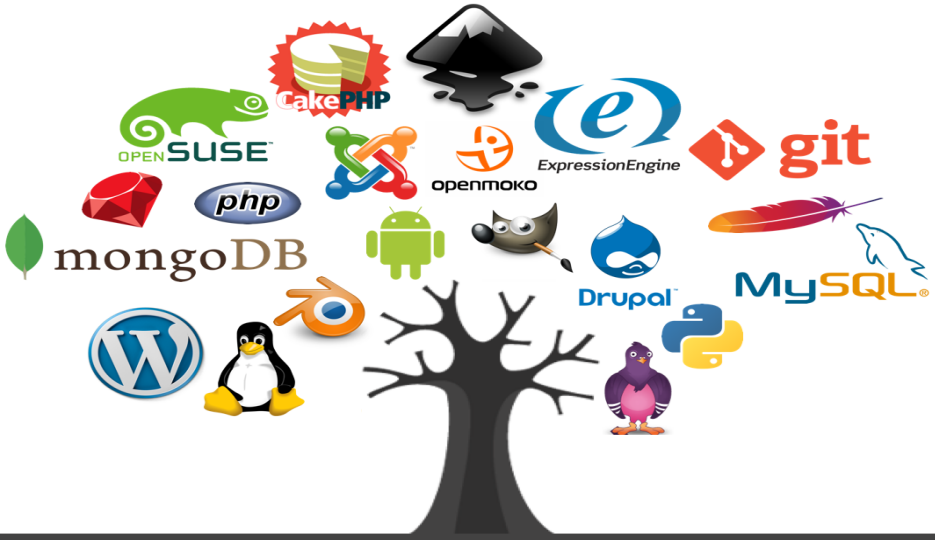
Open Source



서호진, 이규민, 이영섭, 이제신, 홍승현

목차

여는 말 2

오픈소스란? 3

오픈소스의 장점과 단점 6

오픈소스의 라이선스와 단체 9

오픈소스의 역사 16

마치며... 29

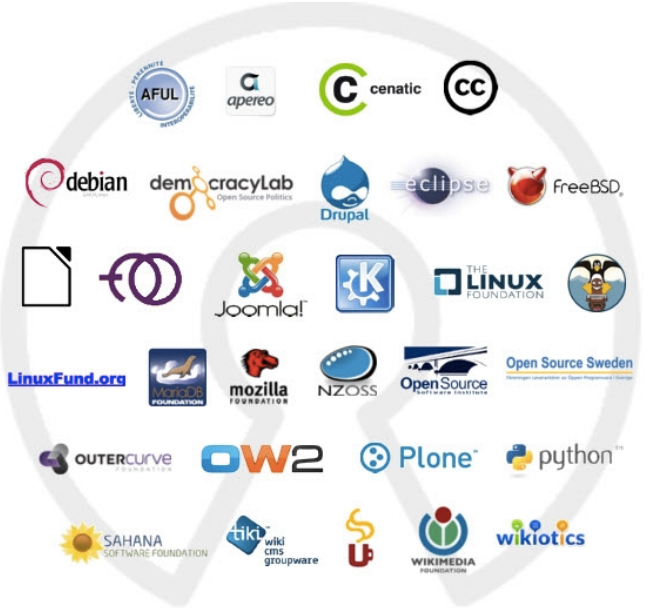
여는말

"개발 테스트를 하는 오픈 소스 소프트웨어 프로젝트는 전체 산업의 기대치를 높이며 소프트웨어의 품질을 향상시킨다" -소프트웨어 개발 테스팅 전문기업 커버리티(Coverity)의 잭 사모차(Zach Samocha)제품 수석 이사-

"실제로 버그를 발견한 개발자와 같은 다양한 개발자의 지원을 모두 받을 수 있기 때문에 실제로 오픈소스 소프트웨어는 매우 큰 이점" - 블룸버즈 비즈니스 위크의 조던 로버트슨(Jordan Robertson)-

오픈 소스는 무상으로 공개된 소스코드 또는 소프트웨어를 의미하며 오픈 소스 소프트웨어 혹은 OSS 라고 불린다. 위 두 사례에서 봤듯이 오픈 소스 소프트웨어는 현재 기업 혁신의 아이콘으로 부상하고 있으며 기업 간 경쟁의 승패를 판가름하는 중요 수단이 되었다는 분석까지 나오고 있다. 당연하게도 오픈소스 소프트웨어는 우리의 실생활에서 매우 자주 쓰이고있지만 일반적은 사람들의 오픈소스에 대한 관심도는 매우 미미한 편이다. 이에 따라 오픈소스에 대한 관심도를 조금이나마 증진시켜보고자 이 소책자를 작성하게 되었다.

오픈소스란?



오픈소스는 어떤 제품을 개발하는 과정에 필요한 소스 코드나 설계도를 누구나 접근 해서 열람할 수 있도록 공개하는 것이다. 보통 소스가 공개된 소프트웨어를 오픈 소스 소프트웨어라고 하며, 소프트웨어 말고도 개발 과정이나 설계도가 공개되는 경우 하드웨어에도 오픈 소스 모델이 적용 가능하며, 글꼴과 같은 데이터에도 오픈 소스 개발 모델이 적용되는 경우가 있다. 단순히 소스를 공개만 하는

것이 아니라, 이를 2 차 창작하는 것을 허용하기도 하고, 나아가 조건 없이 상업적 용도로도 사용할 수 있게 하는 경우가 있다.

오픈소스 소프트웨어는 바로 소스 코드가 공개된 프로그램이다. 대부분의 오픈 소스 소프트웨어는 무료로 사용 가능 하기 때문에 프리웨어와 많이들 혼동하는데, 프리웨어는 무료로 사용 가능한 프로그램이고, 오픈소스는 소스코드가 공개된 프로그램이기 때문에 엄연히 다른 개념이다. 자유 소프트웨어와 비슷하지만, 오픈소스 소프트웨어가 자유소프트웨어보다 조금 더 상위의 개념이다. 쉽게 풀어서 설명하자면 통상적으로 자유 소프트웨어는 자유 소프트웨어 재단과 GNU 프로젝트와 관련된 소프트웨어에서 자유를 중시하는 의미에서 사용하고, 오픈소스 소프트웨어는 소스의 형태 자체를 중시하는 말이다.

일부 소프트웨어는 그것을 만든 사람, 팀 또는 조직 만이 독점적으로 제어 할 수 있는 코드를 가지고 수정할 수 있다. 사람들은 이러한 종류의 소프트웨어를 "독점(proprietary)" 또는 "클로즈드 소스(closed source)" 소프트웨어 라고 부른다. 독점 소프트웨어의 원래 작성자만 해당 소프트웨어를 합법적으로 복사,검사 및 변경을 할 수 있다. 독점 소프트웨어를 사용하려면 컴퓨터 사용자는 소프트웨어 작성자가 명시적으로 허용하지 않은 소프트웨어로는 아무 것도 하지 않을 것이라는 점에 동의해야한다.

예를 들어, MicrosoftOffice 나 Photoshop 같은 소프트웨어를 처음 실행할 때 라이센스에 서명해야한다. 그러나 오픈소스 소프트웨어는 그와 다르게 일반 사용자 (User)입장에서는 프리웨어나 오픈소스 소프트웨어나 단순하게 공짜로 사용하는 일반 사용자 입장에서는 프리웨어나 오픈 소스 소프트웨어나 단순히 공짜로 사용할 수 있다는 점에서는 비슷할 수 있지만, 소스코드를 보고 이해할 수 있고, 수정할 수 있는 개발자 입장에서는 크게 다르다. 예를 들어, 상용 또는 프리웨어 프로그램을 사용하는 사람들은 버그를 발견했다 하더라도 소스 코드를 모르니 수정할 수 없고, 사용자가 새로운 아이디어가 떠올랐다 해도 그것을 곧바로 프로그램에 적용시킬 수도 없다. 비교적 간단한 프로그램은 리버스 엔지니어링으로 어셈블러 수준에서 뜯어고칠 수는 있으나 코드가 공개된 것보다 몇 백 배는 어렵기도 하고, 저작권 같은 문제가 얽히고설키기에 하려는 사람은 없다고 보면 된다. 하지만 사용자가 프로그래밍 언어를 아는 경우 소스가 공개되어 있다면 본인이 직접 소프트웨어의 문제를 수정하거나 개선을 할 수 있게 되는 것이다. 또한, 개발하던 프리웨어가 개인적인 사정이나 회사의 사정에 따라 개발이 중지되면 그대로 사장되는 경우가 종종 있는데, 오픈 소스 소프트웨어는 소스가 공개되어 있기 때문에 다른 개발자/개발사에서 이를 이어 받아서 새로이 개선해 나가면서 개발하는 것이 된다. 그래서 개발자와 사용자가

일치하는 개발 및 시스템, 네트워크 분야에는 웬만한 클로즈드 소스 상용 소프트웨어는 명함도 못 내밀 정도로 고품질의 오픈 소스 소프트웨어가 넘쳐난다. 그러나 그러지 않는 분야에선 말 그대로 취미 수준에 머물러 있는 경우도 많다.

오픈소스의 장점과 단점

오픈소스의 개념은 소스코드를 공개하여 유용한 기술을 공유함으로써 전세계의 누구나 자유롭게 소프트웨어를 개발하거나 업그레이드 하는 것에 참여할 수 있어 특정의 기술자가 소프트웨어를 개발하는 것보다 단기간에 더 안정적인 소프트웨어를 개발할 수 있다는 개념에서 추진되었다.

오픈소스의 장점은 무료로 다운로드 및 수정과 재배포가 가능하므로 일반적으로 소프트웨어의 초기 개발비용이 적게 들어간다. 기술적인 측면에서는 최신 기술 정보 및 문제점과 해결책을 공유하는 형태로 자유롭게 운영되기 때문에 독점 프로그램에 비해 기술발전 속도가 빠르다. 또한 주로 오픈 포맷 또는 프로토콜을 사용하기 때문에 서로 다른 소프트웨어간 상호연동성이 보장되고, 오픈소스의 개발 과정에서 전세계에 있는 수많은 우수한 개발자들이 직접 개발과 디버깅 과정에 참여하기 때문에 In-house 에서 폐쇄적으로 개발되는 독점 프로그램에 비해 비교적 안정적으로 동작한다.

이러한 장점으로 인해 운영체제(Operating System)는 말할 것도 없고 웹, 데이터베이스, 그래픽, 클라우드, 컴퓨팅 등 실로 다양한 분야에서 다양하고 유용한 오픈소스를 쉽게 접할 수 있다. 예를 들어 우분투(Ubuntu), 파이어폭스(Firefox), 마이 SQL(MySQL), 버추얼박스(VirtualBox), 7 집(7-zip) 등이 대표적이다.

오픈소스는 장점도 있지만 단점도 있다.

오픈소스는 누구나 자유롭게 이용하기때문에 이로 인해 체계적이지 못한 소프트웨어들이 발생할 수 있다. 또한 개발한 소프트웨어를 유료화 할 수 없어 수익을 창출하지 못하면서

오히려 다른 사람이 도용한 코드로 유료화 할 위험도 있다. 개발자들이 선의로 배포한 소스코드를 상용 소프트웨어 제작사들이 도용해서 유료 프로그램을 제작하는 사태도 벌어진다. 그리고 오픈소스 라이선스의 미준수로 인해 저작권법 위반 및 특허소송의 위험에 놓일 수도 있다. 또다른 단점으로는 무료로 사용되고 있는 오픈소스 소프트웨어이기

오픈소스 라이선스와 단체

소프트웨어가 1980 년대 이후 고부가 가치 산업으로 성장함에 따라, 지식재산권 및 라이선스계약을 통하여 소프트웨어의 복제, 배포 ,수정에 대하여 제한을 가하려는 움직임이 나타났다. 이러한 움직임에 반대하여 리처드 스톨만은 "누구나 자유롭게 '실행, 복사, 수정, 배포'할 수 있고, 누구도 그런 권리를 제한하면 안 된다"라는 GUN 프로젝트를 전개한다.

스톨만은 이러한 프로젝트를 여러 방면으로 지원하기 위해 자선단체인 자유 소프트웨어 재단(FSF, Free Sofrware Foundation)을 창설하였으며, 설립 이후부터 1990 년대 중반까지 자유 소프트웨어 재단 기금은 GNU 프로젝트의 자유 소프트웨어를 작성하기 위해 소프트웨어 개발자를

고용하는데 대부분 사용되었다. 1990 년대 중반 이후로 이 재단의 직원들과 자발적인 기여자들은 대개 자유 소프트웨어 운동과 자유 소프트웨어 커뮤니티를 위한 법적, 구조적 문제에 대한 작업을 처리하고 있다.

FSF 는 GNU 프로젝트로 배포된 프로그램을 법적으로 보호할 목적으로 GPL(General Public License)라는 라이선스를 제작하게 된다. GPL 은 누구에게나 아래의 다섯 가지의 의무를 저작권의 한 부분으로서 요구한다.

- 프로그램을 어떠한 목적으로든지 사용할 수 있다. 다만 법으로 제한하는 행위는 할 수 없다.
- 프로그램의 실행 복사본은 언제나 프로그램의 소스 코드와 함께 판매하거나 소스코드를 무료로 배포해야 한다.
- 프로그램의 소스 코드를 용도에 따라 변경할 수 있다.
- 변경된 프로그램 역시 프로그램의 소스 코드를 반드시 공개 배포해야 한다.

• 변경된 프로그램 역시 반드시 똑같은 라이선스를 취해야 한다. 즉 GPL 라이선스를 적용해야 한다.

GPL 은 현재 가장 많은 오픈소스 소프트웨어가 채택하고 있는 라이선스로 오픈소스 라이선스들 중에서 가장 많이 알려져 있고 의무사항들도 다른 라이선스에 비해 엄격한 편이다. 현재까지 1989 년 1 월에 GPL-1.0, 1991 년 6 월 GPL-2.0, 2007 년 6 월 GPL-3.0 총 3 가지의 버전이 발표 되었다. 적용 사례로는 리눅스 커널, 깃, 마리아 DB 등이 있다.

1990 년대 GPL-2.0 이 발표될 때 FSF 는 동시에 LGPL(the Library General Public License)이라는 라이선스도 발표하게 된다. 이 라이선스는 조금 약화된 GPL 로써 당시 소프트웨어 라이브러리에 대하여 GPL 이 너무 엄격하며, 비효율적이라는 의견의 많아짐에 따라 반영된 결과이다.

GPL 과 LGPL 의 차이점은 적용 범위에 있다. GPL 은 라이브러리를 포함하여 모든 프로그램에 사용되며, LGPL 은 라이브러리에만 국한된다. 또한 LGPL 은 GPL 로 변경될 수 있지만, 한번 GPL 로 선언된 코드는 LGPL 로 변경될 수 없다는 점이 있다.

또 다른 차이점으로는 공개의 여부 이다. LGPL 은 코드를 정적 또는 동적 라이브러리로 사용한 프로그램을 개발하여 판매/배포할 경우에는 프로그램의 소스코드를 공개하지 않아도 되며 LGPL 코드를 사용했음을 명시만 해주면 된다. 단,

코드를 단순히 이용하는 것이 아니라 이를 수정 또는 이로부터 파생된 라이브러리를 개발하여 배포하는 경우에는 전체 코드를 공개해야 한다.

원래는 한정된 라이브러리에만 적용하려는 의도로 'Library GPL'이라는 이름을 붙였으나, 모든 라이브러리에 적용된다는 오해를 사 1999 년 2.1 버전(LGPL-2.1)으로 발전하면서 'Lesser GPL'로 변경되었다. 적용 사례로는 모질라 파이어 폭스가 있으며 아래는 LGPL 의 배포시 의무사항을 나타낸다.

• 각 복제본에 적절한 저작권 고지와 보증책임이 없음을 명시해야한다.
• LGPL-2.1 라이선스를 언급하는 고지사항과 보증책임 관련 고지사항을 원본 그대로 유지해야한다.
• 프로그램을 양도 받는 모든 이들에게 프로그램과 함께 LGPL 라이선스 사본 제공해야한다.
• 라이브러리 형태로의 수정을 허용하며, 수정사실과 날짜를 파일에 명기해야한다.
• 원본저작물과 파생저작물을 LGPL 또는 GPL 에 의해 배포되어야 한다.
• 원본저작물 및 파생저작물에 대한 소스코드를 제공하거나, 요청시 제공하겠다는 약정서 제공해야한다.
• 응용프로그램을 배포할 경우, LGPL 라이브러리를 사용하고 있다는 사실을 명시해야한다.

사용자가 라이브러리를 수정해도 응용프로그램을 사용할 수 있도록 허용해야한다

시간이 지남에 따라 자유 소프트웨어의 '자유'라는 단어가 일반인들에게 '무료'로 인식되고, 까다로운 GPL 조항 때문에 상용 SW 개발에 이용할 수 없어 대다수 기업들이 자유 소프트웨어 운동에 참여하기를 꺼려하자 소스코드 공개에 보다 많은 참여를 이끌어내기 위하여 에릭 레이먼드, 브루스 페런스 등은 '오픈소스'라는 새로운 용어를 제안했다.

이러한 '오픈소스'는 1998 년 오픈소스 SW 활성화 및 오픈소스 SW 에 대한 인증을 담당하는 OSI(Open Source Initiative)가 결성되면서 널리 사용되기 시작했다. OSI 는 오픈소스에 해당하는 라이선스의 최소한의 기준을 정의(OSD : Open Source Definition) 해놓고 이 정의에 따라 인증, 관리 및 촉진시키는 일을 한다.

OSI 에서 인증하는 라이선스들로는 GPL-2.0, GPL-3.0, LGPL-2.1 등 약 84 가지가 존재하며, 그 중 위에서 나열한 3 가지의 라이선스와 BSD(Berkeley Software Distribution) License,

Apache License 2.0, MIT License 가 가장 널리 쓰이는 라이선스이다.

BSD (Berkeley Software Distribution) License 는 버클리의 캘리포니아대학에서 제작한 라이선스로 소프트웨어 라이선스라고도 할 수 없을 만큼 효력이 미미하다. 이 라이선스 아래에서 제작된 소프트웨어는 아무나 개작할 수 있고, 수정한 것을 제한 없이 배포할 수 있다. 또한 수정본의 재배포는 의무적인 사항이 아니므로 소스코드의 공개를 꺼려하는 상용 소프트웨어에서도 사용할 수 있다.

GPL 의 경우에는 자유 소프트웨어의 철학("자유")을 실천하고자 하는 의도가 반영되어 있다고 할 수 있지만, BSD 라이선스는 그 자체가 공공기관에서 만들어낸 것이므로 공공의 몫으로 돌려주자는 의미가 강하기 때문에 라이선스 자체에는 아무런 제한 없이 누구나 자신의 용도로 사용할 수 있도록 한 것이다.

Apache License 2.0 은 1999 년 6 월에 창단된 ASF(Apache Software Foundation)에서 제작한 라이선스로 ASF 의 모든 소프트웨어에 적용되며, BSD 와 마찬가지로 소스코드 공개의 의무가 발생하지 않는다. 다만 "Apache"라는 이름에 대한 상표권을 침해하지 않아야 한다는 조항이 명시적으로 들어가 있고, 특허권에 관한 내용이 포함되어 있어 BSD 보다 법적으로 완결된 내용을 담고 있다.

MIT License 는 매사추세츠 공과대학교에서 자기 학교의 소프트웨어 공학도들을 돕기 위해 제작한 라이선스로 X11 License 또는 X License 로 표기되기도 한다. 이 라이선스는 BSD 라이선스를 기초로 작성된 결과물로써 이 라이선스 아래에서 제작된 소프트웨어들을 개조한 제품을 반드시 오픈 소스로 배포해야한다는 규정이 없으며, GPL 의 엄격함을 피하려는 사용자들에게 인기가 있다.

오늘날 우리는 범람하는 오픈소스 라이선스들 사이에서 살아가고있다. 우리는 이러한 라이선스들을 올바르게 인지해야 하며, 오픈 소스를 사용할 때는 적절한 절차에 의하여 사용해야한다. 그렇지않으면 오픈소스 가지고 오는 엄청난 자유가 법적 분쟁이 되어 돌아오는 양날의 칼이 될 수 도 있다

오픈소스의 역사

1950 년대와 1960 년대에 거의 모든 소프트웨어는 공동 작업으로 일하는 학자 및 기업 연구원에 의해 만들어졌고, 종종 public-domain-software 로 공용됐기 때문에 일반적으로 소스는 배포되었으며, 오픈소스는 필수적이지는 못했다.

1953 년 UNIVAC 의 레밍턴 랜드의 A-2 시스템은 소스코드를 고객에서 공개하였고 자유 오픈소스 소프트웨어의 첫번째 예라고 여겨진다.

1969 년 MIT 대학과 벨 연구소에서 멀틱스(Multics)라는 여러 사용자를 동시에 지원하고 기능이 많은 우수한 운영체제인 운영 체제를 개발하였다. 하지만 실험적 첨단 기능이기 때문에 너무 크고 복잡하였고 GE/Honeywell 의 비싼 대형 컴퓨터 에서만 돌아갔다. 이후 AT&T 의 벨 연구소의 켄 톰슨과 데니스 리치는 1970 년대 초부터 멀틱스의 편리한 기능을 계승하면서도 연구실 수준에서 살 수 있는 DEC PDP 계열의 미니 컴퓨터에서도 돌아갈 수 있도록 기능을 대폭 축소하고 단순화시킨 운영체제를 개발하였고 이를 유닉스(Unix)라고 하였다.

1975 년 BASIC 은 조직적 추진없이 공동 개발 되었던 소프트웨어의 다른 예이다. BASIC 은 'Beginner's All-purpose Symbolic Instruction Code(초보자용 다목적 기호명령 코드)'의

약자이다. 1963 년 다트머스 대학교의 존 케메니(John Kemeny)와 토머스 커츠(Thomas Kurtz)가 개발하였다. 본래는 대화형 메인프레임 시분할 언어로 설계되었으며, 퍼스널 컴퓨터에 채용됨으로써 널리 사용되는 언어가 되었다. 처리속도가 느리다는 단점이 있지만 배우기 싶고 수정이 간단한 등 장점이 많아 사용이 많이 됐으며 '마이크로소프트'에서 베이직 인터프리터를 만들었다.

또한 개방형 표준과 매우 유사하게 연구 기관 ARPANET (Advanced Research Projects Network)에 접근 할 수 있는 연구원은 전기 통신 네트워크 프로토콜을 개발하기 위해 Request for Comments 라는 프로세스를 사용했다. 이 덕분에 인터넷의 탄생을 가져 왔다.

Donald Knuth 의 TeX 와 SPICE 등의 소프트웨어도 개발되어 사용되었다.

1960 년대 후반에는 운영 체제 및 프로그래밍 언어 컴파일러가 발전하면서 소프트웨어 생산 비용이 하드웨어에 비해 크게 증가했다. 점차 증가하는 소프트웨어 산업은 하드웨어 제조업체의 번들 소프트웨어 제품과 경쟁하고 있었으며 리스 컴퓨터는 소프트웨어 지원을 필요로하면서 소프트웨어에 대한 수익은 없고 자신의 요구를 더 잘 충족시킬 수 있는 고객 제조업체의 소프트웨어 비용이 하드웨어 제품 비용과 함께 드는 것을 원하지 않았다. 1969 년 1 월 17 일 미국과 IBM 의 반독점 소송에서 미국 정부는

번들 소프트웨어가 반 경쟁적이라고 비난했다. 일부 소프트웨어는 무료로 계속 제공되지만 제한적인 라이센스 하에서 만 판매되는 소프트웨어가 점점 늘어나고 있다.
1970 년대 초에 AT & T 는 정부와 학술 연구자에게 유닉스 초기 버전을 무료로 배포했지만, 이것은 수정 된 것을 재배포하거나 배포 할 수 있는 권한이 없었기 때문에 현대의 의미에서 자유 소프트웨어 가 아니었다. UNIX 가 1980 년대 초반에 널리 보급 된 후 AT & T 는 무료 배포를 중단하고 시스템 패치를 부과했다. 다른 아키텍처로 전환하는 것이 매우 어렵기 때문에 대부분의 연구원은 상용 라이센스를 지불했다..
1970 년대 후반과 1980 년대 초반에 컴퓨터 벤더와 소프트웨어 전용 회사는 소프트웨어 라이센스에 대한 부과를 시작하고 소프트웨어를 "프로그램 제품"으로 마케팅하고 저작권 , 상표 및 임대를 통해 자산으로 간주되는 새로운 소프트웨어 개발에 법적 제한을 가하기 시작했다. 계약. 1976 년에 빌 게이츠 는 애호가들에게 공개 서한 이라는 제목의 에세이를 작성했다. 이 책 에서 애용자 가 라이선스 비용을 지불하지 않고 Microsoft 의 Altair BASIC 제품을 광범위하게 공유하면서 낙담했다. AT & T 는 1979 년 Unix 시스템을 판매함으로써 수익을 올릴 수 있다고 결정했을 때 라이선스를 시행하기 시작했다. IBM 은 1983 년 2 월 8 일자

발표 서에서 구입 한 소프트웨어로 더 이상 소스를 배포하지 않겠다는 정책을 발표했다.

1955 년에 설립 된 SHARE 사용자 그룹은 무료 소프트웨어를 수집하고 배포하기 시작했다. "SHARE Program Library Agency"(SPLA)는 정보와 소프트웨어, 특히 자기 테이프를 배포했다.

1980 년대에는 자유 소프트웨어 운동과 나란히 소스 코드가있는 소프트웨어가 BBS 네트워크 에서 공유되었다 . BASIC 및 기타 해석 언어로 작성된 소프트웨어는 소스 코드로만 배포 할 수 있었으며 그 대부분은 프리웨어였다. 사용자가 그러한 소스 코드를 수집하고 수정을 논의하기 위해 특별히 보드를 설정하기 시작했을 때 이것은 사실상 오픈 소스 시스템이었다.

가장 확실한 사례 중 하나는 가장 많이 사용 된 BBS 시스템 및 네트워크 중 하나 인 WWIV 이다. 처음에는 Wayne Bell 이 BASIC 에서 개발했습니다 . 자신의 소프트웨어를 "modding"하고 mods 를 배포하는 문화는 매우 커져서 소프트웨어가 첫 번째 Pascal , 그 다음 C ++ 로 포팅되면 등록된 사용자에게 배포되고 계속해서 mods 를 공유하고 자신의 코드를 컴파일한다.

1980 년대 초 Usenet 과 UUCPNet 의 출현으로 프로그래밍 공동체가 더욱 깊어 지고 프로그래머가 소프트웨어를

공유하고 다른 사람들이 작성한 소프트웨어에 기여할 수 있는 더 간단한 방법을 제공하게 되었다.

1980 년대 초반 APPANET 과 UNIX 해커들 커뮤니티들은 UNIX 와 C 에 수렴하기 시작했고 1984 년 mit 해커인 리차드 스톨만은 '자유 소프트웨어'를 증진시키고 X 윈도우 프로젝트가 mit 에서 UNIX 를 위한 GUI 를 개발하기 위해 시작한 대부분의 UNIX 를 향상시키기 위해서 'GNU 프로젝트'를 시작하였다. 이것의 목표는 컴퓨터 사용자가 자유롭게 다음과 같은 자유권에 기반한 소프트웨어를 개발 및 제공하여 컴퓨터 및 컴퓨팅 장치 사용을 자유롭고 통제할 수 있도록 하는 것이다.

Richard Stallman 은 1983 년 GNU 프로젝트를 시작하여 소스 코드의 사용에 제약이 없는 완전한 운영 체제를 작성했다. 이 사건은 유저들이 소스코드를 주지 않아서 프린터기가 수리될 수 없어 발생하였다. 스톨만은 또한 1985 년에 GNU 프로젝트의 목적을 설명하고 자유 소프트웨어의 중요성을 설명하기 위해 GNU 선언문 을 발표했다 . GNU 프로젝트와 이것의 성명서가 나온 실질적인 이유는 리차드 스톨만과 심볼릭 Inc 사이의 심볼릭스가 MIT 코드를 기반으로하는 Lisp 머신에 대한 업데이트에 MIT 접근법을 제공하는 것에 대한 의견 충돌이었다. 출시 직후, 그는 기존의 "자유 소프트웨어"라는 용어를 사용하고 그 개념을

홍보하기 위해 자유 소프트웨어 재단을 설립했다. 자유 소프트웨어 정의는 1986 년 2 월에 출판되었다..

1989 년에 GNU 일반 공중 사용 허가서 의 첫 번째 판이 출판되었다. 업데이트 된 버전 2 가 1991 년에 출판되었다. 1989 년에 일부 GNU 개발자가 Cygnus Solutions 라는 회사를 결성했다. "GNU Hurd"라고 불리는 GNU 프로젝트의 커널은 계속 지연되었지만 대부분의 다른 구성 요소는 1991 년에 완성되었다. 특히 GNU Compiler Collection 과 같은 일부 제품은 시장 리더가 되었다. GNU Debugger 와 GNU Emacs 도 주목할만한 성공을 거두었다.

리눅스는 1991 년 핀란드 헬싱키 대학의 리누스 토발즈(Linus Benedict Torvalds)라는 학생에 의해 만들어진 운영체제이다. 당시 21 살의 대학생이었던 리누스 토발즈는 앤디 타넨바움(Andy Tanenbaum)교수가 학생들의 학습을 주 목적으로 개발한 미닉스(MINIX)를 사용하던 중에 유닉스와 호환되는 공개된 운영체제의 개발 계획을 MINIX 사용자 모임에 발표하였다. 이 대학생이 개발하던 운영체제는 앞에 소개했던 GNU 시스템에 적합한 커널이었고 이것이 바로 리눅스이다. 또한 그는 개발한 운영체제가 "단지 취미이며 GNU 처럼 거대하거나 전문적인 것은 아니다."라며 프로젝트 초기의 의도를 밝혔고 그로 인해 초기의 리눅스는 이식성이 고려되지 않은, 다만 i386 계열에서 운영되는 유닉스 호환 운영체제를 목표로 하는 프로젝트였다. 초기 버전 0.01 은

1991 년 9 월 17 일 인터넷을 통해 공개되었고, 가장 기본적인 커널만을 포함하고 있었으며, 실행조차 되지 않는 수준이었다. 그리고 같은 해 10 월에 첫 공식버전 0.02 가 발표되었고 이는 bash(GNU Bourne Again Shell)와 gcc(GNU C compiler)정도가 실행될 수 있는 수준이었다.

리눅스의 첫 번째 버전인 0.01 은 1991 년 9 월 17 일 인터넷을 통해 공개되었고, 첫 공식 버전인 0.02 는 같은 해 10 월에 발표되었다. 그 이후 지금까지, 전 세계 수천만의 개발자들이 리눅스 개발에 자발적으로 참여하고 있다. 초창기 리눅스는 설치와 부팅을 하기 위해서는, 미닉스와 같은 다른 운영 체제가 필요했다. 그러나 리로(lilo)와 같은 부트로더가 개발되고, GNU 프로젝트가 만들어낸 모든 유틸리티를 리눅스에서 사용할 수 있게 됨에 따라, 리눅스는 빠른 속도로 미닉스를 능가하게 되었다. 리눅스는 거의 공개 직후부터 엄청난 성장세를 보이는데 그 이유는 여러가지 원인들이 겹쳤기 때문이다. 그 중 하나는 BSD 가 소송에 휘말리면서 대체품에 대한 관심이 커졌기 때문인데 이는 뒤에서 자세히 설명할 것이다. 또한, GPL 라이센스를 선택하고 현재 오픈소스 개발 모델의 모체가 된, 소스 공개와 공개적 개발 모델을 선택한 것도 매우 유효했다. 이전에는 오픈 소스 소프트웨어라도 핵심 개발자 집단이 개발 과정을 독점하고, 배포할 준비가 되었다고 판단 될 때에야 소스를 공개하는 방식을 사용했다. 그런데 리눅스는 누구나 소스를 읽은 후

패치를 작성해 보낼 수 있도록 했다. 패치가 받아들여지면 그 패치를 보낸 사람은 기여자가 되어 다음 릴리스 노트에 이름이 박혔다. 이는 사람들의 과시욕을 충족시키는 방식이었고, 사람들은 돈을 받지 않고도 커널 버그 수정과 기능 추가에 매달렸다. 초기 리눅스는 기능이 불완전한 운영체제였지만 자체 커널 개발에 난항을 겪고 있던 GNU 프로젝트가 리눅스 커널에 관심을 가졌고, 리눅스 커널과 GNU 유틸리티가 결합하면서 비교적 완전한 운영체제로 거듭났다. 그 후 Sun, IBM 과 같은 대기업들이 리눅스 개발을 지원하기 시작하는 등, 리눅스가 IT 세계에서 가지는 입지는 빠르게 탄탄해졌다. 현재는 상당수의 웹 서버와 모바일 장치를 구동하는 운영체제이다. 특히 서버 쪽에서 리눅스는 엄청난 점유율을 보여주는데 그 이유는 무료이고, 리눅스에 익숙한 인력도 많고, 안정성도 높은 편이라 서버 운영체제로서는 최적이기 때문이다. 또한 모바일 시장에서 구글 안드로이드 운영체제와 국내의 가정용 공유기의 대부분이 리눅스를 기반으로 한다. 안드로이드에 대해서는 뒤에서 자세하게 이야기 할 예정이다.

USL 대 BSDi 는 1992 년 유닉스 시스템 연구소 (Unix System Laboratories)가 버클리 소프트웨어 디자인 (Berkeley Software Design, Inc)과 유닉스 운영체제 관련 지적 재산권에 관한 캘리포니아 리전트 (Leents of California)의 소송이었다. 소송은 판사가 USL 의 지적 재산권의 유효성에 의문을 제기

한 후 1993 년 법정에서 해결되었으며 Novell (당시 USL 을 구입 한 사람)과 BSDi 는 Berkeley Software Distribution (BSD)에 대해 더 이상의 소송을 제기하지 않기로 동의했다. 나중에 BSD 배포본의 범위로 발전 할 것이고, 각각 BSD 배포판은 독자적인 특정 잠재 고객의 강점과 시장에 맞게 조정 될 것 이다. 이 소송은 AT & T 의 벨 연구소 (Bell Labs)에서 유닉스 소스 코드에 대한 라이센스를 가진 버클리 캘리포니아 대학 (University of California, Berkeley)의 컴퓨터 시스템 리서치 그룹 (CSRG)에 뿌리를 두고 있다. CSRG 에서 운영 체제 연구를 수행하는 학생은 유닉스를 수정하고 확장했으며, CSRG 는 1978 년부터 AT & T 의 축복으로 수정 된 운영 체제를 여러 번 발표했습니다. 이 버클리 소프트웨어 배포 (BSD)에는 저작권으로 보호 된 AT & T Unix 소스 코드가 들어 있기 때문에 AT & T 의 Unix 용 소스 코드 라이센스가 있는 조직에서만 사용할 수 있었다. CSRG 의 학생과 교수진은 TCP / IP 스택의 소프트웨어 코드를 감사하고 모든 AT & T 지적 재산을 제거하고 1988 년 BSD 라이센스에 따라 "Net / 1"로 일반 대중에게 공개했다. 버클리 CSRG 가 곧 마감 될 것이라는 것이 분명 해지자 CSRG 의 학생과 교수진은 남은 AT & T 코드를 BSD 에서 제거하고 자신의 코드로 대체하기위한 노력을 시작했고 이 노력으로 1991 년 Net / 2 가 BSD 라이센스하에 공개되었다.

Berkeley Software Design (BSDi)은 Net / 2 의 소스를 확보하여 누락 된 부분을 채우고 Intel i386 컴퓨터 아키텍처에 이식했다. 그런 다음 BSDi 는 BSD / 386 운영 체제를 판매하였고 이는 BSD / 386 에 AT & T 지적 재산권이 없다는 BSDi 의 주장에 동의하지 않는 AT & T 의 분노를 불러 일으켰다. AT & T 의 유닉스 시스템 연구소 (Unix System Laboratories) 자회사는 1992 년 4 월 뉴저지 주 BSDi 에 소송을 제기했다.이 소송은 추후에 캘리포니아 대학의 리전트 (Regents of the University of California)를 포함하도록 개정되었다. 수년 동안 USL 과 BSDi 가 나중에 무료 BSD 로 개발 될 소프트웨어에 대해 더 이상 소송을 제기하지 않기로 합의했다는 일반 대중의 공감하에 정착의 세부 사항이 당사자들 사이에서 비밀로 유지되었다. 2004 년 11 월 USL 대 BSDi 합의 합의서 사본이 Groclaw 웹 사이트에 게시되었으며, 캘리포니아 주 공공 기록법에 의거 한 캘리포니아 주립 대학교 총장실 (The Regents of the Counsel)에서 가져 왔고 이는 유닉스 법률 역사상의 중요한 연결 고리가 되었다. 합의서의 가장 두드러진 점은 다음과 같다.

- 4.4BSD-lite 는 분쟁이없는 파일을 포함하여 배포된다다. 대학이 Net / 2 로부터 전환 할 것을 장려한다.
- 대학은 특정 파일의 배포를 중단한다.

- USL 은 분쟁중인 파일 사용자에게 3 개월의 유예 기간을 부여했다.
- USL 이 배포 한 특정 파일은 대학 저작권 고지를 전송한다.
- USL 은 특정 파일의 무료 배포를 허용한다.
- 대학은 특정 파일에 대한 USL 의 권리에 대한 법적 시도를 적극적으로 지원하지 않는다.

오픈 소스 이니셔티브 (OSI)는 오픈 소스 소프트웨어를 홍보하는 데 전념하는 비영리 단체이다. 이 조직은 1998 년 2 월 Bruce Perens 와 Netscape Communications Corporation 에서 영감을 얻은 그룹의 일원 인 Eric S. Raymond 가 설립하여 Netscape Communicator 제품의 주요 소스 코드를 게시했다. Raymond 는 2005 년 2 월까지 창립 당시의 회장이었고, Russ Nelson 과 Michael Tiemann 이 간략히 그 뒤를 이었다. 2012 년 5 월에 새 이사회가 Simon Phipps 를 회장으로 선출했으며 2015 년 5 월 Allison Randal 이 이사회 임기 마지막인 2016 년을 준비하기 위해 Phipps 가 사임 할 때 대통령으로 선출되었다.

또한 자유 소프트웨어 운동과 OSI 의 관계를 살펴보면 현대 자유 소프트웨어 운동 (1980 년대 초 리차드 스톨만이 출시)과 오픈 소스 이니셔티브는 유닉스와 인터넷 자유 소프트웨어, 해커 문화의 공통된 역사에서 태어났지만 근본적인 목표와 철학은 다른데 오픈 소스 이니셔티브 (Open

Source Initiative)는 창립 멤버 마이클 티만 (Michael Tiemann)의 말에서 "자유 소프트웨어"와 관련된 도덕적 대립 태도를 버리고 "실용적인 비즈니스 케이스"에 대한 오픈 소스 아이디어를 홍보하기 위해 "오픈 소스"라는 용어를 선택했다. 1999 년 초에 OSI 의 공동 설립자 인 Perens 는 서로 다른 접근 방식 때문에 Stallman 's Free Software Foundation (FSF) 지지자와 OSI 사이에서 발전하고 있던 "분열"에 반대했다. Stallman 은 OSI 의 실용적인 초점과 중앙의 "윤리적 인 명령"을 고려한 것을 무시한 것에 대해 OSI 를 예리하게 비난했고 그가 자유 소프트웨어를 정의했기 때문에 자유 소프트웨어의 근원 인 "자유"에 중점을 두고 있습니다. 그럼에도 불구하고 스톨만은 자신의 자유 소프트웨어 운동과 오픈 소스 이니셔티브를 동일한 자유 소프트웨어 공동체 내에서 별도의 캠프로 묘사했으며 철학적 인 차이에도 불구하고 오픈 소스와 자유 소프트웨어의 지지자는 종종 "실제 프로젝트에서 함께 작동한다"고 인정했다.

Git 은 컴퓨터 파일의 변경 내용을 추적하고 여러 사람이 사용하는 파일에 대한 작업을 조정하는 버전 제어 시스템이다. 주로 소프트웨어 개발의 소스 코드 관리에 사용되지만, 모든 파일 세트의 변경 내용을 추적하는 데 사용할 수 있는데, 분산 개정 관리 시스템으로서 속도, 데이터 무결성 및 분산 된 비선형 워크 플로우에 대한 지원을 목표로 한다.

Git 은 2005 년 Linux 커널 개발을 위해 Linus Torvalds 에 의해 만들어졌으며 다른 커널 개발자는 초기 개발에 기여했다. 2005 년 이래로 현재 유지 관리자는 Junio Hamano 이다. 대부분의 다른 분산 버전 제어 시스템과 마찬가지로 대부분의 클라이언트 - 서버 시스템과 달리 모든 컴퓨터의 모든 Git 디렉토리는 네트워크 액세스 또는 중앙 서버와 독립적으로 전체 기록 및 전체 버전 추적 기능을 갖춘 본격적인 저장소이다. 또한 Git 은 GNU General Public License 버전 2 의 조건에 따라 배포되는 무료 소프트웨어이다.

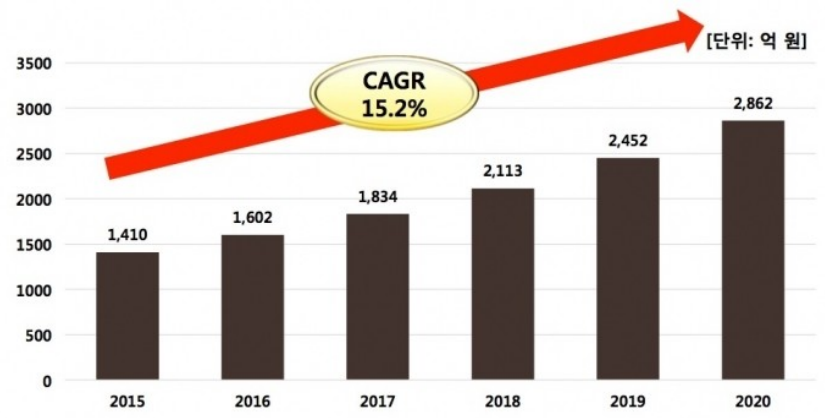
GitHub 는 분산 버전 관리 툴인 Git 을 사용하는 프로젝트를 지원하는 웹호스팅 서비스이다. 루비 온 레일스로 작성되었고 영리적인 서비스와 오픈소스를 위한 무상서비스를 모두 제공한다. 2009 년 Git 사용자 조사에 따르면 GitHub 는 가장 인기있는 Git 호스팅 사이트이고 2011 년 조사에서는 가장 인기있는 오픈소스 코드 저장소로 꼽혔다. Git 은 텍스트를 입력하여 명령을 전달하는 반면에 GitHub 는 GUI 를 통해 명령을 전달한다는 점이 가장 큰 차이점이다.

안드로이드(Android)는 리눅스 커널을 기반으로 구글에서 제작한 스마트폰과 같은 모바일 플랫폼의 운영체제와 미들웨어 및 중요 애플리케이션이 포함된 소프트웨어 집합니다. 구글은 안드로이드 OS 에 대해 리눅스 커널의

제너럴 퍼블릭 라이선스(GPL)를 따르고 있으며, 새로운 운영체제의 버전 공개와 동시에 소스를 공개하고 있다. 2017 년 현재 안드로이드는 세계에서 가장 대표적인 오픈소스 플랫폼이며, 세계 최다 사용자를 보유한 운영체제이다. 또한 안드로이드는 리눅스 커널 위에서 동작하며, 다양한 안드로이드 시스템 구성요소에서 사용되는 C/C++ 라이브러리들을 포함하고 있다. 안드로이드는 기존의 자바 가상 머신과는 다른 가상 머신인 달빅 가상 머신을 통해 자바로 작성된 응용 프로그램을 별도의 프로세스에서 실행하는 구조로 되어 있다.

2005 년 7 월에 구글의 인수 이후 2007 년 11 월 5 일에 결성된 오픈 핸드셋 얼라이언스(OHA)의 목표인 모바일 기기의 공개 표준의 결과물로써 발표되었다. 리눅스 커널 2.6 에서 빌드 되었으며 안드로이드는 1 년 후 2008 년 10 월 21 에 오픈소스로 선언되었다. 구글은 네트워크와 텔레폰 스택을 포함하는 완전한 소스 코드를 아파치 라이선스로 공개하였다. 그러나 현재 API 만 완전한 공개소스인 상태이며 VM 에 대한 소스는 공개하지 않고 있다.

마치며



◇ 오픈소스 SW 시장 규모 및 전망 [자료=정보통신산업진흥원]

필자들 또한 대학교에서 학문을 공부하고있는 대학생들이며 오픈소스에 대해 거의 문외한인 사람들이었다.
아마 이 소책자를 읽기 전 대부분의 사람들과 비슷한 사람들이라고 자부한다.

하지만 이 안의 내용 이외에도 오픈소스 소프트웨어는 우리가 생각하는 상상 이상의 놀라움을 보여주고있다. 당장의 2017 년 국내 오픈소스 소프트웨어 시장 전망을 보더라도 내년도 국내 오픈소스 소프트웨어 시장 규모가 올해보다 14.5% 증가한 1 천 834 억원에 달할 전망이며 매년 15.2%씩 성장해 2020 년에는 2 천 862 억 원에 이를 것으로 예상되고있다. 29 일 정보통신산업진흥원의 오픈소스

소프트웨어 시장 및 트렌드 조사에 따르면 2016 년 오픈소스 소프트웨어 시장 규모는 전년보다 13.6% 오른 1 천 602 억 원을 기록할 것으로 예측되고 있다.
우리들 또한 이 소책자를 작성하면서 오픈소스의 중요성, 이미 널리 쓰이고 있는 광범위함과 밝은 미래성을 보고 경악을 금치못하였다.
이 책자를 읽고 오픈소스 소프트웨어에 대해 어느정도 관심을 가지게 되었기를 바라며 혹여나 가지게 되었다면 오픈소스 소프트웨어에 대해 더 깊이 조사해보며 얼마나 광활하게 널리 퍼져있는지를 몸소 느끼기를 바란다.