**1. 오픈소스**

**1). 오픈소스란 무엇인가?**

오픈소스 공식 홈페이지에서 정의하는 '오픈소스'는 소스 코드가 공개되어 있고 자유로운 이용과 개발이 인정되고 있다는 것을 의미한다. 따라서 오픈소스 소프트웨어라하면 오픈소스 형식을 택한 소프트웨어를 말한다. 오픈소스로 인해 소스코드가 공개되어 사람들은 쉽게 접근할 수 있게 되었다. 하지만, 이러한 부분으로 인해 가끔 오픈소스를 '공짜 소프트웨어'라 인식하기도 한다. 물론 소스코드를 무료로 이용할 수는 있지만 오픈소스를 장려하는 단체인 OSI(Open Source Initiative)에 따르면 오픈소스를 무료 소프트웨어인 프리웨어와 동일선상에 봐서는 안된다는 입장이다. 오픈소스 프로그래머는 그들이 만든 오픈소스 소프트웨어에 돈을 청구할 수 있다. 그러나 많은 경우에서 오픈소스 라이선스를 따르고, 오픈소스 라이선스가 소스코드를 공개하도록 요구하기 때문에 프로그래머들은 소프트웨어 자체를 유료화하기 보다 소프트웨어 서비스를 지원 하는 것이 수익이 있다고 본다. 이러한 방식으로 인해 오픈소스 소프트웨어는 무료로 남아있는 것이고 개발자들은 그 소프트웨어를 설치,사용,오류해결해주는 것을 도와줌으로써 돈을 번다. 즉, 오픈소스 소프트웨어의 '무료'는 부산물과 같은 개념이다.

**(1) 다른 소프트웨어와 오픈소스 소프트웨어의 차이점은 무엇일까?**

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

**(2) OSI 의 '오픈소스의 정의'**

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

1. 자유로운 재배포
2. 원시 코드
3. 파생 저작물 허용
4. 저작자의 원시 코드 원형 유지
5. 개인 및 단체에 대한 차별 금지
6. 사용 분야에 대한 차별 금지
7. 사용 허가의 배포
8. 특정 제품에만 유용한 사용 허가의 금지
9. 다른 소프트웨어를 제한하는 사용 허가의 금지

**2. 자유소프트웨어**

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

1. **자유소프트웨어란**

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

Thus, “free software” is a matter of liberty, not price. To understand the concept, you should think of “free” as in “free speech,” not as in “free beer”.

(그러므로, '프리소프트웨어'는 자유의 문제이지 가격의 문제는 아닙니다. 이 개념을 이해하려면 당신은 '자유'를 '공짜맥주'가 아닌, '언론의 자유'와 같은 예를 생각해보면 됩니다.)

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

##### **(1) 자유소프트웨어의 4가지 '자유'**

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

자유 0 ) 프로그램을 어떠한 목적을 위해서도 실행할 수 있는 자유.

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

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

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

##### **(2) 자유소프트웨어와 오픈소스의 차이점**

자유소프트웨어와 오픈소스에 대해 간략히 알아보았다. 하지만, 이 둘의 차이점은 아직까지 애매모호할 수 있다. OSI에 이 주제와 관련된 글이 실릴 정도로 많은 사람들이 오픈소스와 자유소프트웨어를 잘 구분하지 못한다. OSI에 따르면, 자유소프트웨어와 오픈소스가 같은 라이선스를 언급하고 있어도 그 둘은 다른 뿌리에서 왔다는 입장이다. 자유소프트웨어의 개념은 1980년대부터 시작되었으며 소프트웨어를 변형하고 복사하고 분배하고 실행할 자유 그 자체에 집중을 한다. 하지만, 오픈소스의 개념은 1997년도 부터 시작되었다. 자유소프트웨어 운동을 이끌었던 리처드 스톨만 역시, 오픈소스는 개발 방법론인데 반해, 자유소프트웨어는 사회적인 운동이라고 말한 바가 있다.

**3. 오픈소스 라이선스**

**1) 카피레프트**

카피레프트는 카피라이트의 개념을 전제로 한다. 카피라이트는 원작자는 복사물 혹은 행위에 대한 제한적인 독점권을 가짐을 의미한다. 카피라이트는 작업하는 즉시 부여되며 어떠한 등록도 요구되지 않는다. 자연스럽게 모든 권리가 원작자에게 주어지는 것이다. 그 말은 즉슨, 아무도 원작자의 허가없이 다시 재배포하거나 수정할 수 없음을 말한다. 그리고 사용이 가능하다는 원작자의 허가를 라이선스라고 하며 어떤 명확한 조건이 붙어서 나온다.

카피레프트는 원작자의 소프트웨어 또는 문서를 수정할 수 있고 다시 배포할 수 있는 권한을 말한다. 카피레프트가 도입된 배경은 다음과 같다. 당시 독점소프트웨어는 '카피라이트'를 적용하여 누구도 코드를 변형할 수 없음을 명시해 두었다. 자유소프트웨어 재단의 리처드 스톨만은 소스코드가 공개된 소프트웨어를 카피라이트를 적용하지 않고 퍼블릭 도메인에 둔다면, 다른 누군가가 약간의 수정을 한 뒤에 그것을 독점 소프트웨어로 만들 수 있을 것이라 보았다. 이는 자유소프트웨어가 지향하는 바와는 전혀 맞지 않는 것이었고,리처드 스톨만은 '라이트'와 반대되는 의미인 '카피레프트'를 생각해낸다. 리처드 스톨만이 말하는 카피레프트는 다음과 같다.

'어떤 소프트웨어의 저작권이 보호받고, 저작자는 재배포를 허용하며, 수정권한을 주지만 재배포할시에는 똑같이 카피레프트를 적용해야한다.'

앞서 말했듯이 카피레프트는 카피라이트의 개념을 전제로 한다. 흔히들 카피레프트가 저작권을 반대한다고 볼 수 있으나 이는 잘못된 해석이다. 쉽게 말해서, 저작권(카피라이트)를 다른사람에게도 부여하여 수정과 재배포 또한 자유롭게 할 수있음을 의미한다고 보면 된다.

**2) 오픈소스 라이선스**

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

**(1)오픈소스 라이선스 종류**

**GNU 일반 공중 사용 허가서(GNU GPL 혹은 GPL)**

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

**LGPL 라이선스**

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

**아파치(Apache) 라이선스**

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

##### **모질라(Mozilla) 퍼블릭 라이선스(MPL)**

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

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

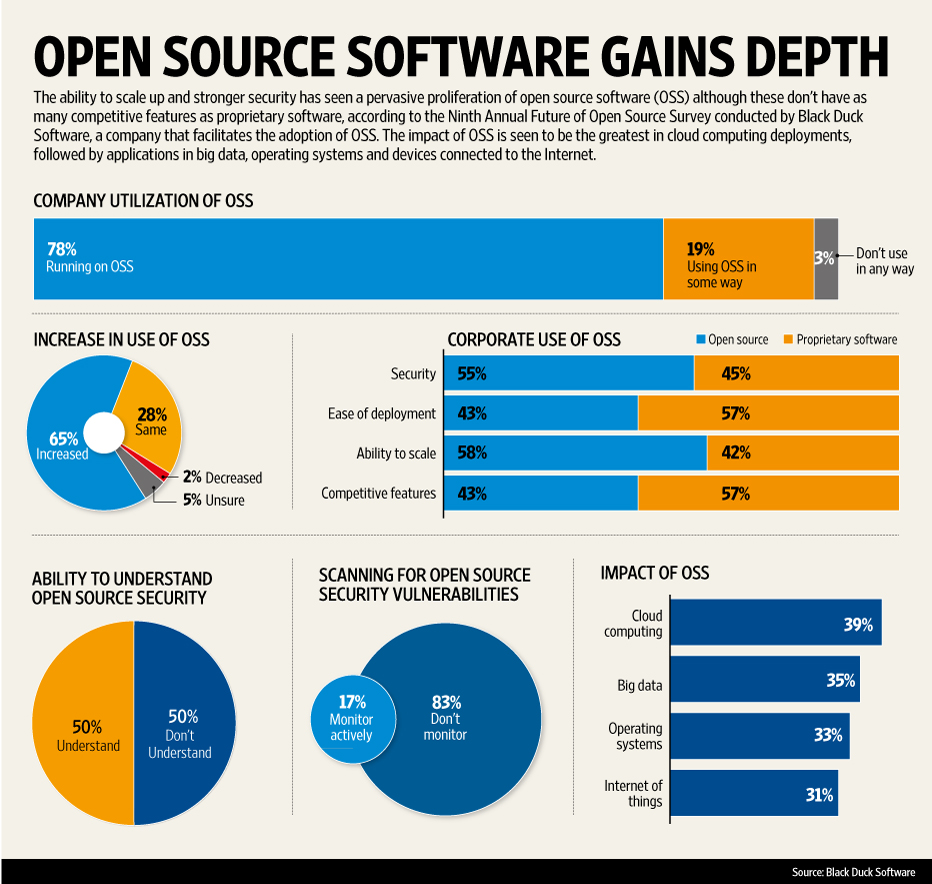
**다양한 오픈소스 라이선스들**

* Eclipse - **이클립스 공용 허가서**(Eclipse Public License, EPL).
* OpenJDK - GPL v2 (GNU GPL 버전 2)
* PostgreSQL - **PostgreSQL License**
* Docker - Apache License 2.0

**(2)오픈소스 라이선스 비교**



**(3)가장 인기있는 오픈소스 라이선스**



2017년 가장 많이 사용하는 오픈소스 라이선스의 순위를 매긴 것이다. BSD라이선스를 기초로 MIT 대학에서 만든 라이선스인 MIT가 1등을 차지하였다. MIT 라이선스를 따르는 소프트웨어를 개조한 소프트웨어를 반드시 오픈 소스로 배포해야 한다는 규정이 없으며 GNU 일반 공중 라이선스의 엄격한 제약 피하려는 사용자들에게 인기가 있다고 한다.

**4. 오픈소스 프로젝트**

대부분 오픈소스 프로젝트 구성원은 다음과 같다.

**컨트리뷰터**: 컨트리뷰션을 하는 모든 사람들을 말한다.

**커미터**:컨트리뷰션의 내용을 리뷰하고 프로젝트에 반영할지 결정하는 사람. 커미터가 없는 경우도 있다.

**메인테이너**:프로젝트 방향 설정, 관리하는 사람. 보통 커미터 중 일부가 맡으며, 없는 경우도 있다.

**저작자**:프로젝트 만든 사람 또는 조직을 말한다.

1. 오픈소스 프로젝트 활동
2. **컨트리뷰션**

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

오픈소스 프로젝트를 진행하기 위한 개발 도구들을 간단하게 살펴보자.

1. **버그 트래커**

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

1. **버전 컨트롤**

버전 관리(version control, revision control), 소스 관리(source control), 소스 코드 관리(source code management, SCM)란 동일한 정보에 대한 여러 버전을 관리하는 것을 말한다. "버전"을 통해서 시간적으로 변경 사항과 그 변경 사항을 작성한 작업자를 추적할 수 있다. 간단한 버전 관리 방법으로는 처음 작성한 코드에 버전 번호 1을 부여한다. 변경 사항이 생기면, 버전 번호를 2로 증가 시킨다. 이처럼 추후 변경 사항이 발생 시마다 버전 번호를 1씩 증가시킨다. 대표적인 버전관리 시스템으로는 'GIT', 'Subversion'등이 있다.