니다. 저녁이 되자 리차드 스톤먼은 바쁘게 컴파일 러로 작업을 했고, 그때 바로 공개 배포는 하지 않았습니다. 누구한테 소스코드를 보여줄지 조심스러웠습니다. 6월 공개하는 날 즉시 다운로드하였고, 그것을 가지고 놀았습니다. 그로부터 몇 가지 지침을 얻었고, 제가 리차드 스톤먼에게 소스코드를 보냈을 때 그는 제가 어떻게 이 기술을 빨리 이해했는지 놀라워했습니다.

(인터뷰: Richard Stallman - GNU Project의 설립자)

GNU의 본질은 자유 소프트웨어라는 것입니다. 원하는 대로 바꿀 수 있고, 복사본을 재배포하고, 공유하고, 개선하여 다른 사람들 역시 이익을 얻는 것입니다. 이 행위는 사람들이 공동체를 구성하는 것을 가능하게 해줍니다.

(인터뷰: BRUCE PERENS - Open Source Definition의 저자)

자유 소프트웨어는 저작권을 가지고 있습니다. 소유자가 있고 라이센스가 있습니다. 일반 퍼블릭 도메인이 아닙니다.

(인터뷰: Richard Stallman - GNU Project의 설립자)

소프트웨어를 퍼블릭 도메인에 둔다면, 다른 누군가가 수정을 하고 그것을 독점 소프트웨어 패키지로 만들 수 있습니다. 협력하고 공유하는 자유를 갖지 못함을 의미합니다. 그것을 막기 위해 저작권을 사용합니다. 해당 소프트웨어의 저작권이 보호받고, 저작자들은 복사본의 재배포를 허용하고 수정권한을 주지만, 재배포할 때 저작권의 조항에 의거해야 합니다. 해당 조항들로 인해, 다른 사람들과 협력할 수 있는 자유를 가지게 됩니다.

GNU/Linux와 자유소프트웨어 성장에 있어 중요한 건 소프트웨어와 철학에 기초한 사업 창출이었다. 사업의 시작을 위한 발화점은 스탠포드 대학의 Electronics Research Lab입니다.

ERL로 알려진 이 연구소는 최초의 GNU와 리눅스 비즈니스 창안자가 영감을 얻은 곳이다. 마이클 티만은 GNU 자유소프트웨어에 관련한 컨설팅과 서비스를 판매할 생각으로 '시그너스 소프트웨어'를 창업했습니다.

리차드 스톤먼은 돈을 벌 수 있는 여러 가지 다양한 방법들을 제안했습니다.

(인터뷰: Richard Stallman - GNU Project의 설립자)

자유소프트웨어 운동의 초기부터 수익을 낼 수 있다고 저는 믿었습니다. 자유소프트웨어의 이점 중 하나는 서비스나 지원을 위한 자유로운 시장이 존재한다는 점입니다. 사업에서 소프트웨어를 사용하고 좋은 지원을 원한다면, 부를 사람들을 선택하고 기술 지원을 제공하는 사업을 하는 사업자를 골라잡을 수 있습니다. 독점 소프트웨어는 지원이 독점체계이고 소스를 가진 회사가 하나라도 그들만이 지원을 할 수 있어 횡포의 위험이 있습니다. 마이크로소프트의 경우가 그렇습니다. 그들의 지원이 나쁜 것은 당연한 건지도 모릅니다.

(인터뷰: MICHAEL TIEMANN, Cygnus Solutions의 공동 설립자)

자유소프트웨어의 이점은 엄청나지만 시그너스 회사에서 이를 지원하기 위한 비용은 관리직 직원들을 불안하게 하였습니다. 거기에 제가 낸 아이디어는 사내의 엔지니어가 제공해주는 것보다 2~4배가 되는 기술지원과 Hand Holding Capability를 제공해주는 모델을 만드는 것이었습니다. 사람들이 실제로 구매를 할지 결정이 나는 단가의 1/2에서 1/4 수준에서 완성이 가능했습니다. 모든 준비가 끝나 1989년 12월 사업인가를 받았습니다.

(인터뷰: BRUCE PERENS - Open Source Definition의 저자)

시그너스는 자유소프트웨어를 특기로 삼는 첫 사업입니다. 자유 소프트웨어를 지원했고, 뛰어난 소프트웨어를 무료로 구할 수 있었으나 지원을 얻을 수는 없었고 시그너스는 지원사업으로 돈을 벌었습니다.

(인터뷰: ERIC RAYMOND - 성당과 시장의 저자)

GNU 프로젝트는 툴킷을 작성하면서 시작했는데, C 컴파일러, 디버거, 문서 편집기와 같은 기본적인 개발 도구들, 다른 유용한 소프트웨어들을 개발하는 것이었습니다. 그리고 프로젝트의 목표는 이런 소프트웨어들 아래에서 동작하면서 운영체제의 중심이 되는 커널을 개발하는 것입니다. 1990년에 GNU 프로젝트는 목표한 툴킷을 개발하는데 성공했고, 다양한 유닉스 운영체제들에서 널리 사용되었습니다. 그러나 아직 자유소프트웨어에 속하는 커널은 없었습니다.

(인터뷰: Richard Stallman - GNU Project의 설립자)

커널개발은GNU프로젝트의 개발 과정 중 마지막 부분에 속하는 것이었습니다.커널개발이 얼마 되지 않았을 때,리눅스 토발즈가 나타났습니다.그는 하나의 커널을 개발했고 그 커널은 훌륭하고 안정적으로,빠른 속도로 작동했습니다.그것이Linux라고 불리는 커널이었습니다.

(인터뷰: LINUS TORVALDS -리눅스 커널의Creator)

초기 목적은 매우 개인적이었습니다.대학 컴퓨터에서 익숙한 것과 유사한 것을 제PC에서 실행시킬 수 있으면 좋겠다는 목적이었습니다.초반의 영감은 제가 헬싱키 대학에서 쓰던 컴퓨터의SunOS에서 얻었습니다.

(인터뷰: ERIC RAYMOND -성당과 시장의 저자)

1991년부터1993년까지는 리눅스의 유년기였습니다.비교적 불안정한 상태였습니다.

(인터뷰: Richard Stallman - GNU Project의 설립자)

리눅스는 전통적이고 검증된 방식을 사용했습니다.주어진 작업을 맡아서 처리하는 하나의 프로그램을 만들었습니다.이런 방식을 사용한 개발은 예상보다 빨리 이루어졌습니다.

(인터뷰: LINUS TORVALDS -리눅스 커널의Creator)

이런 방식을 모놀리식이라고 하는데,기본적으로 운영체제 자체가 분할할 수 없는 하나의 실제라는 것을 의미합니다.리눅스는 GNU프로젝트와는 여러 차원의 걸쳐 관계를 갖고 있습니다만,자신의 소스코드를 공개하는 것은 좋은 생각이라는 공통된 의견이 주를 이루고 있습니다.

(인터뷰: Richard Stallman - GNU Project의 설립자)

리눅스는GNU프로젝트 차원에서 커널을 개발한 것이 아니었습니다.독자적으로 개발,발표한 커널이 리눅스고 저(GNU설립자)는 그 사실을 알지도 못했습니다.리눅스의 커널에 대해 알고 있던 사람들은 이용할 수 있는 커널을 찾고 있었고,리눅스의 커널을 이용해서 완전한 시스템을 만들려고 했습니다.그리고 완전한 시스템을 갖추기 위해 필요한 구성요소들이 이미 이용 가능한 상태였으나,실제로 이들은 완전한 시스템을 구성할 기회를 갖지는 못했습니다.

왜냐하면 이들은 단지 커널만 빠져 있었던GNU시스템의 모든 부분들을 하나씩 찾고 있었던 것이기 때문입니다.그래서 그들이 이것들을 하나로 모으려고 했고,실제로GNU시스템에 빠져있던 커널의 자리에 리눅스를 끼워 넣고 있었습니다.

(인터뷰: LINUS TORVALDS -리눅스 커널의Creator)

이런 프로그램들이 많이 있습니다.자유 소프트웨어 재단에 의해 작성된 프로그램들도 있고,리눅스처럼 다른 사람들에 의해 작성된 프로그램들도 있습니다.그리고 리눅스와 이런 프로그램들은 서로 공생관계를 맺고 있습니다.프로그램들이 리눅스 위에서 실행되었고,플랫폼으로서 리눅스를 이용합니다.리눅스는 그 프로그램들이 실행될 수 있도록 하기 위해, GNU C컴파일러를 이용합니다.

(인터뷰: Richard Stallman - GNU Project의 설립자)

제가 어려서 학교 다닐 때 선생님들은 우리에게 같이 나누어야 한다고 가르쳤습니다.

그러나 요즘 행정부는 선생님들이 아이들에게 라이센스 관행을 받아들이도록 가르치게 하고 있습니다.만약 소프트웨어를 학교에 가지고 오면 나누는 것은 도둑질이며,감옥에 가게 되니까 나눠 쓰지 말라고 합니다.이것은 사회가 지향해야 할 바가 아닙니다.우린 선한 의지가 필요합니다.도와주는 것이 어렵지 않다면,다른 사람들을 도와줄 수 있는 착한마음이 사회의 기본입니다.도와줄 수 있는 착한 마음이 없다면 무법의 정글이라고 할 수 있겠죠.

Q:사람들이 만약 무차별 복사로 이익을 얻어 창조적인 작업이나 소프트웨어 등을 없애버린다면?

A:양면성이 존재합니다.사람들은 자유소프트웨어를 개발해서 이익을 낼 수 있습니다.그러나 공동체를 가질 수 있는 자유가 더 중요하다는 것입니다.

(인터뷰: MICHAEL TIEMANN, Cygnus Solutions의 공동 설립자)

Q:오픈소스나 자유 소프트웨어가 나누어야 하고 선한 마음에서 비롯해서 해야 하는 것이라면,그것을 지켜보는 사람들은 공산주의처럼 들릴 수 있지 않나요?

A: 1989년도에 공산주의라고 말하는 건 사실 좋게 말한 것이라고 볼 수 있습니다. 그 당시 사람들은 "미쳤다"라는 말을 썼습니다.  
저는 그 사람들이 자본주의라는 단어를 쓰면 좋겠습니다.

(인터뷰: ERIC RAYMOND - 성당과 시장의 저자)

Q: 오픈소스나 자유 소프트웨어가 나누어야 하고 선한 마음에서 비롯해서 해야 하는 것이라면, 그것을 지켜보는 사람들은 공산주의처럼 들릴 수 있지 않나요?

A: 절대 아닙니다. 공산주의 이념은 나눔을 강제하고 있습니다. 오픈소스는 공산주의가 아닙니다. 결정적으로 오픈소스는 강요를 하지 않습니다.

## 2.4 오픈소스라는 단어의 사용

'자유 소프트웨어'라는 말은 후에 오픈소스로 진화하였다. 시그너스가 오픈소스를 지원하기는 했지만 넷스케이프는 오픈소스에 참여한 첫 대형업체였다.

오픈소스는 마이크로소프트에 대항하기 위한 것이었다. 마이크로소프트는 인터넷 익스플로러를 주었지만 아무도 소스코드를 가질 수 없었고 협력관계를 가질 수 없었다. 마이크로소프트는 독점체계를 이용해서 HTTP와 HTML을 제멋대로 변형했다.

시간이 지나면서 넷스케이프 사업이 위험해졌다. 시장에서 넷스케이프의 제품이 살아남을 수 있도록 하기 위해 해야 하는 모든 일들을 할 만큼 많은 인력이 없었다.

'성당과 시장'(다음 챕터에서 다룬다)은 넷스케이프가 소스코드를 릴리즈하는 것에 큰 영향을 미쳤다. 사실 넷스케이프는 이미 오픈소스를 '릴리즈=공개' 하는 것에 대해 논의를 해왔다.

넷스케이프에서는 "우리가 소스코드를 생각할 때는 product를 만들기 위한 것이 아니라고 자체로 권리를 가진 product로 생각해야 한다."라고 말한다. 오픈소스를 행한 넷스케이프는 '리눅스 운영체제=오픈소스 소프트웨어'의 가장 좋은 사례 중 하나라고 생각했다.

이는 리눅스 운영체제에도 많은 관심을 불러일으켰다. 자유소프트웨어는 좋지 않은 마케팅을 펼쳤고, 경영자들이 좋아하지 않을 말이었다. 이러한 문제와 더불어 사람들은 이 말에서 공짜를 생각한다. 돈을 벌수도 없고 팔수도 없다는 잘못된 생각을 가지게 된다.

소프트웨어가 공개되어있고, 소스코드를 사용할 수 있다는 개념으로 넘어서고자 했다. 자유소프트웨어가 오픈소스의 개념으로 확장된 것이다.

크리스틴 피터슨이 제시한 오픈소스의 개념은 리눅스에게 전달되었고 리눅스는 이 개념을 좋아했다. 자유소프트웨어를 오픈소스로 대체하였고 이는 오픈소스의 시작이 되었다.

한편으로는 기존 자유소프트웨어에 대해 일부 사람은 상업화에 대해 공포를 가지고 있었다.

리처드 스톤만과 오픈소스 정의의 저자인 사람과 차이는 모든 소프트웨어는 자유로워야한다는 것과 자유소프트웨어가 아닌 것이 계속 있을 수 있으며, 이것은 자유소프트웨어와 공존 가능하다는 것이다.

오픈소스의 정의에 따른 9가지 권리는 아래와 같다.

1. 자유롭게 재배포 가능하다. 이는 해방을 의미한다. 즉, 가격도 알아서 정할 수 있다.
  - i. 소스코드는 이용 가능해야 한다. 다른 사람이 유지관리를 할 수 있고 기존에 있던 것을 이어서 개발 가능하게 한다.
  - ii. 파생작업이 가능해야 한다. 프로그램을 개선하고, 결과물 재배포를 할 수 있어야 한다.
  - iii. 저자의 오픈소스 통합이 가능하다. 저자의 명예를 유지하기 위해 소스를 가져와서 바꿨다는 걸 밝혀야 한다.
  - iv. 사람과 그룹을 차별하지 말아야 한다. 사람과 그룹의 차별 없이 모두가 소프트웨어를 사용할 수 있어야 한다.
  - v. 분야나 목적에 대해 차별하지 말아야 한다. 사업 분야든 어디든 모든 분야에서 사용 가능해야 한다.
  - vi. 라이센스는 배포가 가능해야 한다. 라이센스를 줄 수 있고 제3자도 다른 사람에게 배포 가능하나 동일한 라이센스여야 한다.
  - vii. 라이센스는 어떤 제품에 특정하면 안된다. 다른 시스템에서도 배포가 가능해야 한다.
  - viii. 라이센스는 다른 소프트웨어를 오염시킬 수 없다. 동봉되어 있는 다른 프로그램까지 자유롭게 되는 것은 아니다.

## 2.5 성당과 시장

에릭 레이먼드가 쓴<성당과 시장>(The Cathedral and the Bazaar)은 자유 소프트웨어 철학을 대변하는 유명한 글이다. 레이먼드는 이 글에서 리눅스 커널 개발과정과 페치메일이라는 오픈 소스 프로젝트를 개발한 경험을 들어 오픈 소스 개발 방식의 유효성을 설명한다. 1997년 5월 27일 리눅스 회의에서 처음 공개되었으며, 1999년 같은 이름의 책에 포함되어 출판되었다.

글에서 저자는 두 가지 방식의 자유 소프트웨어 개발 모델을 대조한다.

- 성당 모델: 출시 때에만 소스 코드를 공개하고 그 사이에는 제한된 개발자들만 소스 코드에 접근할 수 있다. GNU이맥스와GNU컴파일러 모음이 예이다.
- 시장 모델: 소스 코드가 인터넷으로 일반에 공개된 상태로 개발된다. 레이몬드는 리눅스 커널 프로젝트의 리더인 리눅스 투르발스가 이 방식을 발명해 냈다고 쓰고 있다. 그는 또한 페치메일 프로젝트에 자신이 이 모델을 적용한 이야기도 쓰고 있다. 글의 주제는 그가 리눅스 법칙이라고 이름붙인 "보는 눈만 많다면, 어떤 버그라도 쉽게 잡을 수 있다"는 명제이다. 이 말은 많은 사람이 테스트하고 훑어보고 실험해 볼 수 있도록 코드가 공개되어 있으면 버그는 빨리 잡힐 것이라 뜻이다. 이에 대해 성당 모델에서는 소스 코드를 여러 명의 개발자들만 볼 수 있으므로 버그를 잡는데에 엄청난 시간과 노력이 든다고 주장했다.

이 글로 인해GNU이맥스와GCC를 포함한 많은 오픈 소스와 자유 소프트웨어 프로젝트들이 시장식의 열린 개발 모델을 사용하게 되었다. 이 중 가장 유명한 것은 넷스케이프가 넷스케이프 커뮤니케이터의 소스를 공개한 것과 모질라 프로젝트를 시작한 것이다.

성당 모델은 상용 소프트웨어의 전형적인 개발 방식이다. 더욱이 상용 소프트웨어는 보통 출시 때 소스가 같이 제공되지 않는다. 그래서 "시장과 성당"이라는 구문은 종종 오픈 소스 소프트웨어와 상용 소프트웨어를 비유하는 데에 쓰인다. 그러나 원래 글에서는 자유 소프트웨어들 중에 두 종류를 비교하여 설명하고 상용 소프트웨어에 대한 언급은 없었다.

성당과 시장의 예는 소프트웨어 프로젝트에만 있는 것이 아니다. 위키백과는 시장식의 프로젝트이고, 누피디아나 브리태니커 백과사전은 성당형의 프로젝트이다. 아래는 성당과 시장의pdf링크이다.

[http://www.oss.kr/oss\\_community\\_ebookoss\\_publication/show/64be57d5-5919-4069-9643-13827b7c69c3](http://www.oss.kr/oss_community_ebookoss_publication/show/64be57d5-5919-4069-9643-13827b7c69c3)

## 2.6 1990년대

### • 1990년 초반

유닉스의 역사는 다소 복잡하게 되었는데 Berkeley 가 주도한 대학 공동체가 Berkeley Software Fistration (BSD) 라 불리는 변종을 개발한 반면 AT&T 는 System III 와 나중에는 System IV 라는 이름으로 계속해서 유닉스를 개발했다. 1980년대 후반에서 1990년대 초반에 이러한 두 가지 중요한 변형들간의 전쟁은 최고조에 달했지만 여러 해가 지난 후에는 각 변형이 다른 변형의 중요 특징들중 많은 부분을 채택하였다. 상업적으로 System V 는 인터페이스의 대부분이 정식 표준에 채택됨으로써 표준 전쟁에서 이겼으며 대부분의 하드웨어 벤더들은 AT&T 의 System V 로 전환하였다. 그러나 System V 도 결국 많은 BSD 혁신들을 받아들이게 되었으며 그래서 결과적으로 시스템은 두 시스템의 합병 이상이었다. BSD 시스템도 사라지지 않았으며 대신 연구, PC 하드웨어와 단일 목적 서버 (예, 많은 웹 서버들은 BSD 에서 파생된 시스템을 사용하고 있다) 용으로 널리 사용되게 되었다.

### • 1990년

FSF(자유 소프트웨어 재단-리처드 스톤만)는 운영체제 커널을 개발하는데 어려움을 겪었으며, 만약 커널이 없다면 그들의 나머지 소프트웨어들은 소용없었을 것이다.

### • 1991년



리눅스 토발즈는 당시 소프트웨어 운영체계 '유닉스'의 유료화에 대해 반대했다. 리눅스 토발즈는 대학교에서 쓰던 유닉스를 집에서도 쓰고 싶어서 리눅스라고 이름지은 운영체계 커널을 개발했다. 리눅스라는 이름은 리눅스 토발즈의 리눅스와 유닉스를 합하여 만든 이름이다. 사람들은 "GNU/Linux" 라는 용어를 많이 사용했다. 이는 프리소프트웨어의 개념이 등장을 보여준다.

리눅스 토발즈는 리눅스 커널을 GPL로 공개하여 운영체계 소스코드를 공개해 누구나 자유롭게 쓸 수 있게 하였다.

커널은 자유로이 수정될 수 있고 매우 유용한 운영체계를 만들기 위해 FSF 산물과 다른 컴포넌트들 (특히 BSD 컴포넌트들의 일부와 MIT 의 X 윈도우 소프트웨어) 과 병합될 수 있었다.

리눅스 커널은 공개소프트웨어로 개발되면서 기존의 공개 소프트웨어들과 결합되었으며 안정적 운영체계로 발전하였다.

- 8월: 최초의 리눅스 버전 0.01의 완성

- 10월 5일: 첫번째 공식 버전 0.02 발표

### • 1992년

- 3월: 0.95 버전 발표-> 그래픽 사용자 인터페이스 추가

### • 1993년

리눅스 배포판이 나오기 시작하였다. (배포판: 운영체제의 핵심기능으로 리눅스 커널을 채택하고 여기에 다른 많은 응용 프로그램과 도구 모음을 함께 모아 놓은 운영체제이다. 예로는 레드햇 (Red Hat), 맨드레이크 (Mandrake), 수제 (SuSE), 칼데라 (Caldera), 코렐 (Corel), 데비안 (Debian)등이 있다.) 배포판들간에는 차이점이 있지만 모든 배포판들은 동일한 기반인 리눅스 커널 및GNU glibc 라이브러리들에 기초하고 있다. 영국 맨체스터 대학의 맨체스터 컴퓨팅 센터에서 최초의 배포판인 MCC interim을 출시했다.

### • 1997년

자유 소프트웨어 공동체의 리더 그룹들이 캘리포니아에서 회합을 했다. 에릭 레이먼드(Eric Raymond), 팀 오라일리(Tim O'Reilly),

VA 리서치사 사장인 래리 오거스틴(Larry Augustin) 참석했다. 그 회합의 목적은 기존 자유 소프트웨어라는 개념에 대해 거부감을 가진 사람들에게 자유 소프트웨어를 둘러싼 아이디어들을 진흥시킬 방법을 찾는 것이었다. 에릭 레이먼드의 끈질긴 주장으로, 회합에 모인 사람들은 시장 점유 뿐만 아니라 대중들의 인식까지도 점유할 수 있는 마케팅 캠페인이 부족했다는 점에 동의했다. 회의 도중 새로운 용어인 오픈 소스(Open Source)가 나왔다. 그리고 오픈소스라고 말할 수 있는 소프트웨어에 대한 가이드라인을 만들었다. 상용 소프트웨어의 개발 방식을 성당에, 그누와 리눅스의 소프트웨어 개발 방식을 시장에 비유해 '성당과 시장'이라는 글을 쓴다. 중세에 몇몇 건축가들에 의해 만들어지는 성당의 작업방식을 개발자 몇몇이 폐쇄적으로 개발하는 상용 소프트웨어의 개발 방식으로, 사람들이 복적이라는 시장은 리눅스처럼 인터넷을 통해 누구나 개발에 참여할 수 있는 개발 방식으로 본 것이다.

### • 1998년

에릭 레이몬드와 브루스 패런스가 open source initiative(OSI) 설립하였다. 기존 프리 소프트웨어 대신 오픈소스라는 말을 사용하였다. 또한 그들은 다른 공개 소프트웨어의 운동을 주도하였다.

구분	자유 소프트웨어 재단(FSF)	오픈소스 운동(OSM)
주요인사	리차드 스톤만	에릭 레이몬드
입장	원칙주의	실용주의
목적	자유소프트웨어 발전-사유소프트웨어 제거	오픈소스 소프트웨어발전-사안별로 사적 소프트웨어에 대응 - 반 MS
행태적, 규범적 가정	윤리적 접근법-자유에의 대의에서 출발	이기적 접근법- 자기이익에서 출발
공유저작권	공동체를 확장하는 수단	부당한 사용을 방지하는 수단
보상	비금전적 가치-정보의 자유성 강조	금전적 보상도 강조-정보의 가치성 강조

### • 자유소프트웨어 운동의 문화

구분: 자유 소프트웨어 재단 / 오픈소스 운동

주요인사: 리차드 스톤만/에릭 레이몬드

입장: 원칙주의/ 실용주의

**목적:** 자유 소프트웨어 발전-사유소프트웨어 제거/오픈소스 소프트웨어 발전-사안별로 사적 소프트웨어에 대응-반 MS

**행태적, 규범적 가정:** 윤리적 접근법-자유에의 대의에서 출발/이기적 접근법-자기이익에서 출발

**공유저작권:** 공동체를 확장하는 수단/ 부당한 사용을 방지하는 수단

**보상:** 비금전적 가치-정보의 자유성 강조/금전적 보상도 강조- 정보의 가치성 강조

### • 1999년

SourceForge.net 설립

### • 2000년

오픈소스 시장의 규모는 3000억원의 시장규모에서 꾸준하게 증가했다. 리눅스 시장은 리눅스 os, 애플리케이션, 하드웨어, 임베디드 리눅스 교육 및 출판을 하였다. 최근 임베디드 리눅스의 발전이 급속화 되고 있으며, 그것은 각종 가전기기, PDA, 통신 및 네트워크 장비를 비롯한 공장 자동화, 수치 제어 분야 등에 활용되고 있다.

### • 2004년 ~ 현재

급격한 리눅스 시장이 발전에 발맞추어 리눅스 사용자는 20만명 이상으로 추정된다.

## 2.7 오픈소스의 황금기

### • 1999년 8월

1999년 8월 리눅스 월드를 개최하며 오픈소스의 황금기를 열었다.

많은 사람들이 리눅스 커밍아웃파티라고 불렀으며, 리눅스 초고수들과 하드웨어 해커들이 모인 장소였다. 이 날, 리누스 토발즈는 기조연설을 하였으며 6천여명의 군중들이 정오부터 줄을 서기 시작했다.

또한 리눅스월드에서 리차드 스톤먼이 리누스 토발즈 상을 받아 연설을 시작했다.



"리누스 토발즈 상을 자유 소프트웨어 연합에 준다는 것은, 스타워즈의 Han Solo 상을 반란군에게 준거나 마찬가지입니다. 참으로 아이러니한 일이군요. 제가 어떻게 여기까지 왔는지 말씀드리겠습니다.

15년 전에 컴퓨터를 사용하려고 했다면, 유일한 방법은 폐쇄적 소프트웨어를 사용하는 것이었습니다. 사용자를 나누고 통제하는 소프트웨어 말이죠. 많은 사람들은 싫어했지만 대안이 없어 어쩔 수 없이 사용했습니다.

따라서 우리들 중 일부는 대안을 만들겠다고 다짐했습니다. 자유로운 운영체제를 개발해야겠다고 말했습니다. 프리 소프트웨어 운영체제, 유저들이 컴퓨터를 사용하면서 자유를 가질 수 있는 기회를 제공하는 것이죠.

많은 사람들은 "아이디어는 좋지만 너무 어려워서 결코 안될 거야, 따라서 참여하고 싶지도 않아."라고 했습니다. 그러나 모든 사람들이 그렇게 말한 것은 아니었죠. 분명히 커널은 완료할 수 있다는 것을 알고 있었습니다.

그러나 진행도중 다른 사람이 더 나은 커널을 우리보다 빨리 만들었습니다. 옛날에는 전략이 있었습니다. 사람들에게 자유의 중요성을 부각시켜 주의를 환기시키자는 것이죠.

컴퓨터를 쓸 때 가질 수도 있고 잃을 수도 있는 자유, 어떻게 그런 일을 할 수 있을까요? 이 전략이 실제로 가능하도록 바꿀 수 있는 방법은 사람들이 사용하고 있는 운영체제가 GNU 시스템이라는 것을 널리 알리는 것이었습니다.

사람들이 이를 알게 되면, 왜 우리가 이 시스템을 만들었는지 알아볼테고, 이 문제에 대해 생각해 보겠죠. 부탁드립니다. 다른분 들에게 이것이 GNU 시스템이라는 것을 알려주세요. 이것은 GNU와 리눅스의 조합입니다. GNU/LINUX라고 부를 수 있습니다."

마지막으로, 이 달에 리눅스 회사에서 레드햇 소프트웨어 주식이 상장되었는데, 228%가량 폭등한 리눅스 운영체제 기반의 소프 트웨어였다. 레드햇 소프트웨어에 관한 비하인드가 있는데, 이 소프트웨어의 개발자들이, 주식이 폭등하였음에도 제대로 된 몫을 받지 못하였다. 그러나 개발자들은 코드가 필요해서 개발한것이라 신경도 쓰지 않았다고 한다.

1999년 12월 9일, VA 리눅스가 상장되어 거래가 시작되었다. 가격이 30달러에 형성되었으나 320달러 이상으로 솟아 올랐으 며 종국에는 766%가 오른 235~265달러로 역사상 가장 잘 된 기업공개로 끝이 났다.

## 3. 오픈소스의 문화

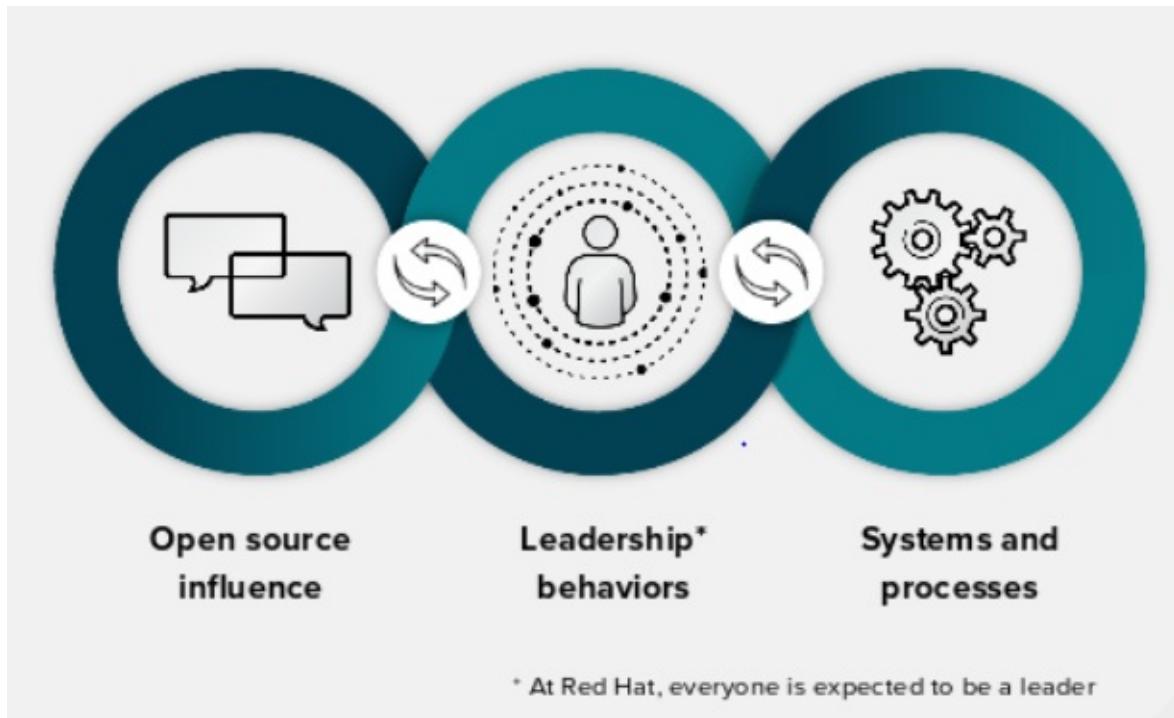
### 3.1 Intro

현대에 이르러 특수한 경우를 제외하면 오픈소스를 활용하지 않은 개발은 찾아보기가 어렵다.

예전에는 뜻이 맞는 개발자끼리 소스를 공유하거나, 대형 재단의 주도로 특정 라이센스에 맞게 오픈 소스 프로젝트가 이루어 졌다면, 현대에는 다양한 방법으로 다양한 기관이나 회사, 혹은 개인이 수요에 의해 자발적으로 주도하는 모습이다.

또한 오픈소스 소프트웨어의 최대 장점은 사용자가 프로그래밍 언어를 알기만 하면 직접 소프트웨어의 문제를 수정하거나 개선을 할 수 있어 소프트웨어의 완전함이나 안정성이 더욱 높아진다. (git을 통한 공유를 한다면) 일반적인 closed 소스 소프트웨어와는 비교도 할 수 없을 정도의 품질을 보여준다.

지금부터 여러 분야에 걸친 오픈소스 프로젝트들을 간략하게 알아보고, 현대 오픈소스의 문화와 현황을 확인해 보자.



## 3.2 Programming Language

현대에 이르러서 만들어진 프로그래밍 언어는 대부분 오픈소스 프로젝트로 관리되어지고 있다.  
아래 다섯개의 프로그래밍 언어는 대부분 10년 내에 만들어진 언어들이다.

**Go** - [Github](#) | [Home](#)

# GOLANG



구글이 2009년에 발표, 2011년에 정식 출시한 프로그래밍 언어이다.

다른 언어의 긍정적인 부분을 유지하고, 공통적인 문제들을 해결하기 위해 만들어진 언어인 만큼 대형 프로젝트에서의 생산성이 뛰어난 장점이 있다.

2015년에는 Go로 만들어진 Go 컴파일러가 만들어지고, 최근에는 모듈이라는 패키지 관리 기능이 추가되어 더욱 완성도가 높아졌다.

현재 드롭박스, 클라우드 플레이어, 넷플릭스, 사운드 클라우드, 트위치, 우버 등 수많은 회사들이 채택해 사용하고 있으며 구글의 수 많은 프로젝트에서 사용되고 있다.

현재는 Go2가 개발중에 있다고 한다.

**Kotlin** - [Github](#) | [Home](#)



Pycharm을 만든 개발사 JetBrains에서 2011년에 공개한 프로그래밍 언어이다.

Java에 비해 개발자 중심의 간결한 문법과 기능들이 제공된다. 하나의 예는 Kotlin의 변수는 Nullable과 NotNull로 나누는데, 이를 '?>' 하나로 바꿀 수 있다.

JVM 기반의 언어이기에 Java와의 상호 운용이 가능하며 안드로이드, JS, JAVA, HTML5, iOS, 라즈베리 파이 등 다양한 기기를 개발할 때 사용할 수 있다.

2017년에 구글에서 안드로이드 공식 언어로 채택해 크게 인기가 올라갔다.

### **Swift - [Github](#) | [Swift](#)**



애플이 WWDC 2014에서 공개한 iOS와 macOS를 대상으로 한 프로그래밍 언어이다.

초기에는 swift의 특징은 Fast, Modern, Interactive였으나, 이후 Safe, Fast, Expressive로 바뀌었다.

기존 Objective-C의 단점을 보완하고, 각종 기능을 보완했다는 좋은 평이 많지만, 버전업을 함께 따라갈수록 하위호환성이 어려워진 점은 단점 중 하나로 꼽히고 있다.

2015년에 차기 버전을 공개하면서, OS X, iOS, Linux용으로 아파치 라이센스 2.0인 소스코드를 공개했다.

### **Python - [Github](#) | [Home](#)**



생각보다 오래된 1991년에 만들어진 언어. 2008년에 Python3이 나왔다.

파이썬이라는 이름은 개발자가 좋아하는 코미디 프로그램에서 따왔고, 크리스마스에 심심해서 만들었다는 이야기가 있다.

현재는 비영리 파이썬 소프트웨어 재단이 관리하고 있고, 개방형, 공동체 기반 개발 모델을 갖고 있다.

문법이 쉽고, 생산성이 높아 사용자가 많은 언어 중 하나다. 다양한 기능을 가진 모듈이 오픈소스로 널리 풀려 있기에 초보자부터 전문가까지 넓은 사용자 층도 갖고 있다.

파이썬을 이용한 CPython이나 Stackless Python, 자바 가상머신 위에서 돌아가는 Jython 외에도 수많은 인터프리터들이 존재한다.

디자인 철학이 존재할 만큼 스타일이 확실한 언어 중 하나.

소프트웨어학부 1학년 1학기 필수 과목으로 채택되는 영광을 가졌다.

[R - Cran-Github | Home](#)



1993년 오클랜드 대학교에서 개발한 통계, 그래프 작업용 인터프리터 프로그래밍 언어.

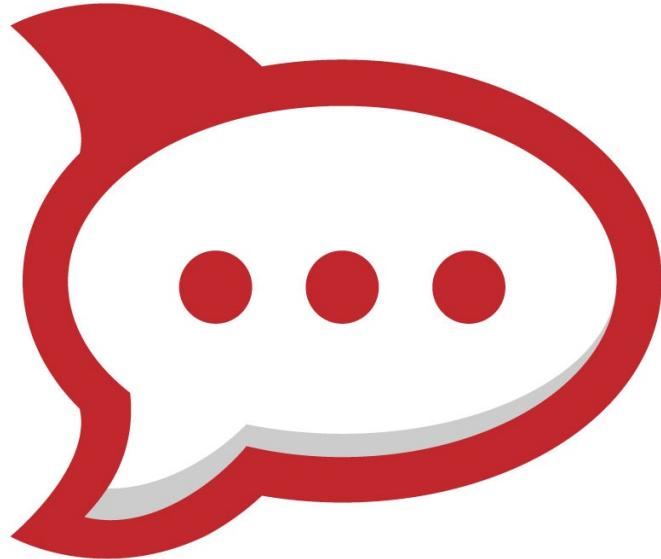
R을 본떠 설계한 pandas라는 Python Library가 존재한다.

또한 R에서 사용 가능한 수많은 통계 관련 패키지가 개발되어 있어 기능 확장이 수월하다.

GNU 라이센스를 따르며, 최근까지 업데이트가 진행되고 있고, 컴퓨터 성능이 향상된 현재에 들어서 더욱 빛을 보고 있다.

### 3.3 Programming Framework & Library

Rocket Chat - [Github](#) | [Home](#)



# ROCKET.CHAT

패키지 방식을 도입한 슬랙과 같은 오픈소스 웹 채팅 플랫폼.

모듈이 도입되어 있기 때문에 다양한 기능을 지원하는데, 실시간 번역이나 이모지 등은 기본이다.

다양한 OS에서 지원이 가능하다. 자체 채팅 플랫폼을 구축하려는 회사나 단체에서 인기가 많다.

Node.js - [Github](#) | [Home](#)

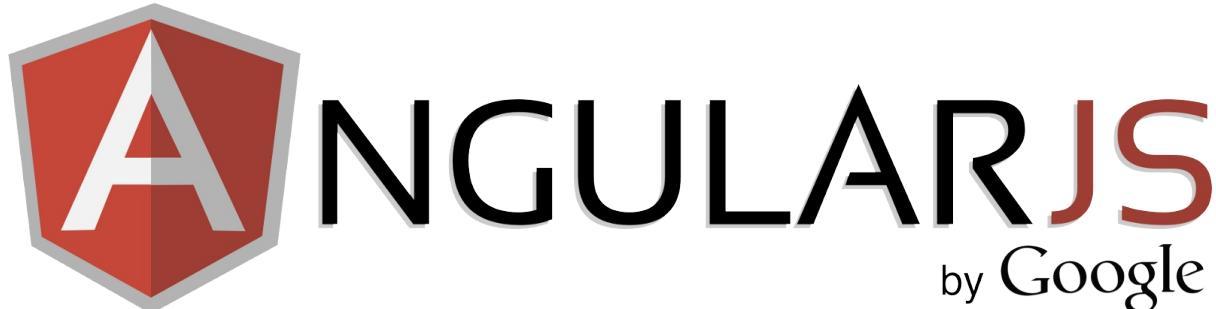


웹 서버와 같이 확장성이 있는 프로그램을 제작하기 위해 고안된 Framework.

작성 언어로 JS를 활용하며, 내장 http서버 라이브러리를 포함하고 있어 웹 서버에서 아파치 등의 별도의 소프트웨어 없이 동작하는게 가능하다.

2009년에 처음 고안되었고, 2011년 마이크로소프트와 파트너쉽을 맺게된다. 현재는 v8 버전이 출시되어 있다.

### AngularJS - [Github](#) | [Home](#)



자바스크립트 기반의 오픈소스 Front-End Web Framework.

2010년에 구글이 발표했고 MIT라이센스의 오픈소스이다.

웹 페이지 개발중에 마주치는 문제를 해결하고, 테스트를 수월하게 하기 위한 목적으로 개발되었다.

HTML은 정적인 마크업 언어이지만, AngularJS를 활용하면 HTML을 확장시켜 다이나믹 어플리케이션을 지원하게 된다.

### D3.js - [Github](#) | [Home](#)



웹 브라우저 상에서 동적인 정보 시각화를 구현하기 위한 JS 라이브러리.

2011년에 개발되었고, 2년전인 2016년 6월에 4.0이 개발되었다.

뉴욕 타임즈에서는 interactive 기사를 제작할 때 D3.js를 적극 활용하고 있다.

**Scratch - Github | Home**



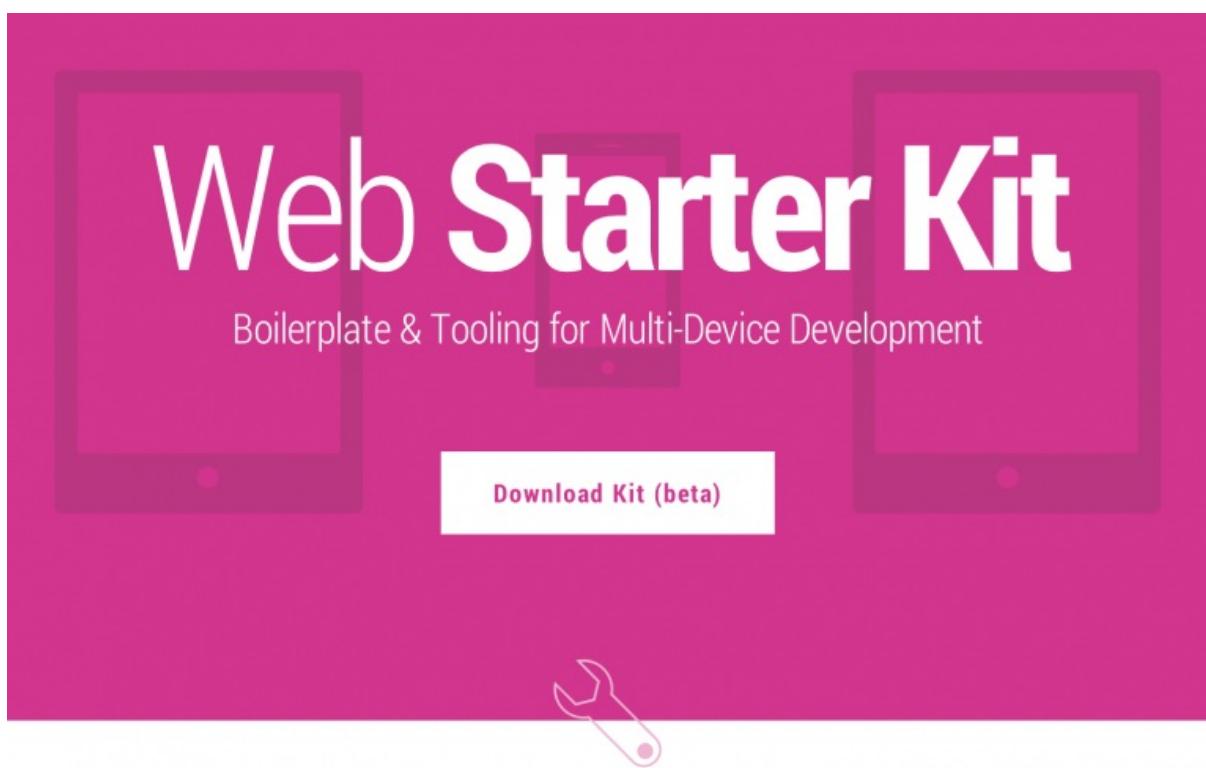
아이들에게 컴퓨터 코딩에 관한 경험을 쌓게 하기 위한 목적으로 설계된 교육용 프로그래밍 언어 및 환경.

블록을 이용해 코딩을 하기 때문에 입문자들도 쉽게 코딩할 수 있다.

MIT Media Lab에서 2006년 발표했고, 2009년 1.4 버전이 발표되었다.

1.4 버전은 GNU 라이센스로 오픈된 오픈 소스이다.

### **Web Starter Kit - [Github](#) | [Home](#)**



웹 개발을 위해 구글이 발표한 성능 지향적 Front-End Framework.

Node.js를 이용한 각종 기능들이 제공되고, 구글이 권장하는 가이드라인대로 세팅되어 있다.

아파치 2.0 라이센스로 오픈되어 있는 오픈소스.

### **Webkit - [Github](#) | [Home](#)**



웹 브라우저를 만드는 데 기반을 제공하는 Framework.

MacOS Safari 브라우저의 엔진으로 사용하기 위해 c++으로 개발되었으나, 현재는 안드로이드, iOS 할 것 없이 여러 플랫폼에서 사용하고 있다.

BSD 2.0, GNU LGPL 2.1 라이센스의 오픈소스이다.

### **React Native (React) - [Github](#) | [Home](#) | [exp](#)**



페이스북이 개발한 웹 UI 개발을 위한 자바스크립트 라이브러리.

Reactive는 마이크로소프트가 창안한 개념으로, 비동기 처리 등을 깔끔하게 처리할 수 있는 방식이다.

React는 2011년 페이스북의 뉴스피드에 적용되었고, 2012년에는 인스타그램에 적용되었다.

2013년에 JavaScript Conference US에서 MIT 라이센스로 발표된 오픈소스이다.

React Native는 React의 방법을 모바일로 확장하는 페이스북의 오픈소스이며, 자바스크립트로 작업하면서 인터페이스는 네이티브 위젯으로 표시하는 방법이다.

모바일 환경에서 네이티브 UI를 손쉽게 구현할 수 있다.

## **Yarn - [Github](#) | [Home](#)**



페이스북이 개발한 npm을 대체할 의존성 관리 패키지 매니저.

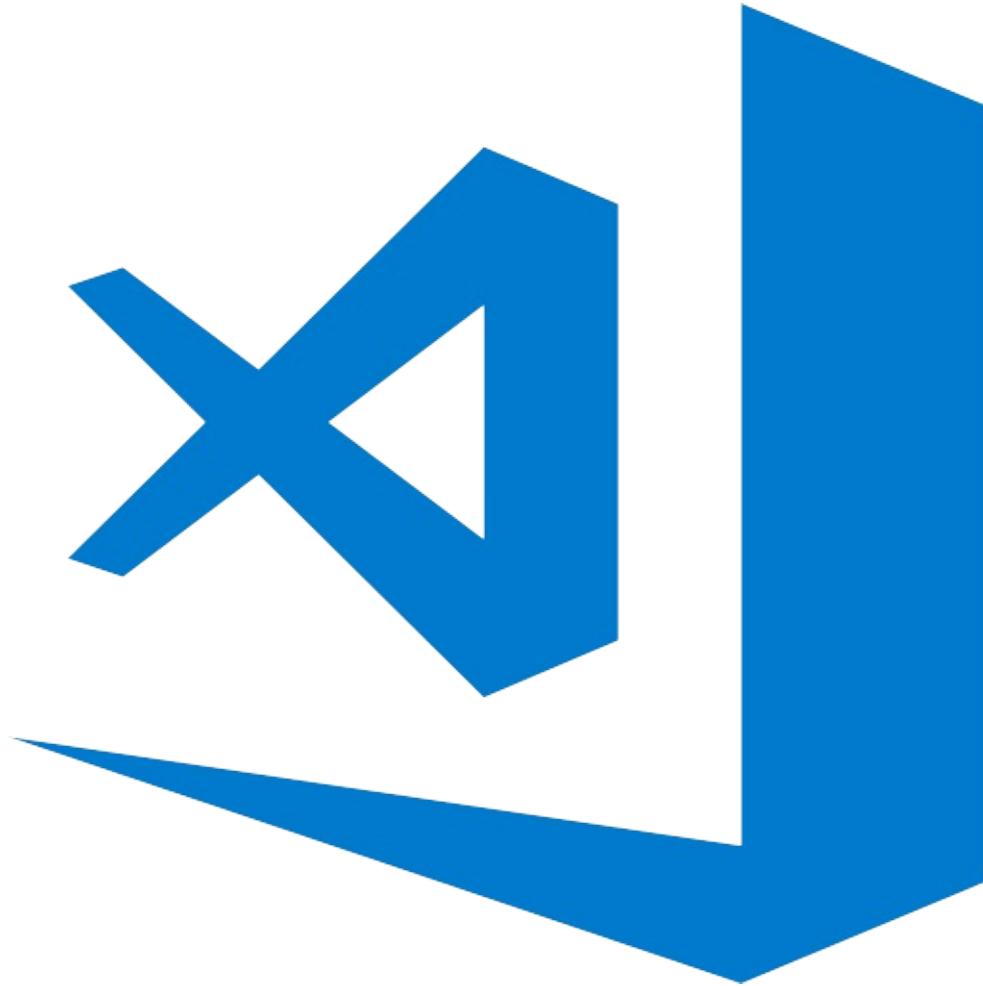
Yarn은 npm에 비해 더 빠르고, 더 안전하다고 알려져 있다. 또한 다양한 명령어를 제공하며, 무엇보다 대규모 프로젝트에 최적화되어 있다.

BSD 2 라이센스를 기반으로 한 오픈소스.



## 3.4 Software

**Visual Studio Code** - [Github](#) | [Home](#)



Microsoft가 Windows, macOS, Linux용으로 개발한 소스코드 편집기이다.

Visual Studio Code는 Github가 개발한 일렉트론 Framework를 기반으로 구동되지만, 일렉트론 기반인 ATOM을 포크하지는 않았다.

2015년 4월 MIT 라이센스로 공개된 오픈소스이다.

**vi(vim)** - [Github](#) | [Home](#)



Emacs와 함께 유닉스 환경에서 가장 많이 쓰이는 문서 편집기.

초기 vi는 1976년에 공개되었고, 현재는 BSD라이센스로 공개된 오픈소스이다.

지금 많이 쓰이는 VIM은 vi에 비해 향상된 기능들을 제공한다. Windows와 Linux, macOS를 포함한 다양한 환경을 지원한다.

Vim Script를 이용해 편집 환경을 변경하거나, 정규식, 문법강조등 다양한 기능이 있지만 처음에 배우기 어렵다는 단점이 있다.

자체 Vim 라이센스를 따르는 오픈소스이다.

**apache - github | Home**



Apache License를 규정한 Apache Software Foundation이 개발한 웹 서버 소프트웨어.

흔히 APM이라 불리는 Apache, PHP, Mysql(Mysqli, MariaDB등으로 대체하는 추세)의 한 프로그램이며, 현재 nginx에 비해 2배가 넘는 점유율을 가진 인기있는 웹 서버 소프트웨어이다.

이 외에도 ASF는 다양한 프로그램을 공개해 오픈소스 프로젝트를 운영하고 있다.

apache는 당연히 아파치 라이센스를 사용하는 오픈소스이다.

### [nginx - Github | Home](#)



2004년 개발된 웹 서버 소프트웨어. 아파치에 이어 현재 사용하고 있는 웹 서버 소프트웨어 점유율 2위를 차지했다.

apache에 비해 가볍고 높은 속도를 목표로 하고 있다. apache에 비해 대규모 웹 서버에서 많이 사용하고 있다.

C로 작성되어있고, BSD 라이센스를 따른다.

### [Filezilla - Github | Home](#)



c++으로 만들어진 다양한 OS를 지원하는 FTP 프로그램.

2001년 2월에 발표되었고, 최근까지도 업데이트가 될 정도로 유명하고, 활발히 진행되는 GNU 라이센스를 따르는 오픈소스 프로젝트이다.

**OpenOffice** - [Github](#) | [Home](#)



OpenOffice는 1999년 Sun Microsystems(현 Oracle에 인수됨)이 인수한 StarOffice에 기반을 두고 있다.

이때 Sun사는 2000년 StarOffice는 MicroSoft Office에 대항하기 위해 소스코드를 공개했다.

2002년에 Apache Software Foundation이 관리를하게 되면서, Open Office로 이름이 바뀌게 된다.

그 후인 2012년에 상표권 분쟁으로 인해 Apache OpenOffice로 개명하게 된다.

C++과 Java로 프로그래밍되어 있으며, 다양한 포맷을 지원한다.

현재는 아파치 라이센스 2.0을 따르는 오픈소스 프로젝트이다.

**LibreOffice** - [Github](#) | [Home](#)



Oracle의 배타적이고 소극적인 탈 오픈소스 정책에 반발한 개발자들이 The Document Foundation을 설립, OpenOffice 3.3.0 버전을 기반으로 LibreOffice를 개발하게 된다.

여러 IT 기업들의 환영을 받게 되면서, 현재는 Ubuntu의 기본 소프트웨어가 되었다.

C++, Java, Python으로 프로그래밍 되어있고, LGPL v3라이센스를 따른다.

**Notepad++** - [Github](#) | [Home](#)



프랑스 개발자 Don Ho가 2003년 개발한 문서 편집기이자 소스 코드 편집기.

C++ 및 Win32 API로 개발되어 상당히 가볍고 빠르다.

Windows OS를 지원하지만, Linux버전인 Notepadqq가 존재한다.

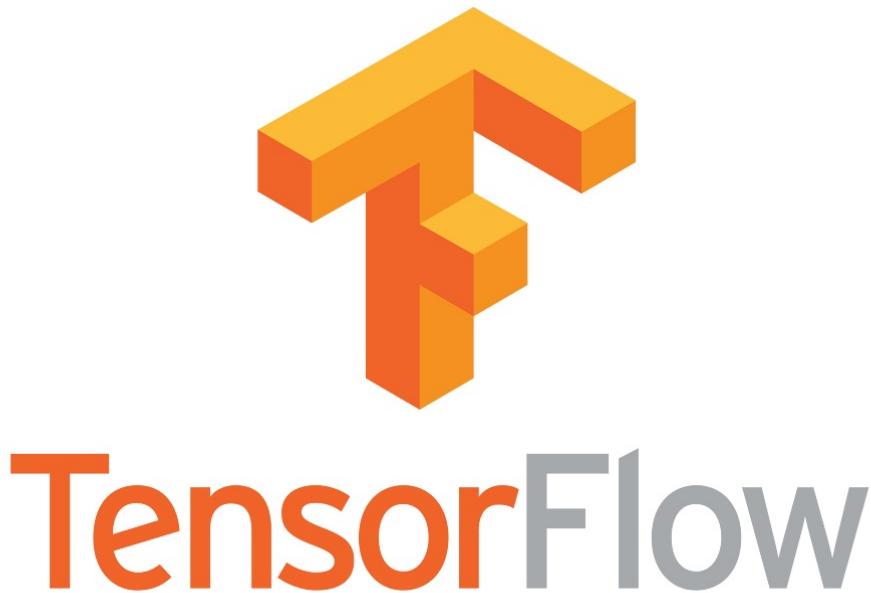
다양한 플러그인들을 제공하며, 지원하는 언어도 상당히 많다.

GNU 라이센스를 따르는 오픈소스이다.



## 3.5 Machine Learning

TensorFlow - [Github](#) | [Home](#)



Google이 Python과 C++를 이용해 개발하고 2015년 발표한 라이브러리.

발표 이후로 Github의 Star를 수천, 수만개를 받아낼 정도의 인기 오픈소스 프로젝트이다. 다수의 머신러닝, 딥러닝(신경망) 모델과 알고리즘을 이용, 활용도를 최대한 높였다.

사실 TensorFlow는 머신러닝만을 위한 라이브러리가 아니고, 대규모 숫자 계산을 위한 라이브러리이다. 실제로 구글은 TensorFlow를 '데이터 흐름 그래프를 사용하는 수치 연산용 오픈소스 소프트웨어 라이브러리'라고 소개하고 있다.

Tensor는 다차원 행렬 계산을 의미하고, Flow는 말 그대로 흐르게 하는 의미이며, Tensor Board를 이용하면 데이터의 흐름을 시각화 할 수 있다.

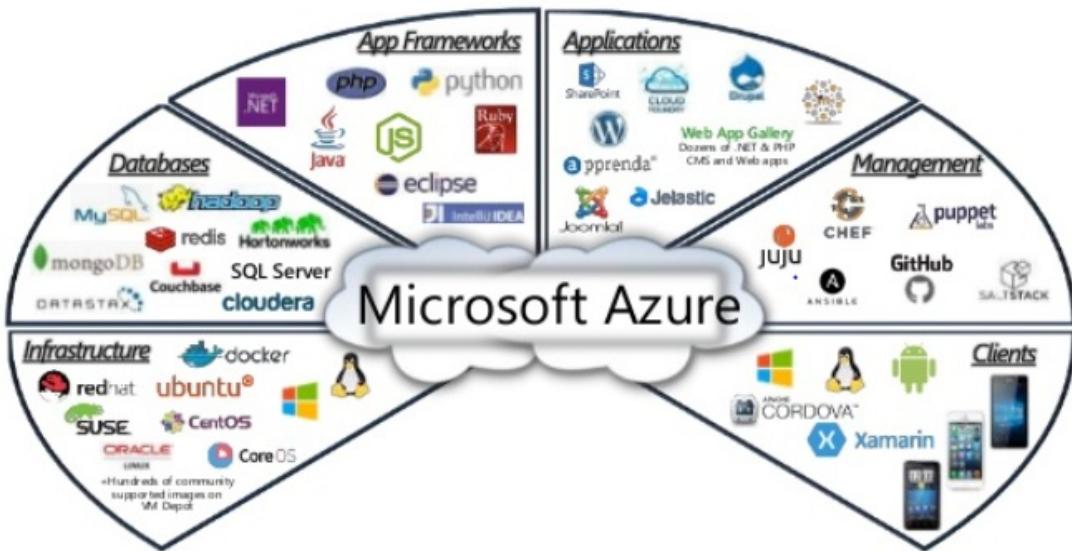
이 외에도 다양한 모듈과 결합하여 사용하기 수월한 점이나 개발자가 소프트웨어의 전체적인 논리에만 집중할 수 있고, 배후의 세부적인 부분은 Tensor Flow가 알아서 처리해준다는 장점이 있다.

아파치 2.0 라이센스의 오픈소스이며, 구글을 비롯한 다양한 기업에서 활용되고 있다.

**Microsoft Azure Open Sources** - [Github](#) | [Home](#)

# Microsoft Azure is an Open Cloud

We've delivered an open, broad, and flexible cloud across the stack



Microsoft는 2010년도 이후로부터 친 오픈소스 행보를 보였는데, 2015년 CEO가 '마이크로소프트는 리눅스를 사랑한다' 라고 발표한 뒤로 여러 오픈소스 재단과 파트너쉽을 맺거나 각종 기술들을 공개하는 등 오픈소스 소프트웨어 생태계를 적극적으로 지원해 왔다.

실제로도 Github에서 활발히 (직업을 등록하고) 활동하고 있는 회사 중 하나가 Microsoft의 직원들이다. 올해 여름에는 Github를 인수하기도 했다.

Microsoft의 클라우드 플랫폼인 Azure 활성화를 위해 수많은 기능을 공개했는데, 그 중에는 머신러닝 툴킷 DMTK를 오픈소스로 공개했다.

[DMTK Github](#) | [DMTK Home](#)

**OpenAI - Github | Home**



OpenAI는 비영리 인공지능 연구 회사로서, Tesla의 머스크와 다른 투자자들이 투자해 만든 비영리 단체이다.

2016년 OpenAI는 강화학습 연구를 위한 플랫폼인 OpenAI Gym의 공개 베타를 출시했고, 소프트웨어 플랫폼인 Universe를 출시하는 등 활발한 행보를 보이고 있다.

DeepMind에 대항해 OpenAI가 새로 개발한 기술은 진화 관점에서 접근했다. OpenAI가 50개 이상의 게임을 통해 소프트웨어를 테스트한 결과, DeepMind가 강화학습 훈련을 적용하는 데 하루가 걸렸던 것을 단 한시간만에 달성할 수 있었다.

이러한 진화 전략은 로봇공학의 표준테스트에서도 성과를 보였다. 10시간을 투입했어야 했을 과제를 단 10분만에 성공했다고 한다.

참고로 OpenAI의 Gym도 파이썬 패키지로 제공하며, 완전 공개되어 있다.

[Github](#) | [News](#)

### Open Neural Network eXchange - [Github](#) | [Home](#)



Open Neural Network eXchange (ONNX)는 Microsoft, Amazon, Facebook등의 기업들이 오픈소스 프로젝트로 공동 개발한 서로 다른 오픈 소스 AI 프레임워크 간의 상호 운용성을 허용하는 심층 학습 모델을 위한 형식을 제시한다.

### Deepmind Open Sources - [Github](#) | [Home](#)



영국의 인공지능 연구 회사이다. 2014년 구글에 인수되었고, 유명한 프로그램인 알파고를 개발했다. 한국에서 머신러닝은 몰라도 알파고를 모르는 사람은 없다.

알파고는 딥마인드가 개발한 알고리즘인 Deep Q-Network(DQN)라는 알고리즘을 사용하는데, 디층 신경망 기술과 Q-Learning을 조합한 딥러닝 알고리즘이다.

딥마인드는 구글에 인수된 이후로 다양한 라이브러리와 기술들을 공개했는데, 그 중에는 구글의 TensorFlow를 활용한 라이브러리인 Sonnet가 있다.

그 외에도 인공지능 기술 테스트 플랫폼인 DeepMind Lab등 다양한 방면의 소프트웨어를 공개하고 있다.

### **DeepDream - [Github](#) | [Home](#)**

#### <초창기>



#### <후반>



DeepDream은 구글 엔지니어가 개발한 컴퓨터 비전 프로그램이다. 최근 자동 포토샵 프로그램으로 유명해졌다. 네이버에 검색하면 신한카드만 나온다.

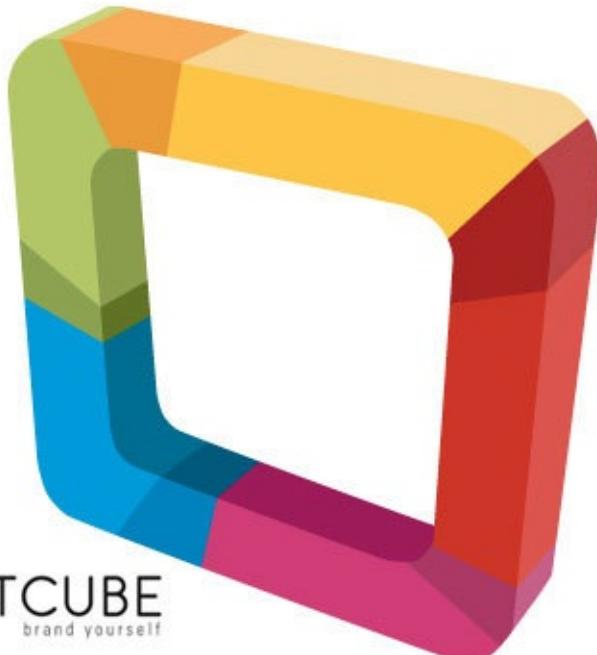
DeepDream은 수많은 사진들을 학습시켜 이미지 구분이 가능하도록 만들어졌고, 신경망 네트워크를 통해 이미지 패턴을 항상시키고 축적된 알고리즘을 토대로 이미지를 만들어 낸다.

TensorFlow를 활용해 매우 유사한 프로그램을 만들어 볼 수 있다.

[Blog](#)

## 3.6 Contents Management System (CMS)

**Textcube** - [Github](#) | [Home](#)



2004년 처음 개발이 시작되었고, 2005년 9월 전문 벤처 기업인 태터앤킴퍼니에 의해 1.0 버전이 개발되기 시작했다.

2006년 4월 태터툴즈 1.0.2가 GPL 으로 공개된 이후 수많은 사용자와 개발자 커뮤니티인 태터앤프렌즈 (TNF), 태터앤킴퍼니 (TNC)가 개발하기 시작했다.

개발자 커뮤니티의 개발 공헌은 날이 갈수록 높아져, 1.1 버전부터는 TNF가 전담할 정도였고, 2007년 적극적 참여 집단인 니들웍스 (Needworks)를 발표하게 된다.

2007년 TNF의 한 개발자에 의해 현재의 이름인 텍스트큐브를 갖게 되었다.

00년대 중반~10년대 초반 인기가 많았지만, 현재는 1.10 버전 이후로 큰 업데이트는 이루어 지지 않고 있다.

추가로, 단축키 기능이 있는데 단축키 중 일부가 Vi 에디터의 (커서)단축키와 동일하다.

**XpressEngine** - [Github](#) | [Home](#)



BBS 프로그램 Zeroboard를 개발한 개발자가 네이버의 지원을 받아 제작했던 웹 사이트 CMS이다. 지금은 네이버가 개발팀을 지원하고있고, 오픈소스 프로젝트로 운영하고 있다.

제로보드를 업그레이드 하기 위해 만들어졌지만, 제로보드와 호환이 되지는 않는다. 애초에 구조부터가 다르다.

여타 CMS와 마찬가지로 개발자가 따로 개발한 애드온, 모듈 등은 각각의 저작권을 가진다. 기본적인 Framework인 XE Core는 GNU LGPL 를 따른다.

XE 1.4/1.5/1.7이 공존하던 2012~2014년에는 호환성 문제가 심했지만, 그만큼 신규 이용자의 유입이 많았다.

올해 10월 17일에 XE 1.11.1 버전이 업데이트 되었다. 전과 비해 개발속도가 빠르지는 않지만 드물게 업데이트가 이루어지고 있다. 다른 버전으로는 1.8 버전을 fork한 커뮤니티에 특화된 Rhymix와 1.x 버전대와 전혀 다른, 네이버가 개발하고 있는 XE3 도 존재한다.

### [Rhymix - Github | Home | Community](#)



XE가 XE 1.8과 XE3을 동시에 개발하면서 업데이트가 느려지자, 개발자 및 사용자들이 XE를 fork해 개발한 CMS이다. XE와 호환성이 좋다.

XE와는 다르게 GPL 라이센스를 따른다. XE에서 Rhymix로 옮기는 것은 수월하지만, Rhymix에서 XE로 옮기는 것은 조금 어렵다. 초창기와는 다르게 세세한 부분에서 바뀐 부분이 많다.

### [GNUBOARD - Github | Home](#)



2000년대 중반에 만들어진 CMS 프로그램. 당시 경쟁 CMS였던 제로보드는 독자 라이센스를 사용하고, 저작권 규정이 있던 것과 달리 GNU 개념의 프로그램이고, GPL 라이센스를 사용하는 CMS이기에 제로보드의 대체 CMS로 인기가 있었다.

영카트라는 온라인 쇼핑몰 플러그인을 제공하는데, 대세에 맞추어 최근 오픈소스가 되었다.

## WordPress - [Github](#) | [Home](#)



2003년 만들어진 세계 최대 오픈소스 저작물 관리 시스템이다. 전 세계 웹사이트의 30%는 wordpress로 만들어 졌을만큼 엄청난 인기를 갖고 있다.

버전업이 될 때마다 수많은 오픈소스 플러그인들이 나오자 CMS 기능을 수행해 하나의 사이트를 만들 수 있게 되었다.

현재에는 GNU GPL v2 라이센스를 따른다.

## 3. 7 Operation System

### Android



리눅스 커널을 기반으로 만들어진 모바일 플랫폼을 대상으로 한 운영체제.

세계 최다 사용자를 보유한 운영체제이자, 세계에서 가장 유명한 오픈소스이기도 하다.

구글에서 커널부터 SDK(Software Development Kit)까지 모두 오픈소스로 풀려있고, 개발에 필요한 도구까지 모두 오픈소스로 풀려있다.

리눅스 커널과 JVM을 사용해 성능이 조금 떨어진다는 평은 있지만, 호환성이 뛰어나다는 장점이 있다.

라이센스는 아파치 라이센스 2.0을 따르고, 리눅스 커널을 수정한 부분에서는 GPL 2.0을 따른다. GPL의 강제성이 돋보인다.

최근에는 구글이 안드로이드 개발에 Kotlin을 이용하고 있다. C, C++, Java, Kotlin 총 4개의 언어를 사용하는 OS인 셈.

### Linux

이 부분은 앞에서 할 얘기랑 겹칠거 같아 일단 비워둡니다.

### Tizen



삼성이 독자적으로 만들었다고 생각하지만, 삼성과 인텔의 주도와 화웨이, 한국 일본의 통신사, 그리고 리눅스 재단등 수많은 기업의 연합으로 만든 OS이다.

리눅스 커널과 HTML5, C++ 을 기반으로 만들어졌다. 2012년 첫 출시를 시작으로, 우여곡절 끝에 2017년에는 타이젠 3.0이 출시되었다.

타이젠은 GPLv2, LGPL, 아파치 라이센스, BSD 라이센스를 따른다.



2014년 초 미래창조과학부의 오픈소스 활성화 계획의 일환으로 만들어진 리눅스 민트 기반의 OS.

기존 리눅스 민트에 비해 한글화가 많이 되어 있다. 홈페이지에는 한글화를 위해 리눅스 민트의 원본 문장을 옮겨 누구나 번역할 수 있게 해놨다.

윈도우 의존도를 낮추기 위해 만들어진 만큼, 다양한 서드파티 소프트웨어가 내장되어 있다.

현재는 각종 정부기관과 학교에서 사용하고 있다. 그렇지만 최근에는 업데이트가 되고 있지 않는 점은 최대 단점이다.

클라우드에 특화된 구름OS 베타 버전이 2015년에 발표되었다. 과학기술정보통신부는 보안을 욕하는 기관에서는 구름OS를, 일반 공공기관에서는 하모니카OS를 사용하게 할 예정이라고 한다.

하모니카 OS는 CCL 3.0 라이센스를 따른다.

## CyanogenMod



안드로이드 비공식 펌웨어 중 가장 유명한 펌웨어. C와 C++, 그리고 Java를 이용해 개발되었고 CM이라는 약칭이 존재한다.

CM 이전에는 하드웨어 제조사가 내놓은 순정롬을 개조해 만들었지만, CM은 안드로이드 오픈 소스를 직접 이용해 스마트폰에 맞게 만들기 때문에 높은 자유도를 자랑한다.

CM을 개발하고 있던 개발자를 삼성전자에서 스카우트하기도 했다.

## RHEL



오픈소스 기술이 처음 등장했을 때만 해도 오픈소스가 돈을 벌어다 줄 거라고 생각하는 사람은 거의 없었다.

소스코드가 공개되고 누구나 내려받을 수 있다면, 굳이 그 기술에 돈을 낼 사람은 없을 것이기 때문이었다. 이러한 생각을 뒤집은 기업이 바로 레드햇이다.

레드햇은 아무리 공개된 기술이라고 해도 기업에선 더 편하고 안전하게 오픈소스 기술을 가져다 쓰고 싶을 거라고 판단했다. 레드햇의 예상은 적중했다. 회사 설립 이후 레드햇의 연 매출액은 꾸준히 상승곡선을 그리고 있다.

이와 더불어 레드햇은 오픈소스 기업의 대명사로 자리잡았다..

## FreeBSD



FreeBSD는 한마디로 말하면 4.4BSD-Lite2에 기반한 공개 BSD 유닉스 운영체제이다.

처음에는 x86 아키텍처에 최적화하는 것을 목적으로 시작하였지만, 이에 대한 성과를 바탕으로 현재 알파 시스템에 대해서도 동시 릴리즈가 되고 있고, IA64나 PowerPC 등과 같은 아키텍처에도 포팅이 진행중이다.

FreeBSD의 큰 특징이라 한다면 다음과 같은 점을 들 수 있다.

유닉스 운영체제의 모든 특징 포함 (다중 프로세스, 다중 사용자)

안정적이고 고성능의 TCP/IP 스택

기본적인 IPv6 지원(버전 4부터, 버전 2와 3은 KAME 키트 사용)

다른 운영체제(리눅스, SCO, 솔라리스 등)의 바이너리 실행 가능

통합된 가상메모리/버퍼 설계와 관리

멀티프로세서 지원

기본적인 개발 환경 지원(C/C++, Perl)8. 자유로운 라이센스

## 3.8 Design Material

[Google Design - Github](#) | [Home](#)



구글은 Material Design Lite 이하 MDL이라는 디자인 가이드라인을 2014년 공개했다.

개발자는 MDL을 통해 코드의 가이드라인을 얻을 수 있을 뿐만 아니라, 버튼, 체크박스, 텍스트 박스와 같은 front-end 디자인을 수월하게 할 수 있다.

소스코드는 라이브러리, 프레임워크, 개발환경 등 외부 요소에 의해 변하지 않도록 구성되어 있고, 디자인 아이콘들은 github에 아파치 라이선스 2.0으로 올라와 있다.

[XE icons - Github](#) | [Home](#)

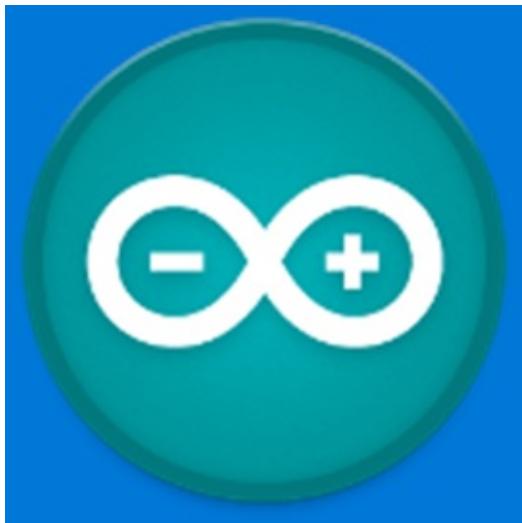


XpressEngine 또한 벡터 이미지의 아이콘의 소스코드를 오픈소스로 운영하고 있다.

웹 사이트 제작에 최적화되어 있고, OFL 라이센스를 따르는 오픈소스 이다. Github에는 css를 포함한 아이콘의 소스가 올라와 있다.

## 3.9 Hardware Tool Kit

**Arduino** - [Github](#) | [Home](#)



오픈소스를 기반으로 한 단일 보드. 2005년 이탈리아에서 만들어진 뒤로, 지금까지 꾸준히 새로운 모델을 출시하고 있다.

아두이노의 장점은 마이크로컨트롤러를 쉽게 동작시킬 수 있다. 아두이노는 컴파일된 펌웨어를 USB를 통해 쉽게 업로드 할 수 있을 뿐만 아니라, 여러 OS를 모두 지원한다.

아두이노의 보드 회로도가 CCL 라이센스로 공개되어 있어 누구나 보드를 수정하고 만들 수 있다는 장점이 있다.

## 참고문헌

[오픈소스란 무엇인가?](#)

[오픈 소스](#)

[리눅스](#)

[우분투](#)

[BSD](#)

[MySQL](#)

[아파치 HTTP 서버](#)

[파이어폭스\(웹 브라우저\)](#)

[워드프레스](#)

[자유·오픈소스 소프트웨어 해킹 역사](#)

[Revolution OS](#)

[성당과 시장](#)

[유닉스, 리눅스, 오픈 소스/자유 소프트웨어의 역사](#)

[리눅스란?](#)

[오픈소스 소프트웨어 연구 보고서](#)