구글 오픈소스 라운드테이블 속기록
이 문서는 2012년 2월 28일에 구글 코리아에서 있었던 “Google Open Source Roundtable”에서 나왔던 이야기들을 취합한 것입니다. 별도의 발표자료 없이 각 패널들을 중심으로 자유롭게 이야기하는 형식으로 진행되었으며 기록은 서기 역할을 맡아 주신 참석자 분들께서 해 주셨습니다.

귀중한 시간을 내어 좋은 경험과 이야기를 나누어 주신 패널 여러분들, 서기 역할을 맡아 수고해 주신 분들 그리고 참가해 주신 분들과 행사에 관심을 가지고 참가 신청을 해 주신 모든 분들께 감사드립니다.
패널: 허태준(Google)
여러 사람을 알게 되어 인맥관리에 좋다
Q. 패치가 들어오면 일일이 다 보는가?
A. 규모가 커지면 조직이 생긴다(느슨하게 나마) (기여도 등)비중에 따라 보는 양이 달라짐

Q. 오픈소스에 진입할려고 하는 초보분들 (완성도가 높은/버그가 별로 없을거같은)경우는 어떻게 해야하나?
A. 완성도가 높고 버그가 중요한게 아니라 복잡도가 증가되고 있으니, 충분히 진입할수 있는 곳들은 있다고 본다.
기술적으로 잘하는것도 중요하지만 익숙해지는게 중요하다 (몇번 리젝당한다고 좌절하지 말자 / 진입장벽(?))

Q. 리눅스 토발즈와 메인테이너들가 사이가 안좋은 ?? 메인테이너에서 내쫓지는 않나?
A. 개인 성격 나름 / 대부분 다음 쓰레드에서는 쿨하게 잊고 넘어간다

Q. 메인테이너가 되는 경우?
A. 자리가 비어서 주변에 하던 사람들에게 권유가 들어옴
메인테이너는 최종적인 결정권한을 하긴하지만, 중요한 컨트리뷰터들이 영향력이 더 크다

Q. 이런 패치는 안받아준다던지, 이런 형태를 권장한다던지
A. 메인테이너와 다른 사람들이 하는 이야기를 안듣고 싸우는 사람들은 받아들여지기 어렵다.
장기적인 방향이나 경험들에 의해 가기때문에 (대부분 긍정적인 방향)
처음엔 어설플 수 있어도 의견을 잘 듣다 보면 영향력이 커질수 있다.

Q. 외국분들과 의사소통
A. 읽는 정도는 대부분 되는데, 쓰는거에 대해 부담을 가지지 말아라.
중국분 사례) 영어는 엉망인데, 그렇게 크게 문제는 되지 않는다. 코드가 더 중요하다.
영어는 의사 소통의 수단일뿐... 자기가 컨트리뷰션할때 큰 문제라 생각 되지 않는다
비영어권 국가들도 상당히 많다.

Q. 메인테이너가 되게 된 계기?
A. 성능이 별로 안좋아서 다시 구현하고 그러다보니 메인테이너가 됨
=> 상당수의 패치가 자기가 하게 되거나 영향력이 크면 자연스레 메인테이너가 되게 됨
(패키지) 이거 했다가 저거했다가 왔다갔다 하는 경우가 종종 있음

Q. 배포판 어떤게 좋나요?
A. 엔터프라이즈 제품들 개발 사이클에 맞춰... 메인 라인 커널을 가져다 패치나 수정을 가하게 됨

Q. 메인테이너가 된다라는건 , 그렇게 중요한 권한인가 ?
A. 그 부분에 대한 관리 책임자(정리해서 제출) 최종권한은 리누스 토발즈

Q. 전에 하던 일보다 메인테이너가 된 후에 일의 양은?
A. 그다지 늘어난것 같지는 않다.

Q. 메인테이너가 커미터가 되기위해 직장을 그만둬야하나요?
A. 무조건 그만두지는 말아라.
채용을 할때 신규의 경우 학교나 논문들을 본다.
영향력있는 대기업들은 오픈소스 프로젝트에 얼마나 기여를 했느냐를 보게 된다.
열심히 파고 들어서 어느정도 영향력을 끼칠수 있는 가능성이 보이면 풀타임으로 가서 1~2년 정도 해보는것도 좋다

Q. 기여를 많이 하면 회사들이 알아주나?
A. 영향력있는 오픈소스 프로젝트에서 어느정도 지분을 가지고 있으면, 해당 회사들은 당연히 알 수 밖에 없으므로, 취업에 충분이 도움이 된다.

Q. 프로젝트마다 프로세스가 다른데 정리된 문서가 잘 없다 (특히 한글문서)
A. 잘 없음. 영어가 안되면 힘듬.
코딩 스타일 가이드가 있으니 그정도는 지켜줘야한다.

Q. 편집기는 뭐쓰세요?
A. 이맥스씁니다.

Q. 영향력이 있는데 발표같은건 안하시나요?
A. 가끔씩 발표도 합니다.
LSF, Kernel Submit 등

리눅스 커널을 조금이라도 했다하면, 삼성이나 LG 같은 대기업에 들어가기가 전혀 어렵지 않다.
Q. 첫 패치는?
A. lib ATA(?) 패치

Q. 패치 제출후 피드백이 늦는경우는?
A. ping .. 계속 건들여보면 된다 (짜증내지 않을정도로)
처음엔 3~4일 그 다음에는 한주간격
ping을 할때는 private mail 로 하는 경우가 많은데,
처음엔 private mail, 그 다음에는 cc 를 넣어 보낸다.

바쁘고 그러니깐 반응이 없는 경우가 종종 있다.
메인테이너가 기분나빠하고 그러진 않다(메인테이너의 의무(?))

Q. 커널같은 경우 버그의 재현이나 없어진걸 확인하는 절차?
A. unit test 환경이 힘들다, race condition , low memory , 테스트 환경을 정형화하기가 힘들다.
리그레이션 테스트는 계속 한다. -> 업스트림에 리포팅
코드로 살펴 볼수 없는경우 여러사람/시간을 두고 테스트 후 반영

Q. 오픈소스를 folk 하는경우?
A. 오픈소스가 발전속도가 느린경우는 잘 될 수 있으나 ,
안드로이드 같은경우 속도가 빠르다보니 따라가기가 힘들수 있음

miui 의경우 ICS는?
기본 UI만 따라간거같은...

Q. 직장을 그만두고 오픈소스에 뛰어들었을때 잘 안된 케이스가 있나?
A. 지금까지는 없는것 같다. 관심이 있고 재미가 있고 잘 할 수 있을거 같으면 (경력에 도움이 될수 있다) - 석사보다는 낫다

Q. 리눅스 커널을 새로 개발하신다면 바꾸고 싶은부분은?
A. 새로 만들기도 쉽지 않고, 오래된 코드들도 많다. 리팩토링도 좀 어렵다. 10~20년정도 가다보면 그런 코드들은 생길수밖에 없다. 그런 부분들을 잘 관리를 하는게 중요할것 같다.
새로 개발한다고 해서 좋아질것 같지는 않다.
리팩토링 같은 패치도 가끔씩 올라온다.

메인테이너도 가는 방향중 하나가 리팩토링같은 경우
(지저분해서 고쳤음, 새기능을 추가했음) 영향력 , 발언권도 세짐

Q. 리팩토링도 리뷰를 하는가?
A. 리뷰함. 시간은 그렇게 오래걸리지는 않는다. 패치를 보낸 사람의 신뢰도가 중요하다. co
메인테이너의 경우 리뷰 스타일도 중요한다.
처음에 진입장벽이라는게 신뢰를 쌓아가는 과정이라고 볼수 있다.

Q. 코드 외에 메일링같은데 질문 / 답변 같은 활동을 해도 영향력을 쌓을수 있는가?
A. 결과적으로는 코드패치이다. 메인테이너는 바쁜 경우가 많으므로...

Q. 메인테이너가 커미터?
A. 메인테이너가 커미터임. 패치를 모아서 commit

Q. 패치를 보낼때는?
A. 메인테이너에게 메일로 보내면 된다.

Q. 리눅스 커널 C++로 사용 ?
A. 싫다.. -> 프로그래머들간의 의사소통의 도구 이기도 함. 워낙 규모가 방대하기도 하고 프로그래머들의 의사소통에 방해가 될 수 있기때문에 권장하고 싶지는 않다.

Q. 다른 언어 ?
A. 좋아하는 C , asm, python , java, javascript / 싫어하는 c++ , ruby, perl
(system enginner 관점)

라이센스 관련
예전보다는 지금은 많이 좋아진것 같음

GPL, APL, BSD

Q. 선호하는 라이센스는?
A. 커널 개발자는 대부분 GPL v2
내가 쓰는 코드들이 다른 개발자들도 쓰고 입닦을수 없게 하기위한 일종의 안전장치(?)

Q. 오픈소스 사용하는데 사용할만한 툴들
이맥스, quilte , st git, gdb, (대단한 툴을 쓰지는 않는다)

Q. 개발환경은?
A. 30인치 모니터, 이맥스 x3, 터미널 두개

Q. 사용하는 배포판
fedora, gubuntu

패널: 허준회(Collabora)
Q. 오픈소스 프로젝트를 어떻게 끌고 나가는가?
(타이젠 SDK를 하면서 보니...)
A. 노키아 미고는 제품화에는 실패했으나 커뮤니티를 만드는 데에는 성공
먼저, 기술적으로 매력적인 프로젝트에는 자연스럽게 사람들이 몰린다.

Q. 좀 더 자세히...
우선 거의 모든 것이 공개되어야 함 - 특히 작업 레포지터리
(소스 공개에 시간차가 있으면 버그를 고쳤는데 알고보니 이미 고쳐놓았으면 맥빠지는 개발자들...)
가능한 모든 인프라를 공개해서 관심을 가지게 함
의사결정 및 회의까지 열어주면 더 좋음
주요 기능 및 방향에 대한 논의도 공개

Q. 참여하는 핵심 멤버가 바라는 점은?
A. 테스트와 버그 리포트를 가장 필요함 그리고 build break 수정도 환영
개발자가 잘 안하는 문서화도 필요함

Q. 기업의 오픈소스는 주로 어떤 때?
A. 사실 기업의 오픈은 잘 안되거나(심비안) 경쟁사에 비해 부족할 때(웹킷) 함
또는 기업의 규모(역량)이 작을 때 참여를 유도하기 위해(Gstreamer) 함

Q. 오픈소스 라이선스 관리는?
A. 큰 회사는 담당자가 다 있음
미리 라이선스의 섞임이나 충돌을 본 다음 내놓음
NIPA의 오픈소스 역량 프라자에서는 작은 기업에 한해서 어느 정도는 해줌
사실 검증은 굉장히 힘들기 때문에(개발자가 여기저기 가져옴) 소스는 물론 바이너리까지 인덱싱한 DB로 검사하는 솔루션도 있음

Q. 회사에서 업무와 함께 오픈소스 참여를 어떻게?
A. 회사 업무에 크게 구애받지 않고 자유롭게 오픈소스는 참여를 full time으로 하는 경우가 있지만 흔하지 않음. 회사에서 인정을 받으면 뭘 하든 내버려두기도 함...하지만 드문 경우.

주로 자기의 관심사보다는 회사와 관련된 문제를 보게 되고 이 부분을 기여하기는 회사 정책상 어려울 수 있음. 이때는 같은 프로젝트라고 하더라도 직적 업무과 관련 없는 다른 모듈을 보면서 기여 시작하면 좋을 듯. 아니면 회사를 설득해서 기여하는 것도 좋다.

가장 의미 있는 건 내가 사용하는 오픈소스의 문제를 해결하면서 참여하는 것. 당장 리눅스를 설치할 것!

Q. 큰 규모 프로젝트의 의견충돌은?
A. 서로 발전할 수 있다면 갈라져서 감
사실 흔한 일은 아님
작은 규모에서는 코어 개발자가 의사가 절대적임.

Q. 웹킷의 규모
A. 참여 정도에 따라 리뷰어 등으로 나뉘어짐
commiter, reviwer 명단은 사이트에서 찾아볼 있음

Q. 오픈해서 안좋을 있는 경우
A. 게으르면 경쟁사에 주도권을 빼앗길 수 있으나, 그런 사례는 찾기 힘듬.

구글러 의견 : 프로젝트 참여
레포지터리를 공개하고 누구나 참여하는 오픈소스가 구글의 개발과 비슷한 것 같음. 누가 활발하게 개발을 이끌어 나가느냐에 따라 프로젝트 주도권이 바뀌기도 함.
다만 영어 장벽이 있지만, 구글 일본의 경우 커뮤니케이션 능력이 좋아 잘 주도함.

Q. 어떻게 웹킷에 들어가게 되었나?
A. 사실 운이 좋았음(초창기+친절한 도움+...)

한글 관련 버그를 잡는 걸로 시작
우연히 버그를 리뷰하신분이 한국에 놀러와서 만나서 함께 작업하면 많은 것을 배움.
개발자가 적거나 아무도 안하는 블루오션 공략
시기도 중요함 – 일찍하면 좋음

Q. 뜨는 것 찾기?
A. 자기 취향에 맞게...또는 직접 만들어서 띄우기(!?)

Q. 소규모 프로젝트의 문제 ?
DirectFB의 경우 상용 제품에 많이 쓰였는데, 한 사람이 개발되다 보니 지속적이 지원이 부족해 GTK+3에서는 지원이 제외됨.

Q. 오픈소스 개발자의 장점은?
A. 충분히 자유롭게 일함
내가 하는 일에 대해 다른 사람과 공유하고 의견을 받을 수 있어서 좋음

Q. 기업이 자신 마음대로 오픈소스를 끌고다니는(?) 경우 어떻게?
A. 개발력이 센 게 갑이니 어쩔 수 없음

Q. 규모가 큰 프로젝트의 기능을 갈아엎음?
A. 성능상의 이점 등을 위해 은근히 빈번하게 있음

Q. 대학에서 바로 풀타임 오픈소스 개발자?
A. 우리나라의 경우 LLVM(서상현)
해외에는 많음(고등학교 때부터)

Q. 오픈소스 참여를 위한 지원이 필요하다면?
A. 멘토/멘티 프로그램 필요. 구글 SoC같은, 또는 대학에서의 교육

Q. 한성대학교에서의 교육?
A(이민석). 닌텐도DS로 교육함

Q. 오픈소스의 한글 지원 문제
A. 외국인에 대한 한글 이해가 부족해 문제가 많음. 국내 개발자 참여 필요.

Q. 오픈소스 회사 매출 구조?
A. 컨설팅, 개발, 교육, …
예)Gstreamer

마무리 : 그놈 개발 참여가 한국에서 적음
진정한 커뮤니티를 느끼려면 그놈에 오세요.

패널: 김국진(Samsung)
- 우리나라에서는 리눅스 커널을 개발하는 사람에 비하면 그것을 컨트리뷰션 하는 사람은 많지 않다. 한국 사람이 더 많이 했으면 좋겠다. 컨트리뷰션이라는 것이 처음 느끼는 마음의 진입장벽만큼 실제로 그 벽은 그리 높지 않다고 생각한다. 다른 개발자들이 컨트리뷰션에 대해 정해진 규약과 같은 기본적인 것부터 친절하게 리뷰를 해준다. 피드백을 받았을 때나 메인라인에 이름이 올라갔을 때의 뿌듯함이 있다.

- 질문 : 예전보다 오픈소스 컨트리뷰터로 들어갈 수 있는 분야가 줄어들었다?
답변: 오타만 잡아서 컨트리뷰션하는 사람도 있고, 코딩 스타일 같은 것을 리뷰만하면서 컨트리뷰션하는 사람도 있다. 그리고 어떤 특정 파일만 업데이트하는 사람도 있다. 아직도 블루오션이 많다. 처음에는 bug fix로 시작을 할 수 있다. 그리고 꾸준한 컨트리뷰션 후 인정받는 기간도 생각만큼 오래 걸리지 않는다.

- 질문 : 코딩을 하면서 Documentation을 많이 신경을 쓰는가?
답변 : 개발자들마다 다르고, 상황에 따라 다른듯하다.

- 질문 : 소스 공개에 대해서 거부감을 가지는 임원이 있다거나, 회사 정책과 어긋나는 부분은 없는가?
답변 : 오픈소스 기여자가 회사에 있으면, 그 기술의 방향에 영향을 미칠 수 있으므로, 회사의 비지니스에도 도움이 되고, 오픈소스의 그것을 윗 분들이 이해하고 있다. 최근 안드로이드 때문에 더욱 그렇게 된 부분도 있는 것 같다.

- 질문 : 큰 회사에 소속되어서 오픈소스 기여자가 될 수 있는 경우는 행복해 보인다. 그렇지 않은 경우는 어떻게 접근해야할까?
- 답변 : 큰 회사에 소속되어서 컨트리뷰션한다고 해서 100% 행복한 것 같지는 않다. 회사에서 하는 일과 별도로 취미로 오픈소스를 하다가 한계를 느끼고 본업을 그만두는 사람도 있다. 개인과 상황에 따라서 많이 달라서 어려운 문제인것 같다.
어떤 분야의 오픈소스이든, 회사의 사업에 도움이 될 수 있으므로 충분히 회사에 있으면서 오픈소스에 기여하고 싶은 사람에게 기회가 될 수 있다.
오픈소스 기여에 대해서 회사의 시선이 이전과 많이 바뀌었음을 많이 느낀다.
큰 대기업들이 오픈소스에 기여할 부분이 분명이 있다.

- 질문 : 언어 장벽은 없나?
- 답변 : face-to-face 토론이 일어나면 어려울 수도 있다. 그러나, 메일링 리스트을 통한 이야기는 크게 어렵지 않다. 그리고 우리가 네이티브가 아니라는 점을 알고 많이 배려를 해준다. 그리고 의사 소통 할 내용이 기술적인 내용들에 한정되어서 어렵지 않기도 하다.

- 질문 : 초보자가 할만한 이슈가 많은 추천할만한 프로젝트가 있는가?
- 답변 : elinux.org라는 프로젝트(http://elinux.org/Android_Mainlining_Project)를 추천한다.
http://kernelnewbies.org/KernelProjects 도 추천

- 질문 : 리뷰할 때 쓰는 툴이 있는가?
- 답변 : 메일링 리스트에서 주고 받을 때 몇 가지 규칙들이 존재한다. 80 칼럼 이내에서 자동

줄바꿈, 메일링 리스트에서 '>'의 원문 표시 다음에 답변을 하는 등... 기회가 된다면 그런 것에 대해서도 세미나를 하면 좋을 것 같다.

- Open Source Contribution을 겁먹지 말라!

패널: 김남형(LG)
- 어떻게 해야 지속성을 가지고 오픈소스에 기여할 수 있는가?
    - 비지니스 모델을 만들어 풀타임 근무를 하는 방법.

- 회사 일만 하다 보면 실력을 키우기가 어렵다
    - 오픈소스를 통해 실력 증진의 기회를 찾게 되면서 시작.

- 동기부여를 하는 방법
    - 피드백이 있다면 지속적인 개발에 대한 동기 부여가 가능하지 않을까?
    - 피드백을 받을 수 있는 좋은 방향(프로젝트)을 선택하는 것도 중요하다.

- 공유와 오픈에 대한 '분위기' 자체가 개발에 대한 동기 부여가 된다.
    - 학창 시절에는 경제적인 여건에 대한 문제 없이 개발할 수 있으나,
    - 지속적인 기여를 위해서는 생활을 유지하면서 지속할 수 있는 방법을 찾아야 한다.
    - 생활과 같이 하기 위해서는 출퇴근 시간을 활용해서라도.
    - 기존 오픈소스 개발자들의 이야기를 듣고 공유하는 기회가 있으면 좋겠다.

- 주어진 일로 하는 것과 하고 싶어서 하는 것은 다르다. 하고 싶어서 하는 쪽이 더 좋은 결과물을 낼 수 있다.
    - 먹고 살기 위해서 이 일을 했다면, 지금까지 하고 있지 못할 것이다.
    - 유행만 따라서 일을 한다면, 좋은 결과를 내기 어려울 것 같다.

- 오픈소스가 있다고 해도, 가져다 사용하는 것도 쉽지 않은 일이다.
    - 여기서 조금만 더 노력하면 직접 오픈소스가 참여하는 것이 가능하다.

- 요즘 진행되는 큰 오픈소스 (= 돈이 되는) 들은 큰 회사에서 스폰싱을 하여 진행하고 있으므로, 풀 타임 오픈소스 개발자를 할 수 있는 가능성이 많아졌다.
    - 풀 타임 오픈소스 개발을 할 수 있다면 근무 환경은 훨씬 좋아진다.
    - 높은 자유도.
    - 회사에서도 자기 회사의 이익을 위해 푸시하는 면이 없지 않아 있으므로, 회사에서도 잘 지원해주는 편.

        이러한 경우 지적 재산권에 대한 문제가 생기지 않는가?

        소프트웨어 라이센스는 복잡하게 얽혀있는 경우가 많다.

        J 사 등에서 특허와 관련된 문제를 겪은 적이 있다. 일부 회사는 자사의 이익을 위해 자사의 코드(장치?)와 호환성이 높은 방향으로 푸시하는 경우가 있다.

            - BSD 진영에서 이런 일은 상대적으로 어렵다. (이런 경우 대부분 리젝)
            - IPR?

        코드를 리뷰하는 사람들이 어떤 회사에서 지원을 받고 있느냐에 따라 리눅스의

경우는 편향된 결과를 가져오는 경우가 있는 것 같다.

GPL 등 이런 것이 뭘까? 이런 것에 항상 궁금증을 가지고 있었다. KLDP 등에 번역된 문서를 본 적이 있지만, 아직까지 라이센스에 대해 쉽게 이해하기가 어렵다.

- 상용 OS 기반의 소프트웨어 개발에 비해 리눅스 기반의 FOSS 개발은 생업을 해결해주기 어려울 것 같다.
- 리눅스 커널 소스에 기여를 하게 되면 기존 개발자들과 소통을 시작할 수 있고, 더 나아가 풀 타임 오픈소스 개발자가 되는 길이 열릴 것 같다.

- 어느 정도 기여해야 네임드(?)가 되는가.
- 어떻게 참여를 시작해야하는 지가 궁금하다.

나의 목표 중 하나도 네임드가 되는 것이다. ^^ 하지만 아무 것도 몰랐기 때문에 호기심 하나로 리눅스 소스를 바닥부터 보기 시작했다.

- 책을 통해 지식을 습득.
- 회사 생활과 병행하기는 어려운 것 같다.
- 과감하게 회사를 그만두고 학습에 매진.
- OS, CA, C 언어 등 아무것도 아는 것이 없었지만, 열정으로 소스코드를 하나씩 파보기 시작.
- 문서의 오타나 소스를 보다 이상하게 여겨지는 부분을 수정해 소스를 보내기 시작.
- 컴파일 과정에서 오류가 발생하는 경우 등 사소한 경우에도 패치를 제작해서 보냄.
- 프로젝트 페이지에서 제공하는 문서만 읽어봐도 참여할 수 있는 부분을 찾을 수 있다.
- 버그 리포팅 & 테스팅을 계속적으로 해주는 것도 대단한 기여.
- 이러한 과정을 통해 자기 역할을 찾아가게 된다.
- 정적 분석 도구를 통해 코드 상의 부족한 부분을 리뷰.
- 보냈던 패치에 대해 좋지 않은 의견도 받았지만 그것 또한 학습의 일부.
    - 피드백의 내용에 관계 없이 도전하자.
- 리눅스 커널은 어떤 파일을 누가 담당하는지 목록이 있기 때문에, 찾아서 메일을 보내는 것이 가능.
- for newbie 사이트를 통해 기초 정보를 습득할 수 있다.

초보들의 오픈소스 프로젝트에 커밋하기에는 diff, patch의 장벽이 있다.
참여하고자 하는 프로젝트의 메일링리스트를 유심히 보다 보면, 어떤 방식으로 참여해야 하는지 알 수 있다.

- 답변을 잘 해주는 메인테이너를 골라보자.
- 개발자를 위한 메일링 외에 사용자를 위한 메일링도 있으니 이 쪽도 참고.
- 같은 메일이라도 메인테이너에 따라 답변이 다르다.
- 방향이나 연관 이슈를 제시해주는 개발자를 만날 수도 있다.

처음 패치를 보내는 것은 어려웠다.

- 처음으로 보낸 패치를 메인테이너가 Welcome to open source! 와 같은 답장을 받아 큰 힘을 얻었다.

구글에 처음 들어왔을 때 경험했던 것과 비슷하다. 구글 소스 코드 트리에서도 다른 조직에서

관리하는 트리에 커밋하는 것은 (괜히) 어렵게 느껴진다.

- 20줄을 커밋했는데, 40줄의 커멘트를 받는다. (장점)
- 유닛 테스트가 타이트하다 보니 소스 코드 량은 점점 늘어나게 된다.
    - 최초의 커밋은 작아도, 피드백을 받아 작업하다 보면 충분히 의미 있는 패치가 될 수도 있다.
- 역시 시도하는 것이 중요.
- 구글 내에서는 가이드가 잘 정립되어 있어서 오픈소스 진영에 비해 접근하기가 비교적 용이.

한국어로 오픈소스 생태계에 대한 소개 글이 있으면 도움이 될 것 같긴 한데, 어차피 오픈소스 세계는 영어 중심이므로 한국어의 의미가 그리 크지는 않을 것 같다.

- 일정 규모가 된다면 로컬 커뮤니티도 의미가 있을 수 있다. (예: 일본어)
- 번역 등에 대한 국가적인 지원이 필요하지 않을까?
    - 기술적인 내용을 포함하여 제대로된 번역을 하기는 정말 어렵다.
    - 선순환 비지니스 모델이 생길 수 있을까?
- 한국어 커뮤니티가 양적/질적으로 좋은 모델을 갖추기에는 생태계가 너무 작다.
- 오히려 글로벌 커뮤니티에서 영어로 소통하는 때에 피드백을 받기가 용이하다.
- 기본적인 의식주에 대한 문제가 해결되는 사회 (= 유럽) 에서는 오픈소스를 시작하고, 풀타임으로 하기에는 더 좋은 여건이 된다.
    - 기본적인 생활 여건이 나아진다면 사용자들의 마인드가 좋아지거나 오픈 소스에 참여 하는 사용자들의 풀이 늘어날 수 있을 것 같다.
- 아직까지 우리나라에서는 BSD나 리눅스에서 FOSS에 참여하여 생업을 해결할 수 있는 기회를 가지기는 어려운 것 같다.

오픈소스에 참여하고 그 이력을 통해 구직을 했는데 초기에는 좋은 결과가 없었지만, 점차 더 오픈소스에 기여한 이력이 구직 활동에 도움이 될 것으로 예상된다.

- 무엇보다 자기의 실력 증진에 도움이 되고,
- 자기가 활동한 내역이 공개되므로 좋다.
- 무조건 새로운 것을 만들어야 할 이유는 없다. 다른 사람들의 코드를 리뷰해주는 것도 중요한 일이다. 리뷰어로 활동하다 레드햇에 입사한 경우도 있다.
- 낯선 프로젝트에 접근할 때에는 makefile 부터 접근하는 것도 좋은 방법이다.