



**OPEN  
SOURCE**



EDIT BY - OPEN SOURCE -04 SCIENCE

# 개요

---

## 오픈소스의 소개

---

1-1. 오픈소스의 정의 (오픈소스란 무엇인가?)

1-2. OSI가 제시한 오픈소스의 조건

2-1. 오픈소스의 특징

3-1. 오픈소스 툴

3-2. 오픈소스 성공 프로젝트

4-1. 오픈소스 소프트웨어 라이센스

4-2. 오픈소스 라이센스 관련 법적 쟁점

5. 오픈소스의 현황과 전망

## 오픈소스의 역사

---

6. 오픈소스의 역사

7. 오픈소스 주요 인물들

## 출처

---

8. 출처

# 오픈소스의 소개

---

## 1-1. 오픈소스의 정의 (오픈소스 란 무엇인가?)

---



- **오픈 소스 소프트웨어(Open SourceSoftware)** 오픈 소스 소프트웨어는 일반적으로 자유롭게 사용, 복제, 배포, 수정할 수 있으며, 소스코드가 공개되어있는 소프트웨어를 말한다.
- **오픈 소스(open source, 문화어: 공개원천)**는 소프트웨어 혹은 하드웨어의 제작자의 권리(특허, 저작권 등)를 지키면서 원시 코드를 누구나 열람할 수 있도록 한 소프트웨어 혹은 오픈 소스 라이선스에 준하는 모든 통장을 일컫는다. 어원에 대해서는 History of the OSI 자료에 따르면, 1998년 2월 3일에 넷스케이프 브라우저의 원시 코드에 어떠한 형태로 공개할까 하는 전략회의에서 붙여진 새로운 용어라고 설명되어 있다.

## 1-2. OSI가 제시한 오픈소스 조건

---



- 자유로운 재배포

라이선스는 여러 다른 소스들의 프로그램을 포함하고 있는 집합적 소프트웨어 배포판의 컴포넌트로서 소프트웨어의 판매나 기부에 대해서 그 어떤 누구도 배제되어서는 안 된다. 라이선스는 로열티 또는 판매 비용이 없다

- 소스 코드

프로그램에는 소스 코드가 포함되어 있어야 하며 소스 코드 형태와 컴파일이 완료된 형태로 배포될 수 있어야 한다. 특정 형태의 제품이 소스 코드로 배포되지 않으면 소스 코드를 구매해야 하며 인터넷을 통해 무료로 다운로드 해야 한다. 소스 코드는 프로그래머가 프로그램을 변경하기에 좋은 형식이다. 알아보기 힘든 소스 코드는 사용될 수 없다. 프리프로세서(preprocessor)의 아웃풋 또는 트랜슬레이터 같은 중간 형태도 사용될 수 없다.

- 파생 작업(Derived work)

라이선스는 변경과 파생 작업이 가능해야 한다. 원래의 소프트웨어 라이선스와 같은 조건 하에 배포되어야 한다

- 소스 코드의 무결성

구현 시 프로그램을 변경할 목적으로 소스 코드와 함께 "패치 파일"의 배포를 허용하는 경우에만, 라이선스는 소스 코드가 변경된 형태로 배포되는 것을 제한한다. 라이선스는 변경된 소스 코드에서 구현된 소프트웨어의 배포를 허용해야 한다. 라이선스는 파생 작업(derived work)을 통해 다른 이름 또는 다른 버전을 만들어야 한다.

- 개인 또는 그룹의 평등

라이선스는 어떤 개인이나 그룹에 차별을 두어서는 안 된다. OSI(Open Source Initiative)에서 말하는 오픈소스 조건 Source: <https://opensource.org/osd>

- 분야에 대한 평등

라이선스는 특정 분야에서 프로그램을 사용하는 것에 대해 제한을 두어서는 안된다. 예를 들어, 프로그램은 비즈니스 또는 유전공학 연구 분야에서 사용될 수 있다.

- 라이선스 배포

프로그램에 대한 권한은 프로그램이 재배포 된 모든 곳에 적용되어야 한다. 추가 라이선스를 발행할 필요가 없다.

- 제품 스펙에 따른 라이선스

프로그램에 대한 권한은 특정 소프트웨어 배포판의 프로그램의 일부에 해당하지 않는다. 프로그램이 그 배포판에서 출판되었고, 그 프로그램의 라이선스의 조건 하에 사용 및 배포된다면 재배포 된 프로그램을 사용하는 모든 당사자들은 원래의 소프트웨어 배포판에서 허용된 것과 같은 권한을 갖게 된다.

- 라이선스는 다른 소프트웨어를 제한하지 않는다

라이선스는 라이선스를 받은 소프트웨어와 함께 배포된 다른 소프트웨어에 제약 사항을 두어서는 안된다. 예를 들어, 같은 미디어에 배포된 모든 다른 프로그램들이 오픈 소스 소프트웨어가 될 필요는 없다.

- 라이선스는 기술 중립적이어야 한다

라이선스는 기술이나 인터페이스 스타일을 한정해서는 안된다.

## 2-1. 오픈소스의 특징

- Fellerand Fitzgerald는 오픈소스 소프트웨어를 연구하는 방법론을 개발하면서 오픈소스 소프트웨어 현상을 분석적으로 접근하기 위한 틀을 제시하고 있다. 아래에서 언급된 내용들은 오픈소스 소프트웨어의 특성들을 표로 나타내었다.

### 1. 무엇이 오픈소스 소프트웨어인가? (what?)

소프트웨어 제품과 프로젝트를 오픈소스 소프트웨어로 정의하는 기준은 무엇인가?	오픈소스 소프트웨어는 그것이 배포되는 조건과 라이센스(OSD)에 의해 정의된다. 그러나 최근에는 그 개념에 상당한 유동성이 존재하고 있다.
어떤 유형의 제품과 프로젝트가 오픈소스 소프트웨어인가?	과거에는 운영체제와 네트워킹체제 프트웨어, 유틸리티, 개발도구, 인프라스트럭쳐 포넌트가 오픈소스 소프트웨어의 주종을 이루었다. 최근에는 생산성 향상과 관련된 응용소프트웨어, 엔터테인먼트 응용소프트웨어들이 많이 개발되고 있다.

### 2. 왜 오픈소스 소프트웨어를 개발하고 사용하는가? (Why?)

오픈소스 소프트웨어 개발의 기술적인 동기는 무엇인가?	훌륭한 코드, 신속한 개발주기, 높은 수준의 질, 신뢰성, 안정성의 획득, 좀 더 개방된 표준과 플랫폼의 확보가 오픈소스 소프트웨어 개발의 기술적인 동기이다.
오픈소스 소프트웨어 개발의 경제적인 동기는 무엇인가?	오픈소스 소프트웨어 개발에 대한 기업차원에서의 동기는 비용과 리스크의 공유이다. 또한 소프트웨어 산업을 일상적 제품(commodity) 산업을 넘어 서비스 산업으로 재편하려는 노력도 경제적 동기가 되고 있다.

오픈소스 소프트웨어 개발의 사회·정치적 동기는 무엇인가?

사용자 스스로가 느끼는 불편한 점을 해결하기 위한 노력, 사회적 평판의 획득, 고도의 능력을 지닌 개발자에게 배우는 것(mentorship), 의미있는 일을 하고 싶다는 의지, 공동체 지향의 이상 주의 등이 사회정치적 동기라고 할 수 있다.

### 3. 오픈소스 소프트웨어가 개발되는 시간적 공간적 환경은 어떠한가? (when? and where?)

오픈소스 소프트웨어 개발의 시간적인 측면은 어떠한가?

오픈소스 소프트웨어는 신속한 개발, 재빠른 제품의 진화, 빈번하고 지속적인 개선된 제품의 제출을 특징으로 가지고 있다.

오픈소스 소프트웨어 개발의 공간적 측면은 어떠한가?

지리적인 측면보다는 사이버스페이스에 의해 경계가 형성되는 팀으로 구성되어 있다. 개발팀은 분산된 형태로 소프트웨어를 개발한다

### 4. 오픈소스 소프트웨어는 어떻게 개발되는가 (how?)

오픈소스 소프트웨어 개발과정은 어떻게 조직되며 조직되는가?

오픈소스 소프트웨어 개발방법론의 핵심은 제품 개발과 디버깅에서 다수의 개발자가 참여하는 병행개발방식(massive parallel development)이다. 개발은 헐겁게 집중화되고(loosely-centralized) 협력적 방식을 통해 이루어지며, 무상의 노력을 제공하는 개발자들이 참여한다. 최근에는 좀 더 개발방식이 조직적으로 이루어지는 양상을 보이고 있으며 소프트웨어 개발에 대한 금전적 보상도 이루어지고 있다.

오픈소스 소프트웨어 모델을 지원하기 위해 어떤 것들이 사용되는가?

다수가 참여하는 병행개발방식은 인터넷을 통해 이루어진다. 인터넷은 의사소통과 협업, 배포플랫폼으로 활용된다. 학술기관, 비영리기관, 민간 기관들, 온라인 Agency service도 오픈소스 소프트웨어 개발을 지원하기도 한다.

## 5. 오픈소스 소프트웨어의 고객과 주요 개발자, 소유자는 누구인가? (who?)

오픈소스 소프트웨어 개발에 공 헌하는 개 인 개발자 의 특성은 무엇인가?	오픈소스 소프트웨어의 개발자는 주로 건전한 의미의 해커, 아마추어 개발자가 아닌 전문개발자, 상당히 동기부여된 개발자들이다. 그러나 개발자들의 평판 지향적인 경향 때문에 매우 겸손하고 자기를 낮추는 문화가 형성되어 있다. 이것은 협력개발을 촉진시키는 데 중요한 의미를 지니고 있다
오픈소스 소프트웨어 제품을 배 포하는 기 업들의 특 성은 무엇 인가?	오픈소스 배포회사들은 성공적으로 기업공개를 했다. 이들 기업들은 오픈소스 소프트웨어 개발자들을 지원해주는 후원자(patron)의 역할을 한다. 이들 기업들은 수익모델을 라이센스 요금이 아니라 새로운면서 찾으면서 소프트웨어 산업의 패러다임을 변화시키고 있다. 이들 기업들에게는 고객서비스, 브랜드 관리, 동료들의 평판이 결정적으로 중요하다.
오픈소스 소프트웨어 제품 사용 자들의 특 성은 무엇 인가?	지금까지 오픈소스 소프트웨어의 사용자들은 전문사용자나 신기술 초기 사용자들이었다. 전통적으로 사용자 풀과 개발자 풀은 상당히 겹쳐 있었다. 그러나 오픈소스 소프트웨어의 사용성이성과 인터페이스가 향상되면서 이러한 양상은 변화할 것이다.

## 3-1. 오픈소스 툴

### 1. 이슈 관리 툴

IT 프로젝트를 수행하다 보면 어느 상황에서든 버그나 이슈 문제는 발생한다. 또한, 수많은 예외사항과 다양한 요구사항도 지속적으로 발생한다. 이러한 내용을 문서로 남겨서 관리하는 것도 어느 정도 한계가 있으며 분실이나 내용 누락 같은 상황에 부딪히게 된다. 이러한 것들을 정리하고, 담당자를 할당하고, 처리하는 프로세스를 총체적으로 관리하는 툴이 이슈 관리 툴이다. 요즘 IT 프로젝트에서 주로 사용되는 이슈 관리 툴로는 Trac, Mantis, Jira, CodeBeamer 등이 있다.

### 2. 형상 관리 툴

소스코드 버전관리를 통한 형상관리 툴은 무수히 많다. 초기 버전관리 시스템인 CVS(Concurrent Version System)에서 나타난 여러가지 문제점을 개선한 SVN(SubVersion)이 등장한 이후 버전관리 시스템의 대명사 자리는 SVN이 차지하고 있다. 형상 관리 툴이란 형상항목(Configuration Items)의 식별과 제어 및 변경과 버전 관련 상태 등을 관리하는 툴을 말한다. 여기서 형상 항목이란 프로젝트 계획, 산출물, 데이터베이스 개체, 소스 코드 등의 비트 스트림과 프로젝트의 기타 모든 항목을 말한다. 요즘 IT 프로젝트에서 형상관리툴은 매우 널리 사용되고 있으며, 프로그램 개발을 하기 위해 개발 툴과 함께 기본적으로 설치하는게 형상 관리툴일 정도로 일반화 되어있기 때문에 이 논문에서 형상 관리 툴에 대한 자세한 설명은 논외로 하겠다.

### 3. 코드 인스펙션 툴(PMD, Checkstyle, Cobertura)

코드 인스펙션 툴로는 PMD, Cobertura, Checkstyle, JavaNCSS, Junit, JDepend 등 여려가지가 있지만 본 논문에서는 대표적이며 가장 기본적인 코드 인스펙션 툴인 PMD, Checkstyle과 Cobertura를 위주로 설명을 한다. 우선 PMD를 살펴보면 자바 소스코드를 자동으로 리뷰하여 잠재적 결함을 찾아주는 도구로써, 이를 립스나 Jbuilder와 같은 IDE 툴과 연동이 가능한 오픈소스 툴이다. 소스코드의 잠재적 버그나 사용하지 않는 코드, 복잡하고 중복된 코드 등을 찾아주며, 규칙 툴 설정이 XPath 표기 방식이라 추가 및 수정이 용이한 특징이 있다.

### 4. 지속적인 통합 툴(Hudson)

Continuous Integration(지속적인 통합, CI)이란, 개발자가 각각 개발한 소스코드를 모아서 필요한 특정 시점에 한꺼번에 빌드하는 방식이 아니라 매일 혹은 매주 일정한 시간에 SourceRepository에 있는 소스들을 자동으로 가져와 빌드함으로써 통합에서 발생하는 오류와 시간을 줄이기 위한 기법이다. 지속적인 통합 툴로는 CruiseControl, Continuum, Luntbuild, Hudson 등 여러 종류가 있지만, 웹 기반

으로 설치 및 관리가 용이하며, 유닛 테스트 관련 기능의 풍부함과 내장된 분산 빌드 기능을 제공하는 Hudson을 대표적으로 설명을 하겠다.Hudson은 지속적인 통합 툴의 일종으로 이러한 지속적인 통합 툴의 특징은 다음과 같다.

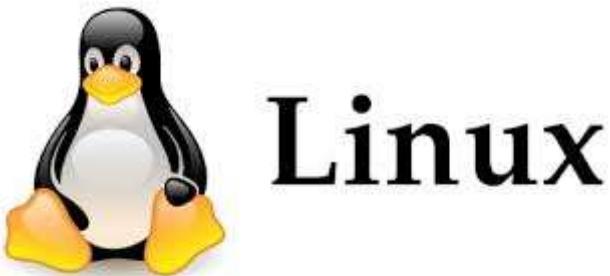
### 지속적인 통합 툴의 특징

- 1. 소스코드 일관성 유지  
CI툴을 적용하기 위해선 Subversion,CVS 등과 같은 SourceRepository 즉, 소스 관리 시스템이 필요하며 개발자들은 소스 코드를 본인의 PC에서 관리하는게 아니라 소스 관리 시스템에서 통합 관리하게 된다.
- 2. 자동 빌드  
소스 관리 시스템에 있는 소스코드를 CI 툴에 설정된 특정 시간마다 자동으로 컴파일, 빌드 과정을 거치게 된다.
- 3. 코드 인스펙션  
소스 코드 빌드 과정 중간에 PMD, Checkstyle, Cobertura등의 오픈 소스 코드 점검툴을 이용해서 코드 인스펙션을 수행함으로써 코드의 품질에 대한 무결성을 유지한다.

## 3-2. 오픈소스 성공 프로젝트

---

### 1. Linux



- GNU에서 관리하고 있는 오픈소스 운영체제
- X86 기반에서 벗어나 임베디드 기기, 모바일 등 다양한 하드웨어에 포팅
- 데스크탑이나 웹서버 위주에서 현재는 클라우드, 빅데이터 등의 분야에서 사실상의 표준 운영체제로 자리 잡음
- 모든 안드로이드 스마트폰 단말(한국 90.1%) □ 전세계 슈퍼 컴퓨터의 93.8% (2013년 11월 기준)
- Google, Twitter, Facebook, Amazon 등
- 개발자 8,000명, 회사 800개, 1,500만줄의 코드
- 3~6개월 주기의 커널 업그레이드 성공한 오픈소스 프로젝트 Source: : Worldwide Linux Server Operation System Environment by Vendors, 2006-2010, March IDC 2011

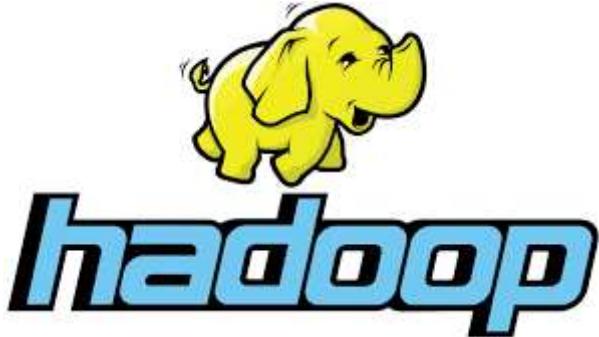
### 2. MySQL(MariaDB)



- 전세계적으로 가장 많이 사용되고 있는 오픈소스 데이터베이스의 하나
- 페이스북 등 거의 대부분의 웹 서비스 업체에서 운영 데이터 저장용으로 사용
- 현재는 클러스터링, 분산(샤딩), 복제 및 백업 등 엔터프라이즈 환경에서 필요로 하는 고급 기능까지 지원
- 오라클이 인수(10억 달러) 후 “오픈코어 전략” 표방
- 오픈소스 결과물의 확장 기능이나 엔터프라이즈 기능 공개하지 않고 있음
- 이에 반발한 마이클 몬티가 막내 딸 이름을 붙여 MariaDB 시작 Apache HTTP Server

- Apache 재단의 대표 프로젝트로 전 세계적으로 가장 많이 사용(46% 점유율)하는 웹서버
- NCSA의 HTTPd 코드를 기반으로 Linux에서 사용할 수 있도록 웹 서버 코드를 재작성
- 커뮤니티 그룹과 미 델라웨어사와 협작하여 Apache Software Foundation 설립 (1999) 성공한 오픈소스 프로젝트

### 3. Hadoop



- 빅데이터 분야의 대표적인 오픈소스 프로젝트
- 거의 모든 빅데이터 업체들이 사용하고 있는 기반 플랫폼 - (예) Hortonworks, Cloudera, MapR, IBM, Oracle
- 빅데이터 관련 소프트웨어들의 생태계 역할 - (예) Pig, Hive, HBase
- 모바일, 사물인터넷에 기반한 급속한 데이터 증가에 따라 더욱 중요도 증가  
Openstack
- 클라우드 업체인 Rackspace와 NASA가 협력하여 만든 오픈소스 클라우드 운영 시스템
- 클라우드 시스템 구축에 필요한 컴퓨팅, 네트워크, 스토리지 자원에 대한 가상화 및 운영 관리는 물론이고 보안, 워플로우 등 거의 모든 요구사항 지원하는 사실상의 표준
- 클라우드 업체는 물론 가상화, 하드웨어, 네트워크, 스토리지 등 다양한 벤더들이 참여 중 MongoDB
- 기존 RDB와는 다른 NoSQL 기반의 데이터베이스
- 빅데이터 시대의 도래와 함께 비정형 데이터를 대량으로 처리할 수 있는 장점 부각
- RDB와 유사한 인터페이스, 폭넓은 사용자 저변, 높은 성능으로 NoSQL 분야에서 사실상의 업계 표준 성공한 오픈소스 프로젝트

#### 4. Pentaho



- BI(Business Intelligence) 및 데이터 통합(ETL 등) 분야의 오픈소스 강자
- Hadoop과 NoSQL 지원을 통해 빅데이터 시장 진입 □ 자본 유치와 고객 확보에도 성공 PostgreSQL(Enterprise DB)
- MySQL(MariaDB)과 함께 주요한 오픈소스 RDB 프로젝트의 하나
- Enterprise DB를 통해 상용화된 버전 제공
- Oracle DB와 뛰어난 호환성 Wordpress
- 오픈소스 블로그 플랫폼이자 WCM(Web Contents Management) 플랫폼
- 전 세계 웹사이트의 22%, WCM 시장의 60% 이상 점유 WSO2
- Apache 라이선스 기반으로 한 오픈소스 미들웨어 스위트
- ESB, BPM, BRM, API Manager, CEP 등의 컴포넌트 성공한 오픈소스 프로젝트

## 4. 오픈소스 라이선스 위반과 저작권 침해

### 1. 오픈소스 소프트웨어 라이선스

#### 1. 개요

계약법적 관점에서 볼 때 소프트웨어가 주로 저작권법에 의하여 보호된다는 사실은 매우 중요하다. 왜냐하면 그 결과로서 소프트웨어의 이용은 일반적으로 권리보유자로부터 사용허락(licence)을 필요로 한다. 라이선스의 이용은 소프트웨어 제공의 표준적 방법이 되었고 이는 공급자와 사용자에게는 계약법적 그리고 저작권법적 문제를 야기한다. 오늘날에는 저작권법에 의한 소프트웨어의 보호가 가능 해졌음에도 불구하고, 소프트웨어 소유자(제작자)는 계약에 의한 소프트웨어 보호를 추구하는 경우가 일반적인 상황인데, 이 경우 저작자와 이용자 간에 저작물이용, 종류 및 방법을 정하고 있는 계약을 ‘라이선스’라 볼 수 있을 것이다. 개발한 소프트웨어를 일반 공중의 자유로운 복제, 개작 등을 허용하는 OSS로 할 것인지 여부는 전적으로 소프트웨어 저작권자(이하 “SW저작권자”라 함)의 자유로운 의사에 달린 문제이다. 그런데, SW저작권자가 특정 소프트웨어를 OSS로 공표할 때는 일반 공중에게 소프트웨어 복제, 배포 등을 이용 허락한다는 취지의 의사표시를 첨부하여 당해 프로그램을 공표하는 방법을 취하게 된다. 이때 일반 공중에게 프로그램의 복제 등을 이용 허락하는 의사표시를 하는 방법으로는 위와 같은 내용이 담긴 당해 소프트웨어의 라이선스를 첨부하는 것이 일반적인데, 오픈소스 라이선스로서 가장 대표적인 것으로 GPL(General Public License)을 들 수 있다.

#### 2. 오픈소스 라이선스의 분류



\* <http://freshmeal.net/stats/#license> 오픈소스SW 라이선스 분포표

2009년 4월 기준으로, 65개의 라이선스가 OSI(Open Source Initiative)14)에 오픈소스 라이선스로 인정되어 등록되어 있다. 하지만 실제로 많이 사용되는 라이선스의 갯수는 한정되어 있다. 그런데 오픈소스 라이선스는 OSS의 본질인 소스 코드 공개 조건 및 전파성의 강함에 따라

#### 1. GPL(General Public License) 유형

2. MPL(Mozilla Public License) 유형

3. BSD(Berkeley Software Distribution) 라이선스 유형으로 분류된다.

1. GPL 유형은 전파성이 가장 강한 라이선스이다. 개작 부분의 소스코드 공개는 필수이고, GPL 대상 코드와 다른 코드를 조합 시킨 프로그램의 경우에 다른 코드에 대해서도 라이선스 조건이 미치게 되어 소스코드 공개, 기타 의무가 발생 한다. 한편, OSS의 라이브러리를 다른 소프트웨어에 링크하여 사용하는 경우, GPL에서는 링크하는 소프트웨어도 GPL에 구속되어 공개되지 않으면 배포할 수 없는 반면, LGPL (GNU Lesser General Public License)에서는 공개 소프트웨어의 라이브러리를 유상 소프트웨어와 링크하여 사용하거나 공개 소프트웨어의 라이브러리를 사용하는 유상 소프트웨어와 함께 배포하는 것이 허락된다. LGPL은 보다 광범위한 사용. 재배포를 인정하고 있기 때문에, 사실상 표준(de facto standard)을 목표로 하는 경우에는 LGPL을 사용하는 것이 권장된다. MPL 유형은 개작 부분의 소스코드 공개는 필수 이지만, 전파성은 강하지 아니하여 다른 코드와 조합 시키더라도 다른 코드 까지 라이선스 조건이 미치지 아니하므로 소스코드 공개가 요구되지 아니한다.

2. MPL 유형의 라이선스에 따라 새로운 소프트웨어를 개발한 경우에 당해 소프트웨어는 다시 같은 라이선스에 따라 공개하여야 한다는 점에서 Copyleft 라이선스와 유사한 반면, 독점적 소프트웨어와 결합을 허용한다는 점에서 순수한 Copyleft 라이선스보다는 덜 제한적이다. BSD 라이선스 유형은 개작 부분을 포함하여 소스코드 공개는 필수가 아니며, OSS를 상용Software와 조합을 하는 것도 허용된다.

3. BSD 라이선스 유형은 근본적으로 오픈 소스 소프트웨어의 무제한적인 상업적 이용 뿐만 아니라 사적 재산으로 보호되는 2차적 저작물(proprietary derivative works)의 개발도 무제한적으로 허용하며 2차적 저작물의 원시프로그램 공개를 강제하지 아니한다. 한편 라이선스 간의 호환성을 확보 여부에 따라 오픈소스 라이선스를 구분할 수도 있다. 첫째, 완벽한 호환가능성을 제공하는 라이선스로는 GPL과 LGPL 등이 있다. GPL을 이용하는 OSS는 다른 군에 속한 라이선스를 이용하는 OSS를 자유롭게 이용할 수 있다. 다만, 통합 개발(Lager Work)에 의해 새로운 소프트웨어가 창작되었을 경우에는 반드시 GPL에 따라 공개하여야 한다. 둘째, MPL과 같이 부분적으로 독점가능성을 허용하는 라이선스 군에 속하는 경우에는 MPL을 포함하여 BSD, Artistic, MIT, Apache 라이선스를 이용하는 OSS와의 호환성을 제공한다. 셋째, BSD, Artistic, MIT, Apache 라이선스를 이용하는 OSS는 자신이 속한 군에 속하는 라이선스를 이용하는 OSS와의 호환성만을 제공한다. 이는 다른 군에 속하는 라이선스를 이용하는 OSS를 이용하거나 이와 결합한 경우 다른 군에 속하는 라이선스를 통

해 재배포해야 함을 의미한다.

주요 오픈소스SW 라이선스 비교						
	무료 이용가능	배포 허용가능	소스코드 취득가능	소스코드 수정가능	2차적 저작물 제공개 의무	독점SW와 결합가능
GPL	○	○	○	○	○	×
LGPL	○	○	○	○	○	○
MPL	○	○	○	○	○	○
BSD license	○	○	○	○	×	○
Apache license	○	○	○	○	×	○

<출처 : [http://www.olis.or.kr/oss/license/open\\_source\\_guide.pdf](http://www.olis.or.kr/oss/license/open_source_guide.pdf)>

## 2. 오픈소스 라이선스 관련 법적 쟁점

### 오픈소스 소프트웨어의 법적 위험 - 전파성의 위험

OSS의 개발은 일반적으로 기존의 프로그램에 서는 해결되지 않았던 특정의 문제를 해결하기 위하여 한 사람의 프로그래머가 스스로 프로그램을 개발하는 것에서부터 시작된다. 일단 작동하는 프로그램이 만들어지면 인터넷 등을 통해 소스코드를 포함해서 공개하고 시운전용 / 버그보고, 버그 수정 / 기능개량의 면에서 협력을 구하는 것에 의해서 이용자나 공동개발자가 나타나고, 그 프로그램에 관한 개발자와 이용자로 구성된 공동체(community)가 형성되며, 그 공동체를 베이스로 하여 버그수정 및 기능개량이 지속되어 간다. 이와 같이, OSS는 개발과정에 다수의 자가 관여함에 따라 개발자가 인식할 수 없는 제3자의 저작권을 침해하는 SW가 내장(embedded)되어 있을 가능성이 있다. 더욱이, 그러한 내장은 소스코드의 공개가 되는 경우에 사후적으로 쉽게 노출되기 때문에, 저작권 침해를 주장하는 자로부터 소송이 제기될 위험이 상용 소프트웨어 보다 높다.

# 오픈소스의 역사

---

## 1960년대 오픈소스의 시작

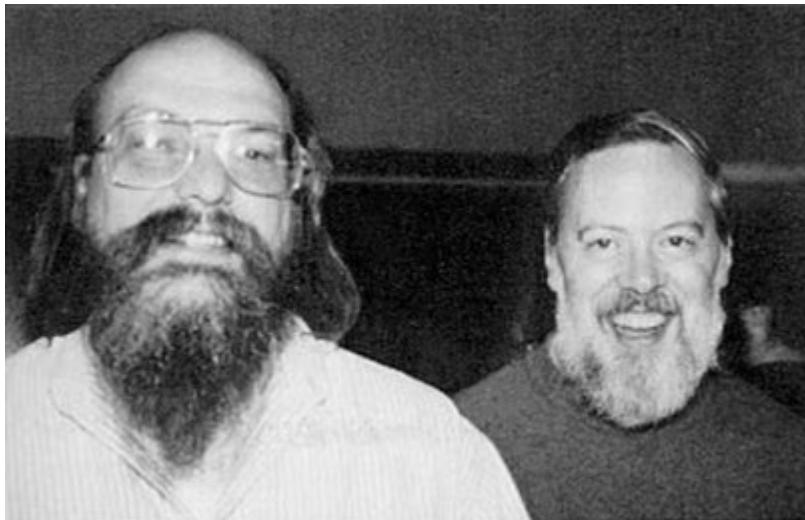
---

오픈소스라는 아이디어의 시작점은 컴퓨터의 정의를 어떻게 내리느냐에 따라 다를 수 있지만 사실 컴퓨터의 시작과 함께라고 할 수 있다. 현대적인 컴퓨터의 시작이라 할 수 있는 1960년 대 하드웨어가 판매의 중심이었으며 소프트웨어는 사람들 사이에서 자유롭게 돌아다니면서 소프트웨어 공유 문화가 퍼져 있었기 때문이다.

## UNIX (1965~)

---

1965년 MIT, AT&T 벨 연구소, General Electric에서는 Multics라는 실험적인 운영체제를 공동으로 개발하는 프로젝트를 진행하였다. 이 프로젝트는 멀티태스킹, 멀티유저를 지원하는 초기 형태의 시분할 운영체제를 만들고자 했던 것이다. 그러나 Multics는 초기의 설계 목표와는 다르게 비대해지고 쓸모없는 운영체제로 개발되어갔고, 프로젝트는 좌초되기에 이르렀다.



Ken Thompson(left) and Dennis Ritchie(right)

그러나 이 프로젝트에 참여했던 켄 톰슨과 몇몇 연구원들은 프로그램에 관한 연구를 효과적으로 수행하기에 적합한 환경을 만들어보자는 의도에서 계속하여 운영체제 개발에 몰두하였고, 그래서 탄생한 것이 초기 형태의 UNIX이다.

당시에는 하드웨어를 운영하는 프로그램을 그 하드웨어의 어셈블리어로 작성하여 사용했는데, 유닉스 역시 미니 컴퓨터인 PDP-7에서 돌아가도록 만들어진 것이었으며, 어셈블리어로 작성되어 있었다. 따라서 다른 기종에 이식하려면 그 기종에 맞는 어셈블리어로 다시 작성해야만 하는 불편을 겪어

야 했다. 이렇게 초기의 UNIX는 기계 의존적이며 기종 간에 호환성이 없는 그런 운영체제였다. 그리고 UNIX를 운영체제로 사용하던 PDP-7 또한 많은 소프트웨어를 제공하지 못하는 상황이었다.



Thompson(sitting) and Ritchie working together at PDP\_11

## UNIX (1973)

---

1973년 이러한 상황 속에서 데니스 리치가 C언어를 개발함으로써, 어셈블리어로 작성되어있던 UNIX는 C언어로 재작성되어 다시 태어나게 되었다. 이제 UNIX는 이식성과 호환성있는 시스템으로써 사용자들로부터 큰 반향을 일으켰고, 벨 연구소를 중심으로 UNIX 사용자 그룹이 형성되며 빠르게 버전업 되면서 퍼져 나가게 되었다. UNIX는 주로 연구와 학습을 목적으로 대학이나, 연구소 등에 무료로 배포되었고, 이를 이용하는 프로그래머들 또한 자연스럽게 서로에게 필요한 프로그램을 공유하는 공동체 분위기가 조성되어 있었다.

Source 프로그램이 공개되어 있었던 UNIX는 많은 대학들과 연구원들에 의해 연구되어 마침내 상업 시장에 진출하기에 이르렀고, Berkeley Unix(BSD), SYSV와 같은 계열로 분화되고, Sun OS, OSF/1.AIX, HP-UX, Solaris, IRIX, SCOUNIX 등과 같은 다양한 버전의 UNIX 운영체제들을 탄생시키는 모체가 되었다.

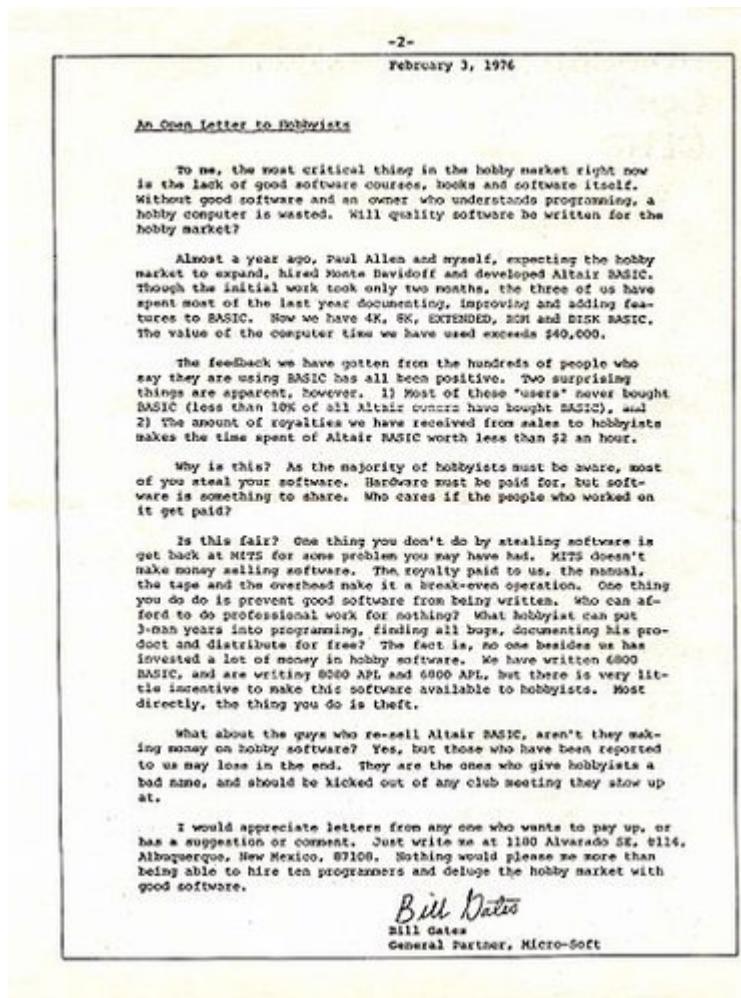
## “An open letter to hobbyists” by Bill Gates (1976)

---

1976년, 한 무리의 해커들과 컴퓨터를 취미로 하던 사람들이 실리콘 밸리에 모여 결성한 훈브루 컴퓨터 클럽에서 발행하던 뉴스레터에서는 그 때 당시 최근 마이크로소프트 창업한 빌 게이츠가 클럽 회원 전원에게 공개편지를 보낸 것을 발견할 수 있다. 그것은 “An open letter to hobbyists”로 독점 소프트웨어라는 당시에는 비교적 새로운 개념에 대한 논점들을 설명하였다. 그때까지 컴퓨터 사용자들은 소유권에 대해서는 별 생각 없이 소프트웨어를 마음대로 사용 가능하였었는데 1976년 친구 폴

앨런과 MITS Altair 8800을 위한 베이식 인터프리터 제조 후 MITS를 통해 판매하면서 소프트웨어 저작권에 대한 인지와 함께 상업용 소프트웨어 시대의 시작을 알리는 것이었다.

그렇게 1970년대 후반 ~ 1980년대 초반에는 서로의 소프트웨어 소스코드를 감추었으며 자신만의 활용을 위해서 변경이 불가능하게 되었다. 즉, 마이크로소프트는 독점적인 소프트웨어 모델을 만든 선구자들 중의 하나인 것이다.



"An open letter to hobbyists" by Bill Gates

Richard Stallman



# 1984~1986 리처드 스톤만과 GNU프로젝트, 그리고 자유소프트웨어 운동/재단

---

리처드 스톤만은 MIT LAB에서 일하며 겪었던 소프트웨어 독자적인 사용과 수정 및 공유 제한과 같은 경험들을 통해 1983년 9월 MIT 일을 그만두고 GNU 운영체제 개발을 시작했다. 그는 유닉스 운영체제와 비슷한 시스템을 개발하고 있었지만 유닉스 운영체제는 아니였기 때문에 “GNU is Not a Unix(GNU)” 프로젝트라는 이름을 붙인 것이다. 유닉스는 독점적이였기 때문에 많은 수의 서로 분리되어 통신을 주고 받는 프로그램들 하나하나 대체물을 만들어 교체하는 방식으로 운영체제를 만들어 나갔다. 또한 이 프로젝트 과정은 그가 프로그램 작성을 도와 달라는 발표를 보고 참여한 많은 사람들과 함께 했다.

GNU에서 가장 본질적인 것은 자유소프트웨어라는 것이며 여기서 자유란 누군가가 원하는 대로 바꿀 수 있는 자유 또는 다른 누군가를 고용해서 자신을 위해 변경하도록 하는 것, 그리고 복사본을 재배포하고 다른 이들과 공유하고 개선하고 다시 공표할 수 있는 것을 말한다. 이것은 사람들이 공동체를 구성하는 것을 가능하게 해주는 자유이며 이것이 자유소프트웨어가 아닌 것으로부터 자유소프트웨어를 구분하는 자유라고 리처드 스톤만은 말한다. GNU프로젝트의 목표는 “모두가 공유할 수 있는 소프트웨어”를 만드는 것이었다. 지적재산권이 기업의 독점적 지위를 높이고 시장을 독차지하며 기술혁신을 가로막을 것이라는 우려와 함께 그는 상용소프트웨어에 대항한 자유로운 대안을 만들기 위한 의지로 자유 소프트웨어 운동을 주도하였고, 1985년 10월, 자유 소프트웨어 재단(FSF)을 설립하였다.

자유 소프트웨어 운동(Free Software Movement)은 리처드 스톤만이 1980년대에 소프트웨어의 본래 생산 유통 방식인 정보 공유의 방식을 복원하고자 한 운동이다. 이 프로젝트의 핵심 작업은 운영체제를 만들어 여러 사람들의 손을 거쳐 더 완성도 높은 소프트웨어를 만드는 것이다. 그리고 이 운동의 목표는 운영 체제만이 아닌 모든 소프트웨어를 자유 소프트웨어로 만드는 것이다. 먼저 자유 소프트웨어의 운동은 3가지의 원칙을 기반으로 한다. 첫번째는 소프트웨어의 작동 원리를 연구하고 이를 자신에 맞게 변경시킬 수 있는 자유이다. 두번째, 소프트웨어를 이웃과 함께 공유하기 위해서 이를 복제하고 배포할 수 있는 자유, 마지막으로 소프트웨어를 향상시키고 이를 공동체 전체의 이익을 위해서 다신 환원시킬 수 있는 자유이다.



Logo of GNU

## 1989 GPL (GNU 퍼블릭 라이센스) 배포

---

자유소프트웨어는 퍼블릭 도메인은 아니며 일반적으로 저작권과 라이센스가 있고 소유자가 있다. 이 때 카피리프트라는 아이디어를 사용하는데 이는 즉, 소프트웨어의 저작권이 보호받는다는 것이고 저작자들은 복사본의 재배포를 허용하며 수정 권한과 같은 추가 권한을 주지만 재배포할 때는 조항들에 의거해야만 한다는 것이다. 이것은 다른 사람과 협력할 수 있는 자유를 얻을 수 있는 것이다.

1989년 배포된 최초의 오픈소스 라이센스인 GNU 퍼블릭 라이센스(GPL)는 회사 보호 목적이 아닌 공동체의 관점에서 쓰여진 라이센스로 MIT ,BSD라이센스의 경우가 보여주듯이 정부 인증 프로그램이라는 목표를 수행하며 이는 단순한 라이센스가 아닌 완전한 철학 오픈 소스 정의에 모티브가 되기도 했다고 할 수 있다.

한편 1980년대를 통해 리처드 스톤먼이 GNU 프로젝트를 건설하는 동안 캘리포니아 대학 버클리 분교의 컴퓨터 과학자들은 버클리 유닉스 또는 BSD AT&T로부터 라이센스 받은 유닉스 커널에 기초 한자유로운 운영체제를 개발하고 있었다. 그러나 법적 문제, 소스코드의 분열로 해커들과 산업계 외부의 사용자들은 그것을 채택하는 것이 늦어졌다.

## 1989 자유소프트웨어 지원사업 CYGNUS

---

GNU/리눅스와 자유소프트웨어 운동의 성장에 있어 주요한 발걸음은 소프트웨어와 철학에 기초한 사업의 창출이었다. 1989년 마이클 티만은 GNU 자유소프트웨어에 관련한 컨설팅과 서비스를 판매 할 생각으로 CYGNUS를 창업하였고 원본 GNU 선언을 가지고 어떻게 돈을 벌지 생각했다. 리처드 스톤만은 자유소프트웨어 운동 초기부터 수익 낼 수 있는 여지가 있다고 믿었다.

자유소프트웨어의 이점 중 하나는 서비스나 지원을 위한 자유로운 시장이 존재한다는 점이였다. 독점 소프트웨어에선 지원은 독점 체계이고 소스를 가진 회사는 대체로 하나여서 그들만이 지원을 해줄 수 있기 때문에 대체로 독점의 횡포 위험이 존재했다. CYGNUS는 이와 같은 자유소프트웨어 지원사업으로 창업을 구상한 것이다.

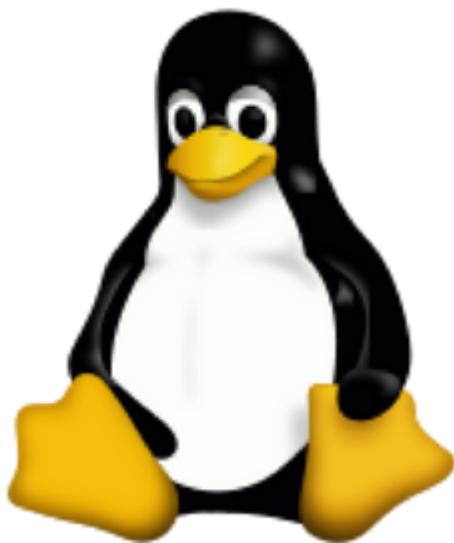


# 1991 Linux의 탄생

---

GNU프로젝트는 툴깃(일종의 도구모음)을 작성하면서 시작했고 이는 C 컴파일러(사람이 읽을 수 있는 숫자로 된 소스코드로부터 컴퓨터가 실제로 실행시킬 수 있는 숫자들로 된 프로그램으로 번역해주는 프로그램), 디버거, 텍스트 편집기/포메터와 같은 기본적인 개발 도구들이였다. 궁극적으로는 이러한 소프트웨어들 아래에서 동작하면서 운영체제의 중심이 되어 다른 모든 프로그램들이 사용하는 자원을 할당하는 프로그램인 커널을 개발하는 것이였다. 1990년대에 이르러 목표한 툴깃 개발에 성공하였고 다양한 유닉스 운영체제들에서 널리 사용되었다. 하지만 자유소프트웨어에 속하는 커널은 없었다.

이는 GNU 프로젝트 개발과정 중 마지막 부분에 속하였는데, 그 때 1991년 리누스 토발즈가 나타났다. 그는 핀란드 헬싱키 대학의 대학원생으로 하나의 커널을 개발했고 그것은 빠르며 훌륭하고 안정적으로 작동하였다. 초기의 목적은 개인적인 것으로 대학 컴퓨터에서 익숙해졌던 것과 유사한 것을 개인 PC에서 실행시킬 수만 있으면 좋겠다 싶어서 만든 것이였다. 그는 헬싱키 대학 2학년 시절 운영체제 수업을 듣고 있던 중, 심심한 나머지 교수가 취미로 만든 교육용 운영체제인 MINIX를 기반으로 자신이 갖고 있던 386 컴퓨터에서 돌아가도록 386보호모드에서 동작하는 리눅스를 개발하여 공개했다. 초반의 영감은 Sun OS로 핀란드 헬싱키 대학에서 사용 중이던 것이였다. 원래 리눅스는 MINIX를 좀 더 잘 써보려는 취미생활로 만든 거라 별 생각없이 GPL라이센스 기반으로 개발 했지만 이는 독자적 개발이고 GNU프로젝트 차원에서의 개발은 아니였다. 이 프로젝트를 GNU가 주목하고 프로젝트를 키워서 현재에 이르고 있다.



Linus Torvalds and Mascot of Linux

## 1992 LINUX의 개선

---

1992년, 리눅스는 0.95로 버전업 되었고, 인텔 x86칩에서 사용할 수 있었으며, 그래픽 사용자 인터페이스(GUI)가 추가되었다. GNU 커널로 개발 중이었던 Hurd의 개발이 순조롭지 않았던, 리처드 스톤만과 FSF는 유닉스 커널과 호환 가능한 커널인 리눅스를 GNU시스템의 커널로 채택하기로 했다. 리눅스는 강력한 GNU C컴파일러인 gcc로 컴파일된 많은 응용프로그램들을 가지게 되었고, 둘의 결합으로 GNU 시스템은 완전한 구조를 갖추게 되었다.

리눅스의 커널 부분은 리눅스 주도하에 개발되었는데, 리눅스는 최대한 확장 가능한, 즉 사용자에게 제어권이 있으며, 어떠한 인터페이스에도 종속되지 않도록 개발을 이끌고자 하는 의지가 있었다. 그는 리눅스의 성공의 원인을 훌륭한 설계 원칙과 좋은 개발 모델 덕분이라고 밝히고 있다.

1991-1993년 리눅스의 유년기라고 할 수 있는 때, 알파 또는 베타 수준으로 비교적 불안정하지만 상용 운영체제보다 안정적이고 전통적이며 검증된 방식을 이용하여 주어진 작업을 맡아서 처리하는 방식의 프로그램인데, 이는 모놀리식(monolithic)으로 기본적으로 운영체제 자체가 분할 할 수 없는 하나의 실체라는 것을 의미하는 것이었다. (반면 마이크로 커널은 단지 서버들의 집합체에 불과 서버들은 각각 서로 다른 일들을 수행하며 미리 약속된 규약 (프로토콜)에 따라 다른 서버들과 통신하는 방식이였다.) 1992,93년도 어느정도 필요한 모든 것이 다 갖춰진 그 때, 기존의 선과 웍스테이션을 대체 가능하며 1/3, 1/4가격으로 1/5~2배 빠른 작업 속도를 보여주었다.

## 1993 아파치 웹서버의 탄생

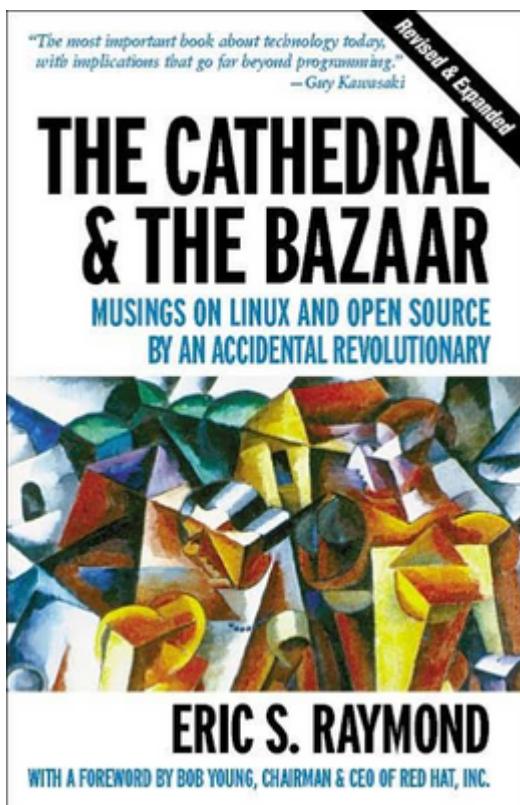
---

리눅스가 컴퓨터 프로그래머들의 세계에서 벗어나려면 리눅스를 필수 기술이 될 수 있도록 해주는 애플리케이션이 필요했다. 이러한 임계점을 지나게 해준 것은 아주 복잡한 웹사이트를 개발하는 데 필 요한 프로그램 아파치 웹서버의 등장이었다. 1993년 아파치 웹서버프로젝트가 시작되었을 때는 유명 ISP(Internet Service Provider)들이 생겨나고 있었을 때인데 IIS(Internet Information Service/Server)와 NT를 사용했을 때보다 리눅스와 아파치를 쓸 때 절감할 수 있는 가격적 이점에 따라 웹서버에서의 아파치 점유율이 증가했다. 또한 신뢰성과 유연성 그리고 확장성 등과 같이 웹마 스터들이 필요로 하는 것을 가지고 있었으며 아파치와 리눅스의 조합은 하나의 기계에 1개 이상의 웹 사이트를 호스팅 할 수 있는 능력과 같은 이점들이 분명히 존재했다.



## 리눅스 기술 지원 회사 등장

리눅스 성장의 중요한 요소 중 하나는 배포와 운영체제 자체에 대한 기술지원을 전문으로 하는 회사들의 등장이였다. 대표적인 예로 1995년 설립된 레드햇이 가장 유명하며 마크 유잉(Marc Ewing)이 좀 더 나은 리눅스 배포판을 원해서 만든 소프트웨어로 시작했다.



## 1997 “THE CATHEDRAL AND THE BAZAAR” by Eric Raymond

다음으로 중요한 역사 중 하나는 1997년 Eric Raymond에 의해 발표된 ‘시장과 성당’이라는 논문이다. 레이먼드는 이 글에서 리눅스 커널 개발 과정과 패치메일이라는 오픈 소스 프로젝트를 개발한 경험을 들어 오픈 소스 개발 방식의 유효성을 설명하였다. 그 때는 자유소프트웨어라는 말을 주로 사용하여 무엇이 자유소프트웨어 세계를 움직이는지 그리고 소프트웨어 공학에서 사용하는 모든 표준 규칙을 지키지 않으면서도 왜 우리가 초고품질의 소프트웨어를 만들 수 있었는지에 관한 것이였다.

두 가지 완전히 다른 개발 스타일이 있는데 하나는 이전의 폐쇄적 개발 스타일인 성당 스타일이다. 이 스타일에는 목표에 대한 상세한 명세와 소규모의 프로젝트 그룹이 체계적 권위주의적인 방법으로 운영된다고 한다. 릴리즈 간격은 매우 길다. 반면 개인대 개인으로 분산적이며 일종의 시장이나 장터 스타일의 매우 짧은 릴리즈 간격의 프로젝트로 외부의 사람들로부터 끊임 없는 피드백을 요구하고 상당히 집중적인 상호 검토 과정을 특징으로 가지고 있다.

기존의 폐쇄적 개발 방법의 모든 이점을 버리고 대량의 개별적 상호 검토라는 단 하나의 이점만을 추구하는 방법이 실제로 전자를 이기고 있으며 실제로 좋은 결과 가져다 주는 것처럼 보인다는 것이다.

## 1998 넷스케이프 소스코드 공개

---

시그너스가 지원을 해주고 있긴 하지만 사실 사업은 그다지 많지 않았기 때문에 넷스케이프는 오픈소스에 참여한 첫 대형 업체라는 점에서 중요하다. 그 때 당시 기본적으로 인터넷 익스플로러를 제공하고 있었던 마이크로소프트에 대항해서 넷스케이프는 오픈소스를 선택한 것으로 볼 수 있다. 시간이 지남에 따라 가격대나 점유율로 인한 http/html 표준 수정을 통해 독점 체계를 이용해서 서버시장에서 몰아내려 하는 등의 사업이 위협성을 걱정해 1998년 넷스케이프는 소스코드 릴리즈하기로 한다. 이는 1998년 초기 '성당과 시장'의 영향 있다고 볼 수 있다.

이 결정은 자유소프트웨어, 지금은 오픈소스라고 알려진 개념에 대한 공개적이 주목을 불러일으키고 오픈 소스의 가장 탁월한 사례 중 하나인 리눅스 운영체제에 대한 관심도 불러모았다.

## 1998 OSI(Open Source Initiative) 설립

---



넷스케이프의 소스 코드 발표에 대한 반응으로, 1998년 2월 3일 캘리포니아주 팔로알토에서 개최된 전략회의에서 Foresight Institute의 사장 Christine Peterson이 제안한 용어 'Open

Source'를 채택하였고, Eric Raymond와 Michael Tiemann을 공동 회장으로 하는OSI(Open Source Initiative)를 설립하였다.

오픈소스라는 용어를 채택한 이유는 사람들은 자유라는 말에서 공짜를 생각하며 돈을 벌 수도 없고 팔 수도 없는 전적으로 잘못된 생각을 갖고 있기에 소프트웨어가 공개 되어있고 소스코드를 사용할 수 있다는 개념으로 넘어서고자 했기 때문이다.

오픈소스 정의를 위한 메타라이센스 필요했고, 데비안 자유소프트웨어 가이드라인을 기반으로 한 것이다. 리눅스 배포판 프로젝트 정책을 만들고 오픈소스 정의에 따라 새로운 이름 붙였다. 오픈 소스는 오픈소스 정의에 따라 9가지 권리를 제공하는 것으로 정리되었다.

## 오픈소스 정의

OSI에서 제공하는 오픈소스 정의 원본 번역본은 다음과 같다.

오픈 소스는 소스 코드에 대한 액세스를 의미하지 않습니다. 오픈 소스 소프트웨어의 배포 조건은 다음 기준을 준수해야합니다.

### 1. 무료 재배포

라이센스는 여러 다른 출처의 프로그램이 포함 된 총 소프트웨어 배포의 구성 요소로서 소프트웨어를 판매하거나 양도하지 못하도록 제한하지 않습니다. 라이센스는 그러한 판매에 대해 로열티 또는 기타 수수료를 요구하지 않습니다.

### 2. 소스 코드

프로그램은 소스 코드를 포함해야하며 컴파일 된 양식뿐만 아니라 소스 코드로도 배포 할 수 있어야합니다. 어떤 형태의 제품이 소스 코드와 함께 배포되지 않는 경우 합리적인 복제 비용 이상으로 소스 코드를 얻고, 바람직하게는 인터넷을 통해 무료로 다운로드하는 방법이 있어야합니다. 소스 코드는 프로그래머가 프로그램을 수정하는 기본 형식이어야합니다. 의도적으로 난독화 된 소스 코드는 허용되지 않습니다. 전처리 기 또는 변환기의 출력과 같은 중간 양식은 허용되지 않습니다.

### 3. 파생 된 작품들

라이선스는 수정 및 파생 된 저작물을 허용해야하며 원본 소프트웨어의 라이선스와 동일한 조건으로 배포 할 수 있어야합니다.

### 4. 저자 소스 코드의 무결성

라이센스는 빌드 타임에 프로그램을 수정하기위한 목적으로 소스 코드와 함께 "패치 파일"의 배포가 허용되는 경우 예만 소스 코드가 수정 된 형태로 배포되는 것을 제한 할 수 있습니다. 라이센스는 수정 된 소스 코드로 빌드 된 소프트웨어의 배포를 명시 적으로 허용해야합니다. 라이센스는 파생 된 저작물이 원래 소프트웨어와 다른 이름 또는 버전 번호를 포함하도록 요구할 수 있습니다.

## 5. 개인이나 단체에 대한 차별 금지

면허는 어떤 개인이나 집단을 차별해서는 안됩니다.

## 6. 노력의 분야에 대한 차별 금지

면허는 특정 분야에서 프로그램을 사용하지 못하도록 제한해서는 안됩니다. 예를 들어, 프로그램이 비즈니스에서 사용되거나 유전 연구에 사용되는 것을 제한하지 않을 수 있습니다.

## 7. 라이센스 배포

프로그램에 첨부 된 권리는 프로그램이 재배포 된 모든 사람에게 해당 당사자가 추가 라이센스를 발급 할 필요없이 적용해야합니다.

## 8. 라이센스는 제품에 고유하지 않아야합니다.

프로그램에 첨부 된 권리는 프로그램이 특정 소프트웨어 배포의 일부가되어서는 안됩니다. 프로그램이 해당 배포에서 추출되어 프로그램 라이센스 조건에 따라 사용되거나 배포되는 경우 프로그램을 재배포하는 모든 당사자는 원래 소프트웨어 배포와 함께 부여 된 것과 동일한 권한을 가져야합니다.

## 9. 라이센스로 다른 소프트웨어를 제한해서는 안됩니다.

라이센스는 라이센스가 부여 된 소프트웨어와 함께 배포되는 다른 소프트웨어에 대한 제한을 두어서는 안됩니다. 예를 들어, 라이선스는 동일한 매체에 배포 된 다른 모든 프로그램이 오픈 소스 소프트웨어 여야한다고 주장해서는 안됩니다.

## 10. 라이센스는 기술 - 중립적이어야 함

라이센스의 조항은 개별 기술 또는 인터페이스 스타일에 근거 할 수 없습니다.

오픈 소스 정의는 원래 데비안 자유 소프트웨어 지침 (DFSG) 에서 파생되었습니다 .

이와 같이 오픈소스의 열기가 뜨거워짐에 따라 데이터 베이스제공자들이 태도를 바꿔서 오라클, 사이 베이스 그리고 다른 중요한 데이터베이스 제공자들이 리눅스로 포팅하기 시작했다. 이것이 중요한 이유는 오픈 소스, 특히 리눅스가 신뢰를 얻기위해서는 각각의 소프트웨어 회사들의 그들의 어플리케이션을 이쪽 플랫폼을 기반으로 포팅할 것을 약속해야 하기 때문이다.

## 1999 레드햇 소프트웨어 리눅스 회사 중 최초로 주식 상장

### 1999 VA LINUX SYSTEM 기업공개 및 공개적 주식 거래 시작

**■ Securities:** The computer-manufacturer's stock, sparked by a new investor frenzy over the operating system, soars 698% on first day of trading.

By ASHLEY DUNN  
TIMES STAFF WRITER

VA Linux Systems, a maker of high-end computers that run the hot Linux computer operating system, saw its shares soar a record-breaking 698% on its first day of trading.

ASHLEY DUNN/TIMES STAFF WRITER

**Leapin' Linux**

Since the initial public stock offering of Red Hat Inc. in August, the stocks of new Linux-related companies have soared. On Thursday, VA Linux Systems broke the stock market record for the largest one-day gain of an initial offering, closing at \$239.25 a share, up 698%. Linux-related companies, that have gone public recently, ranked by performance since 1999 offering:

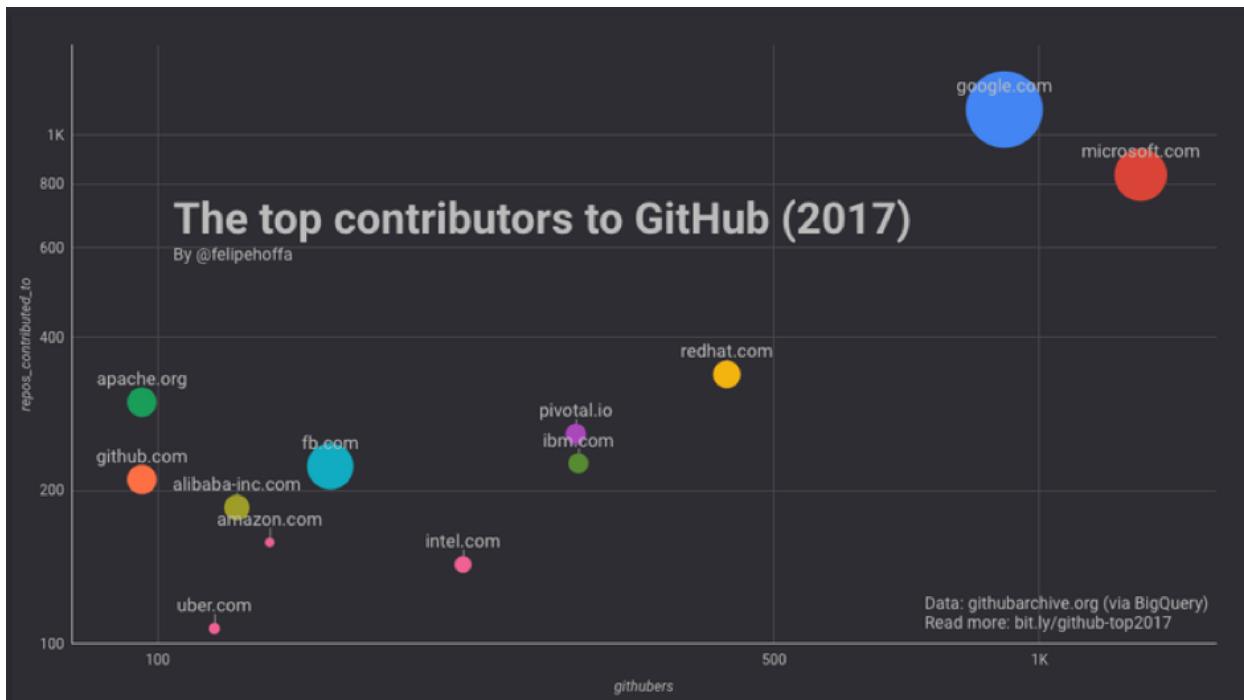
## 2000년대 이후

### 2005 Git 탄생

Linux 커널은 굉장히 규모가 큰 오픈소스 프로젝트로 Linux 커널의 삶 대부분은(1991-2002) Patch와 단순 압축 파일로만 관리했다. 2002년에 드디어 Linux 커널은 Bitkeeper라고 불리는 상용 DVCS를 사용하기 시작했다. 하지만 2005년에 커뮤니티가 만드는 Linux 커널과 이익을 추구하는 회사가 개발한 Bitkeeper의 관계는 틀어졌다. Bitkeeper의 무료 사용이 제고된 것이다. 이 사건은 Linux 개발 커뮤니티(특히 Linux 창시자 리누스 토발즈)가 자체 도구를 만드는 계기가 됐다. Git은 Bitkeeper를 사용하면서 배운 교훈을 기초로 아래와 같은 목표를 세웠다.

그것은 바로 빠른 속도, 단순한 구조, 비선형적인 개발(수천 개의 동시 다발적인 브랜치), 완벽한 분산, Linux 커널 같은 대형 프로젝트에도 유용할 것(속도나 데이터 크기 면에서)이다.

Git은 2005년 탄생하고 나서 아직도 초기 목표를 그대로 유지하고 있다. 그러면서도 사용하기 쉽게 진화하고 성숙했다.



2000년대 이후, 많은 기업들이 오픈소스를 사용하고 있으며 상용 소프트웨어 벤더에서도 오픈소스를 가져다가 사용하기 시작했다. 특히, 웹 기반 서비스(예: SNS) 업체에서 오픈소스 활용이 폭발적으로 증가했다.

구글은 오픈소스 기술에 가장 적극적으로 관여하는 세계적인 기업 중 하나로, 오픈소스 개발자 플랫폼인 Git Hub에는 900명이 넘는 구글 소속 공여자와 1000개가 넘는 구글 관련 저장소가 있다. 그 예로는 GWT(Google Web Toolkit) 등이 있다.

페이스북 또한 개발자 문화가 발달한 대표 기업으로 오픈소스 또한 장려되어 페이스북이 관리하고 있는 오픈소스 프로젝트 상당히 많다. 대표적인 예로는 React.JS가 있다.

이 외에도 Twitter, Netflix, Naver, kakao, Samsung 등 오픈소스에 참여하는 기업들과 오픈소스 문화 기반의 회사들 및 스타트업들이 많이 생겨나고 있다.

## 5. 오픈소스의 현황과 전망

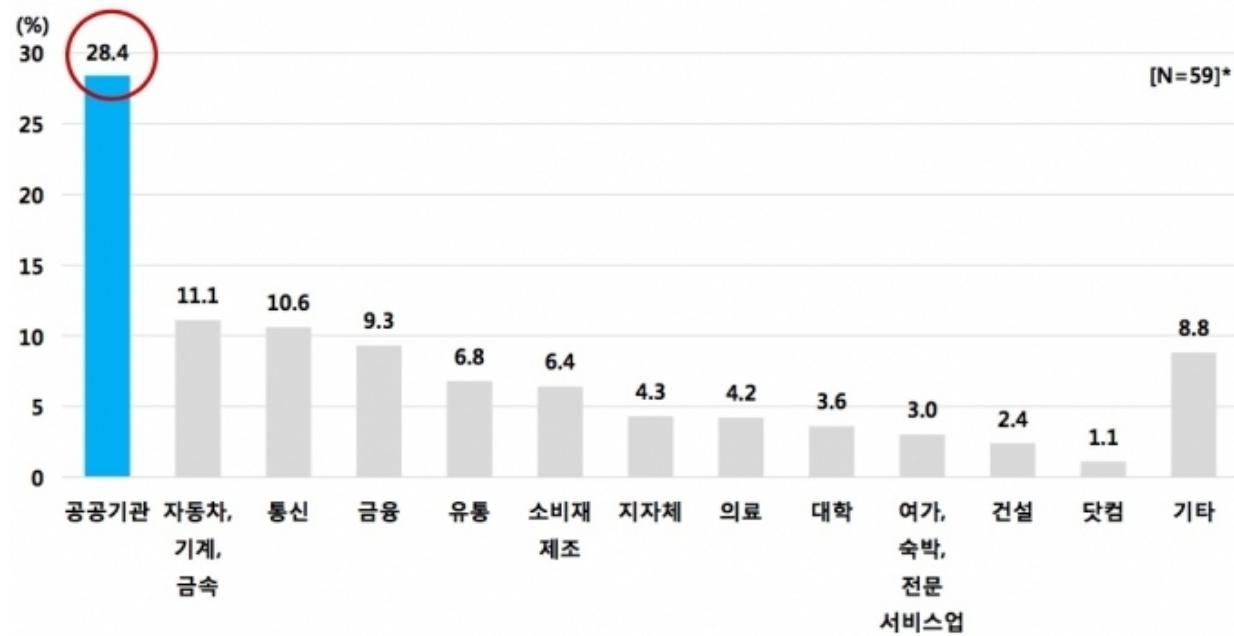
### 국내



◇ 오픈소스 SW 시장 규모 및 전망 [자료=정보통신산업진흥원]

국내 오픈소스 산업은 지속적으로 성장하고 있다.

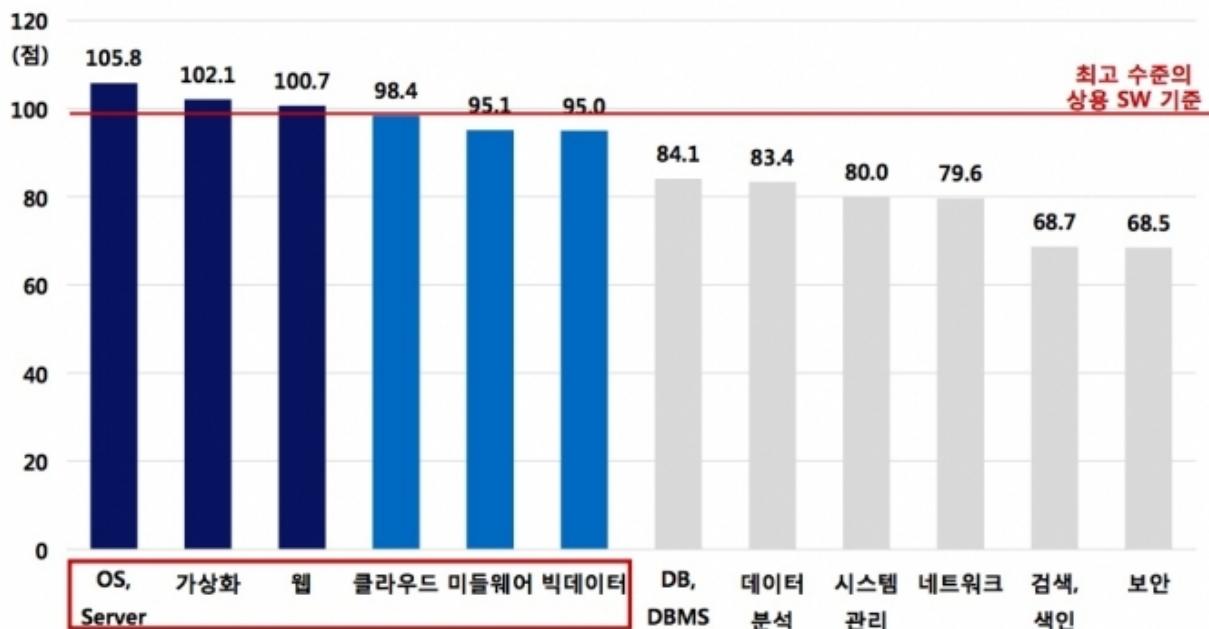
2016년 기준 연평균 성장률이 SW시장은 12%, 기술지원 시장은 17%로 이는 해가 갈수록 점차 늘어날 것으로 추정된다.



◇ 오픈소스 SW 주요 판매처

오픈소스 판매처 중 가장 큰 비중을 차지하는 곳은 공공기관이다.

이는 최근 정부에서 오픈소스 SW 활성화 정책을 추진함에 따라 공공기관이 오픈소스 SW를 적극적으로 도입하기 때문으로 풀이된다.



◇ 오픈소스 SW vs 비공개 SW 성능 수준 평가

오픈소스 SW 성능을 비공개 SW와 비교할 경우 OS, server, 웹에선 우위를 보였고, 클라우드, 미들웨어, 빅데이터 등의 분야도 상용 SW 수준에 근접한 것으로 나타났다. 반대로 시스템 관리, 네트워크 등의 분야에선 아직 뒤쳐지는 모습을 보였다. 이는 해당 분야가 상대적으로 다른 분야에 비해 폐쇄적인 모습(클로즈드 소스)을 보이는 것이 반영된 것으로 풀이된다.



◇ 오픈소스 SW 활용 시 장점

오픈소스 SW 활용 시 장점으로 "비용 절감"이 90.8%로 가장 높았다. 그 뒤로 44.1%는 최신 기술 적용을, 43.0%는 특정 벤더 종속 완화를 들었다. 그러나 "성능 우수", "엔지니어 능력 향상"을 장점으로 뽑은 비율은 각각 3.2%, 8.9%에 불과했다. 이는 오픈소스가 아직까지 성능면에선 상용 소프트웨어에 비해 뒤쳐져 있다는 인식을 반영한 것으로 보인다.

The screenshot shows the official website of the White House under President Barack Obama. The top navigation bar includes links for BLOG, PHOTOS & VIDEO, BRIEFING ROOM, ISSUES, the ADMINISTRATION, the WHITE HOUSE, and our GOVERNMENT. A banner at the top left reads "Your WEEKLY ADDRESS". Below it, a text box states: "The President restates his commitment to small business as key to economic recovery—from the Recovery Act to Financial Stability to Health Reform—and pledges more to come." A large video thumbnail of President Obama speaking is on the right. Navigation arrows at the bottom left indicate pages 1, 2, 3, and 4. On the left side, there's a section titled "A NEW FOUNDATION" with a photo of a speech and links to "Learn More" and "Watch the Video". On the right side, there's a search bar and a "PHOTO of the DAY" featuring a flag and a speech. The central column contains sections for "THE BLOG" and "FEATURED LEGISLATION".

- 미국

오바마 정부는 2009년 백악관 홈페이지를 100% 오픈소스를 이용해 만든바 있다. 백악관 이전에 일부 행정기관에서 오픈소스 SW를 사용하기도 했지만 백악관이라는 상징성 있는 기관에서 이를 채택한 것은 큰 의미가 있는 일로 받아들여졌다. 미국에서는 백악관 외에도 기상청, 미 항공우주국(NASA), 법무부, 국방부 등이 오픈 소스 SW 기반의 애플리케이션과 인프라를 도입했다.

- 영국

2011년 2월 IT정책을 총괄하는 국무조정실 빌 맥클루게이지 CIO가 “산업이 더 많은 오픈소스 기반의 솔루션 제공을 원한다”고 발언하면서 오픈소스 SW 도입 의지를 천명했다. 영국 정부의 오픈소스 SW 정책은 모든 제안에서 오픈소스솔루션 도입 정도를 평가하고, IT시스템의 핵심 요소로써 상호운용성을 포함하도록 하는 것이다. 이 외에도 일반적인 업무에서 오픈소스 SW 사용을 권장하고 있다.

- 러시아

오픈소스 SW 도입에 적극적인 태도로 임하고 있다. 작년 말 블라디미르 푸틴 러시아 총리는 2015년까지 정부기관의 정보망을 리눅스 기반 오픈소스 SW로 전환하는 계획에 서명했다. 통신부 주도로 정부기관 서버 OS와 드라이버, 애플리케이션을 오픈소스 SW 기반으로 전환한 러시아는 연방 재정 지원을 받는 모든 단체로 그 대상을 확대한다는 계획이다.

## 7. 오픈소스 주요 인물들

- **Richard Matthew Stallman**



**Richard Matthew Stallman**은 자유 소프트웨어 운동의 중심 인물이며, **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 세우는데 본질적인 영향을 준 인물이며, 이는 독점 소프트웨어 개발과 공급에 대한 대안이 되었다.

1985년, **스톨만**은 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, 리눅스 토르발스는 GPL로 리눅스 커널을 발표했다. 이를 통해 완벽하게 기능하는 GNU 시스템인 GNU/리눅스 운영 체제가 탄생하게 되었다.

**리처드 스тол만**의 정치적이고 도덕적인 동기는 그를 매우 논쟁적인 인물로 만들었다. 코드를 공유하자는 개념에 동의하는 프로그래머들 중 많은 수가 스тол만의 도덕주의적인 입장과 개인적인 철학에는 동의하지 않았다. 이러한 논쟁의 한 결과로 자유 소프트웨어 운동의 대안인 **오픈 소스 운동**이 생겨났다.

---

자유 소프트웨어 운동을 이끈  
리처드 스톤만

---



---

• **Linus Benedict Torvalds**



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

헬싱키 대학 2학년 시절 운영체제 수업을 듣고 있던 중, 심심한 나머지(Just for Fun), Andy Tannanbaum 교수가 “취미로” 만든 교육용 운영체제인 MINIX를 기반으로 자신이 갖고 있던 386 컴퓨터에서 돌아가도록 386 보호모드에서 동작하는 리눅스를 개발하여 공개했다. 이와 관련해 Monolithic Kernel 구조와 **MMicro Kernel** 구조에 대해서 타넨바움교수와 토르발즈간의 논쟁은 유명하다.

"Linus의 MINIX"라는 뜻을 담아 "리눅스(Linux)"라고 이름을 지었다.

원래 리눅스는 미닉스를 좀 더 잘 써 볼려는 취미생활로 만든 거라, 별 생각없이 그냥 공개한 것 같은데 이 프로젝트를 **GNU**가 주목하고 프로젝트를 확 키워서 현재에 이르고 있다. 아직도 커널부는 손수 작업한다고 한다.

참고로 리눅스 공개 자체로는 리눅스가 얻은 직접적 경제적 이익은 거의 없는 수준이다. 실제로 돈 벌려고 공개한 것도 아니기도 했고, 하지만, 이로 인해서 '자신의 이름이 붙은 운영체제'를 만들었다는 명예와 이로 인한 평생 직장을 보장받았다. 그리고, 더불어 레드햇 같은 리눅스 회사의 주식을 조금 받았는데, 처음 받을 땐 별 게 아니었지만, 나중에 레드햇 주가가 폭등하면서 부자가 되었다.

그의 종특은 맘에 안드는 것은 뭐든지 까는 것으로, 거친 언사도 서슴지 않으며, 특히 리눅스 개발 커뮤니티의 분위기를 살벌하게 만드는 일등 공신이 리눅스 본인인데, **오픈소스 개발**이라고 하면 누구나 편하게 자신의 코드를 커밋할 수 있을 것 같지만 현실은 시궁창으로, 리눅스에 함부로 손을 댄 사람은 리눅스에게 쌍욕을 먹고 멘붕을하게 되기 일쑤다.

소스 코드 리뷰에서 심각하게 결함이 있는 부분이나 마음에 안드는 부분을 가차없이 까내린다.

이는 공동 작업시 리눅스가 상대하는 사람의 기분보다는 작업의 결과물의 퀄리티를 더 중시하기 때문이다. 다시 말하면 코드 퀄리티를 강한 언사를 통해서 유지할 수 있다면 그쪽을 선택한다는 것.

IT쪽 인사들이 어느정도 이런 면들을 갖고 있기도 하지만 리눅스는 그 정도가 다소 극단적인 편이라고 보면 될 듯. 여기에는 같이 일하는 커널 개발자들이 대체적으로 에고가 강한 것도 한 몫 한다고. 다만 기여자 입장에서는 이런 강한 피드백을 받으면 기분이 많이 상하는 건 사실이라 커뮤니티 내에서 논란이 몇 번 일은 적이 있다.



---

- **William Henry Gates(Bill Gates)**



**William Henry Gates(Bill Gates)**는 미국의 마이크로소프트 설립자이자, 기업인이다. 어렸을 때부터 컴퓨터 프로그램을 만드는 것을 좋아했던 그는 하버드 대학교를 자퇴하고 폴 앤런과 함께 마이크로소프트를 공동창립했다. 그는 당시 프로그래밍 언어인 베이직 해석프로그램과 알테어용 프로그래밍 언어인 알테어베이직을 개발했다.

그는 13세 때 상류층 사립 학교인 레이크사이드 스쿨에 입학했다. 8학년이 되었을 때, 학교 어머니회는 자선 바자회에서의 수익금을 텔레타이프라이터 단말기와 제네럴 일렉트릭 (GE) 컴퓨터의 사용시간을 구매하는 데 사용하기로 결정하였다. 게이츠는 이 GE 시스템에서 베이직(BASIC)으로 프로그래밍하는 것에 흥미를 갖게 되었으며, 이에 프로그래밍을 더 연습하기 위해 수학 수업을 면제 받기도 했다.

그는 이 시스템에서 동작하는 틱택토 (Tic Tac Toe) 게임을 만들었는데, 이는 그가 만든 최초의 프로그램으로 사람이 컴퓨터를 상대로 플레이할 수 있게 되어 있었다. 또한 다른 게임인 달 착륙 게임을 만들기도 하였다. 그는 입력된 코드를 언제나 완벽하게 수행하는 이 기계에 매료되었다.

게이츠가 훗날 회고한 바에 따르면, 당시의 기억에 대해 그는 '그때 그 기계는 나에게 정말 굉장한 것이었다'라고 말했다. 어머니회의 기부금이 바닥나자, 게이츠와 몇몇 학생들은 DEC의 미니컴퓨터의 사용 시간을 샀다.

이 시스템 중 일부는 PDP-10이라는 것으로 컴퓨터 센터 코퍼레이션(CCC)에서 생산된 것 이었는데, 훗날 게이츠를 포함한 네 명의 레이크사이드 스쿨 학생(폴 앤런, 릭 와일랜드, 켄트 에번스)은 이 시스템의 운영 체제가 가진 버그를 이용해 공짜로 컴퓨터를 사용한 것이 발각되어 이 회사로부터 사용을 금지당하기도 했다. 심지어 컴퓨터를 1달동안 사용을 금지 하기도 하였다.

고등학교 졸업 후 하버드 대학교에 진학하여 응용수학을 전공했으나 재학 중 1975년 폴 앤런과 함께 **マイクロソフト**를 설립하고 학업을 중단했다. 당시에 그는 사업이 안 풀리면 학교로 돌아갈 예정이었으나 **マイクロソフト**의 성공으로 그럴 일은 없었다. 그래서 **빌 게이츠는 컴퓨터의 황제로 불리고 있다.**



## 8. 출처

---

- Revolution OS
- [Linuxjournal.com](http://Linuxjournal.com)
- [Opensource.org](http://OpenSource.org)
- Pro git
- <https://en.wikipedia.org/wiki/>
- 김병일, “오픈소스 라이선스 위반과 저작권 침해”, 한국저작권위원회
- 오병래, “오픈소스 기반의 프로젝트 개발환경 구축 방안 및 품질 향상 방안에 관한 연구”, 서강대학교 정보통신대학원
- 송주현, “오픈소스 소프트웨어의 가치와 경제적 영향에 대한 연구”, 연세대학교 법무대학원
- [https://namu.wiki/w/오픈\\_소스](https://namu.wiki/w/오픈_소스)
- [https://terms.naver.com/entry.nhn?  
docId=1228317&cid=40942&categoryId=32837](https://terms.naver.com/entry.nhn?docId=1228317&cid=40942&categoryId=32837)
- [https://terms.naver.com/entry.nhn?  
docId=2275543&cid=42238&categoryId=51178](https://terms.naver.com/entry.nhn?docId=2275543&cid=42238&categoryId=51178)
- <https://www.hankookilbo.com/News/Read/201804070914926234>
- <http://www.itworld.co.kr/news/96536>
- <https://www.oss.kr/news/show/04c08650-6200-42ca-b4e3-cf9f87beda67>
- <http://www.etnews.com/201109060344>
- <http://www.bloter.net/archives/18224>
- <https://www.slideshare.net/JerryJeong2/ss-58804386>
- <https://www.polarsys.org>
- <https://www.slideshare.net/JerryJeong2/ss-58804386>

- <http://www.itworld.co.kr/print/50540>
- [https://www.oss.kr/oss\\_trend](https://www.oss.kr/oss_trend)
- [https://en.wikisource.org/wiki/Open\\_Letter\\_to\\_Hobbyists](https://en.wikisource.org/wiki/Open_Letter_to_Hobbyists)
- <http://woodforest.tistory.com/229>
- <http://www.ddaily.co.kr/news/article.html?no=115248>