**오픈소스 소프트웨어 설계 및 실습 과제 2**

**2018202074 김상우**

**오픈소스 바르게 사용하기**

**-우리가 아는 오픈소스**

-1991.08.26 Linus Torvalds가 UNIX교육용 버전인 MINIX의 메일링 그룹에 공개(Linux의 시작)

Linux를 기반으로 Fedora, Ubuntu, CentOS 등의 Linux 배포판 파생.(Android도 Linux 커널 기반) Chrominum OS도 Linux에서 시작임을 GNU/Linux Distribution Timeline에서 참고 가능

오픈소스 종류

(좌측부터 Chrominum, Bootstrap, Tomcat)

Chrominum: 구글 브라우저로 Chrome, 웨일(네이버)의 기반 Bootstrap: 트위터가 공개한 웹 프런트엔드 플랫폼(반응형 웹페이지 개발용) Tomcat: 웹 개발 환경에 사용되는 웹 애플리케이션 서버

**-오픈소스와 저작권**

오픈소스에서는 저작권자가 소스코드의 사용 범위와 사용자의 의무사항을 라이선스에 명시. 이를 무시할 시 저작권 침해할 가능성이 있으므로 라이선스 확인 및 의무 조항 준수 필요.

위 조건을 지키는 경우에만 이를 무료로 사용가능하다. 그 외 오픈소스 프로젝트 중에는 유로로 사용할 수 있게 하는 듀얼라이선스 적용 혹은 기능을 제한하여 과금을 유도하는 프로젝트도 존재.

많은 오픈소스가 시작부분에 주석으로 저작권과 라이선스 정보 명시(정보 표기가 의무 조항) 소스코드 유지 보수 및 관리를 위해서도 출처 명시 필수. 이는 파일 단위, 함수 단위에도 해당 된다.

**-오픈소스 라이선스**

오픈소스 라이선스:오픈소스로 배포된 소스코드를 사용 시 지켜야 할 규약 등을 명시한 것. 이 조항을 지키는 것이 해당 소스코드 사용 가능이라는 허가와 같음. 이를 이행하지 않을 시 저작권 침해에 해당할 수도 있음. (오픈소스 소프트웨어 라이선스 가이드 3.0 참고) 일반적인 요구사항은 아래와 같다.

o소스 코드를 수정한 정보의 표시 o라이선스 정보의 제공 o동일한 라이선스로 재배포 o소스 코드의 제공

주요 오픈소스 라이선스 특징들은 다음과 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 무료이용가능 | 배포허용가능 | 소스코드  취득가능 | 소스코드  수정가능 | 2차적 저작물  제공개 의무 | 독점 SW와  결합 가능 |
| MIT License | O | O | O | O | X | O |
| BSD 2-Clause  BSD 3-Clause | O | O | O | O | X | O |
| Apache  License 2.0 | O | O | O | O | X | O |
| GPLv2  GPLv3 | O | O | O | O | O | X |
| LGPLv2 | O | O | O | O | O | O |
| MPL | O | O | O | O | O | O |

**-오픈소스 라이선스 확인과 준수**

라이선스는 일반적으로 소스 코드 저장소 최상위 디렉터리에 존재하는 LICENSE문서 혹은 README 문서에서 확인가능. 그 외의 경우는 다음과 같다.

-별도의 홈페이지 존재:LICENSE 메뉴, 소프트웨어 설명 내 표기, 소스코드 내 COPYING 문서, README 문서, 소스코드 내 주석 등에서 확인 가능

-GitHub 저장소가 존재:README문서,소스코드 내 COPYING 문서,LICENSE 문서,소스 코드 내 주석

-검색으로 찾은 소스코드: 구글 및 Stack Overflow가 대표적인 예, 위 경우에는 소스코드 내 주석, 개발자의 답변에서 확인

의무 사항을 준수하는 방법은 아래와 같다.

애플리케이션으로 배포하는 경우: 애플리케이션 설정 메뉴 하위에 법적 공지나 오픈소스 메뉴를 두어 저작권과 라이선스 정보 표기 가능

오픈소스로 재배포하는 경우: 새로운 NOTICE 문서나 서드파티 관련 메뉴에서 저작권과 라이선스 정보 표기 가능

**-오픈소스를 사용하다 보면**

버그 발생시, 자신의 소스코드만 수정시 오픈소스의 다음 버전에선 버그가 수정되지 않으므로 발견된 버그와 직접 수정한 소스 코드가 원래 오픈소스에 적용될 수 있도록 저작권자에게 연락하는 것도 가능하다.

버그 수정 및 불편한 사항, 새로운 의견 전달은 컨트리 뷰션의 시작이 될 수 있다.

**컨트리뷰션 시작하기**

**-컨트리뷰션의 유형**

컨트리 뷰션의 30% 이상이 소스 코드 수정 이외의 영역에서 발생 (More Common Than you Think: An In-Depth Study of Casual Contributors 참고)

오타 수정, 번역, 가이드 문서 작성, 디자인 작업, 의견제시, 프로젝트 스타일 가이드 작성, 중복 이슈 정리 등이 컨트리 뷰션에 속한다.

**-컨트리뷰션의 활동 이유**

불편한 점, 버그 개선 시 개인 사용시엔 오픈소스 버전업 마다 추가 패치 작업 필요. 오픈소스에 참여하는 개발자들의 멘토링이 가능(개발, 영어 실력 향상) 본인 역량 노출 및 본인 홍보의 수단(이력서 사용 및 인턴 활동) 많은 테스트 및 수준 높은 코드 품질 제공, 결과적으로 코딩 능력 향상 및 시야 확장

**-컨트리뷰션 전에 알아두기**

오픈소스 프로젝트 구성원의 역할은 다음과 같다.

사용자: 프로젝트를 사용하는 사암, 컨트리뷰터: 프로젝트에 컨트리뷰션 활동을 하는 모든 사람, 저작자: 프로젝트를 만든 사람 혹은 조직 커미터: 다른 사람의 컨트리뷰션 내용을 리뷰하고 프로젝트에 반영할 권한을 가진 컨트리뷰터. 메인테이너: 프로젝트의 방향을 설정하고 프로젝트를 관리하는 책임이 있는 컨트리뷰터. 커미터 중 일부가 메인테이너가 된다.(커미터와 메인테이너가 없는 프로젝트도 있다.)

오픈소스 프로젝트의 기본 문서

README 문서: 프로젝트 목적, 사용 방법 명시 LICENSE 문서: 오픈소스 라이선스를 명시 CONTRIBUTING 문서: 프로젝트에 어떻게 컨트리뷰션 활동을 할 수 있는지 설명한 문서 , 어떤 유형의 컨트리 뷰션이 요구되고, 해당 프로젝트의 컨트리뷰션 절차가 어떤지 파악 가능, 이 문서는 없기도 하지만 존재시 다른사람들의 컨트리 뷰션을 환영함을 알 수 있다.

이 외에도 코딩 컨벤션이나 이슈 템플릿 등의 문서가 존재.

오픈소스 프로젝트의 구성원은 이슈를 논의하기 위해 이슈트래커 혹은 Slack이나 IRC등의 실시간 채팅 채널을 이용하기도 하며, 기능 제안 및 코드 공유를 위해 GitHub의 Pull requests, 메일링 리스트를 사용하기도 한다.

오픈소스를 찾을 떈 다음과 같은 사이트를 참고할 수 있다.

GitHub의 Explore 메뉴: 유형별 오픈소스, 최신 경향의 오픈소스 확인 가능 Your First PR: 첫 컨트리뷰션으로 시도할 만한 이슈와 도움이 필요한 오픈소스 확인 가능 CodeTriage: 처리할 이슈가 있는 오픈소스를 관심 언어 따라 확인 가능

컨트리뷰션 활동을 할 오픈소스 선택 시 다음을 확인하는 것이 좋다.

-오픈소스 라이선스 하에 배포되었는가 –커뮤니티가 적극적으로 컨트리뷰션을 수용하는가 –커뮤니티가 새로운 사람들에게 우호적인가

**-컨트리뷰션 시 주의 사항**

우선 작업하려는 내용이 이미 진행된 적인 있는 지 확인해야 한다. README문서, 이슈목록, 메일링 리스트, Stack Overflow등을 확인하면 된다. 이슈에 의견을 다는 방법도 있다. 만약 수정하고자 하는 이슈가 변화가 오랫동안 없을 시 작업 시작 전 현재 상황 문의가 필요하다.

규모가 큰 중요기능 개발 시 개발 시작 전에 공유하는 것이 좋다.(프로젝트 진행 방향 및 개방 진행 상태 고려)

작업한 결과물을 프로젝트에 적용해 달라고 요청 시 프로젝트의 컨트리뷰션 가이드에 따라 요청해야 한다.(준수해야 하는 코딩 컨벤션이 존재하거나 사전 테스트가 필요할 수 있으므로)

커뮤니티에서 커뮤니케이션 할 때는 겸손한 자세와 명확한 의사 전달이 필요.

**-컨트리뷰션 이후**

응답을 받지 못한 경우 이슈의 재생성 혹은 의견 생성을 통해 요청이 가능하다. 프로젝트를 fork해 수정 사항을 직접 반영하는 방법도 존재한다.

내용 개선 및 수정 요청을 받을 시 빠른 답변이 필요하다. 어떻게 변경하는 지 모를 시 커뮤니티에 도움 요청이 가능하다.

거절 당할 경우, 거절 이유가 납득되지 않는다면 거절에 대한 피드백과 추가 설명 문의 가능. 단 커뮤니티의 결정을 존중해야 한다. 이 경우에도 fork를 통해 새로운 프로젝트를 생성하는 것도 가능할 것이다.

**-오픈소스 프로젝트 공개하기(**"[Starting an Open Source Project](https://opensource.guide/starting-a-project/)"와 "[The Legal Side of Open Source](https://opensource.guide/legal/)" 기반**)**

**-오픈소스로 공개하는 이유**

원 오픈소스의 새로운 버전 시작 및 중단된 운영의 지속: 컨트리뷰션이 거절될 경우 folk를 통해 새로운 기능 추가 및 별도의 프로젝트 운영 가능, 혹은 중단된 프로젝트를 포크해 운영이 지속될 수도 있다.(C3프로젝트 -> 네이버의 billboard.js프로젝트)

오픈소스 라이선스 의무 수행: GPL, AGPL, MPL 등의 라이선스로 배포되는 오픈소스를 사용하면 라이선스 조항에 따라 산출물로 나온 소프트웨어의 소스코드를 공개해야 할 때가 있다. (Xiaomi kernel 프로젝트 : GPL 의무조항)

소프트웨어의 성장: 외부 개발자와의 문제 공유, 해결 및 다양한 테스트 환경, 오류 신고 등으로 소프트웨어 품질 상승. (Ex: Pinpoint(대규모 분산 시스템용 APM))

사회 공헌:일반적으로 서비스나 제품에 필요한 소프트웨어 개발 시 오픈소스 기술 필요. 얻은 혜택을 생태계에 다시 돌려주기 위해 오픈소스 운영

**-오픈소스 공개를 위한 준비**

ㅇ완성도와 관계없이 README 문서와 LICENSE 문서가 필요., 사용자의 접근과 이해를 도와주고 컨트리뷰션을 유연하게 만듬. (GitHub 사용시 이 문서들을 최상위 디렉터리로)

ㅇ라이선스 선택: 사용 조건, 범위 명시. 오픈소스 프로젝트 공개시 필수. 표준 라이선스가 존재.

ㅇ프로젝트 의존성 확인: 프로젝트에 포함된 다른 오픈소스 프로젝트 들의 라이선스 확인 필요. 모든 라이선스가 perssive 라이선스일시: 제약없이 사용가능 (MIT, Apache, ISC, BSD 등)

하나라도 copyleft라이선스 일시: 승계의무 준수를 위해 동일 라이선스로 재배포. (GNU GPLv2, GNU GPLv3, AGPLv3)

ㅇ적합한 라이선스 선택: 커뮤니티 선택을 고려. 다른 프로젝트 사용을 원할 시: 커뮤니티 선호 혹은 유사 오픈소스 다량 사용 라이선스를 선택. 기업 사용: 특허 내용 포함 라이선스 채택.(Apache 2.0) 기여내용 지속 공개: GNU GPLv3, GNU AGPLv3 고려 사항 없을시 MIT 라이선스(간결함)

ㅇREADME 문서 작성 : 프로젝트 목적 및 용도, 프로젝트 시작 방법, 저작권과 라이선스 정보, 외부 리소스 정보 포함. 추가적으로 컨트리뷰션 진행 방법, 버그 신고 방법, 사용자질문 등록 방법 포함도 가능. 전세계를 대상으로 하는 경우 외국어로 작업하기도 함

ㅇ프로젝트 명 지정 : 기존 오픈소스의 이름을 명시하는 등의 방법으로 명확한 이름을 선택, 이 때 충돌하는 이름을 피해야 함.(기존 제품 상표 침해의 가능성)

**-공개 전 마지막 확인**

문서화: README문서 작성, LICENSE문서 작성, 프로젝트 명의 적절성 확인

코드: 목적과 예외 처리 설명 주석 확인, 코딩 컨벤션에 따라 함수, 메서드, 변수 명 사용. 수정 작업 목록 및 이슈, Pull requests 등에 개인 암호나 공개되면 안되는 정보가 없음. 외부 출처의 오픈소스 프로젝트 코드에 대한 출처, 저작권, 라이선스 정보를 주석으로 명시

개인프로젝트 일 경우 어딘가에 채용되어 있을 시: 회사의 오픈소스 관련 팀 및 지적재산권 관련팀, 법무팀에 프로젝트 공유 필요.

업무로 공개하는 프로젝트의 경우: 사내 오픈소스 관련 팀 및 법무팀에 프로젝트 공유 확인, 커뮤니티 관리 책임 담당자 확인, 2명 이상의 담당자가 프로젝트 관리자 권한 소유

**-주요 오픈소스 SW라이선스 활용 사례**

iOS SDK(Software Developmen Kit)



애플의 앱개발 프로그램이다. 이를 사용시 아이폰 운영체제인 iOS에서 구동되는 앱 개발이 가능하며, 테스트를 비롯 앱스토어 등록/판매도 가능하다는 장점이 존재한다. 이를 통해 개인도 비교적 손쉽게 앱스토어에 등록할 수 있는 앱을 만들고 등록할 수 있는 환경을 애플은 제공하였다. 앱을 애플 기기에 업로드하기 위한 iOS 개발자 프로그램은 추가적으로 구입해야 하나 기본은 무료 오픈소스이다. .

텔레그램 오픈소스



텔레그램은 모바일메신져의 일종으로, 아이폰/안드로이드 텔레그램 앱 소스코드와 API, 프로토콜 등을 무료로 공개, 누구나 보완과 재생산이 가능하게 하였다. 결과적으로 각국의 개발자들이 텔레그램 오픈소스를 이용해 윈도우, 리눅스, 웹 용으로 사용이 가능한 텔레그램을 개발하였으며, 삼성전자는 텔레그램을 기반으로 한 소셜라이저라는 모바일 메신저를 런칭하기도 함.

구글 크로뮴 프로젝트



이전 구글은 크롬이라는 인터넷 브라우저 발매후 크롬의 근간이 된 오픈소스인 크로뮴을 공개했다. 그 결과 많은 소프트웨어 개발자들이 구글 크로뮴 오픈소스를 활용해 새로운 크롬을 제작하거나 크롬 기반의 다양한 응용프로그램을 개발할 수 있게 되었다.

국내 오픈소스 확용시스템.



KRX는 유가증권과 선물 등 거래 시장 개설, 관리 및 유가증권 상장 등의 업무를 담당하는 유일한 유가증권 통합 거래소이다. 안정성이 중요한 이 회사는 KRX IT시스템에 x86 기반의 리눗스 거버를 적용했다.



KT 또한 주요 정보시스템에 오픈소스 SW를 도입했다. 전국에 설치된 KT 인증 시스템 등의 30개에 정보 시스템에 적용한 오라클 DBMS를 오픈소스인 엔터프라이즈DB(EDB) 포스트그래스 플러스 어드벤스트 서버(PPAS)로 교체했는데, 이는 5년간 2000억원의 비용절감 효과를 기대되게 하였다.



행정자치부도 온나라시스템(전국 지방자치단체 사용)에 오라클 DBMS를 PPAS로 교체했다.

출처:

<https://m.blog.naver.com/PostView.nhn?blogId=no1_hanafax&logNo=220532345337&proxyReferer=https%3A%2F%2Fwww.google.com%2F>

<http://m.etnews.com/20150902000045?obj=Tzo4OiJzdGRDbGFzcyI6Mjp7czo3OiJyZWZlcmVyIjtOO3M6NzoiZm9yd2FyZCI7czoxMzoid2ViIHRvIG1vYmlsZSI7fQ%3D%3D#_enliple>