2023 오픈소스 컨트리뷰션 아카데미

Open Source Contribution Academy

uftrace

Project Guide

1 프로젝트 개요

프로젝트 분야 · 활용 언어 · Repository · 난이도 · 참가자 모집 유형 및 우대사항 등

1 프로젝트 개요

프로젝트명 : uftrace

프로젝트 분야 : OS, Mobile, 임베디드, 클라우드, OSS 전반

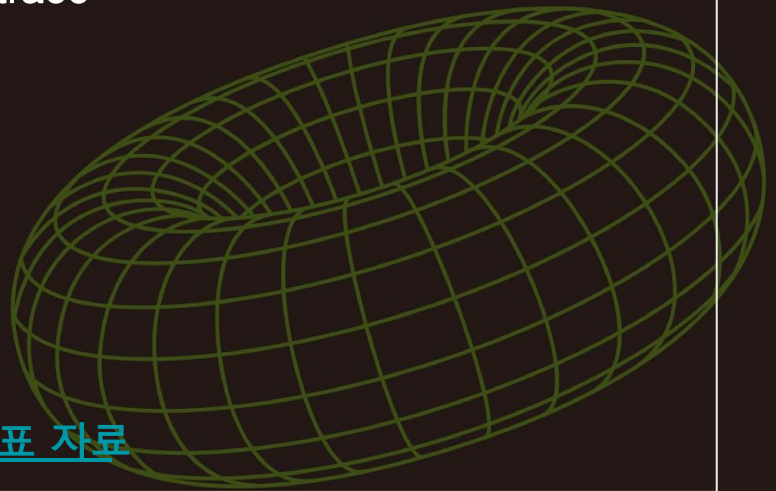
프로젝트 저장소 : https://github.com/namhyung/uftrace

활용 언어 : C, C++, Rust, Python

프로젝트 난이도 : 중

관련 자료

- CppCon 2016 발표 영상
- CppCon 2017 발표 영상
- uftrace 튜토리얼 슬라이드
- 2019 한국소프트웨어종합학술대회 튜토리얼 발표 자료



1 프로젝트 개요

참가자 모집 유형

- Low-level 시스템 프로그래밍에 관심 있으신 분
- 유명 오픈소스 프로젝트에 활동 이력을 남기고 싶으신 분
- 리눅스 기반 C/C++/Rust/Python 프로그램 Tracing/성능 분석 도구 개발에 관심 있으신 분
- 컴파일러 및 바이너리 구조 분석에 관심 있으신 분
- 다양한 프로그램의 내부 구조 및 실행 흐름을 분석하고 싶으신 분 [ex. C, C++ (STL, Boost), Rust (STL), CPython, V8 JS Engine]

우대 사항

- 사전 과제를 수행하여 제출하신 분 (사전과제 링크)
- C/C++/Python/Rust/Bash Script에 익숙하신 분
- git을 활용한 버전 관리 및 협업이 가능하신 분
- JavaScript 기반 Frontend 작업이 가능하신 분
- 영문 문서를 읽고 쓰는데에 어려움이 없으신 분
- 기술 지식을 문서 자료로 정리하는데 관심 있으신 분

2

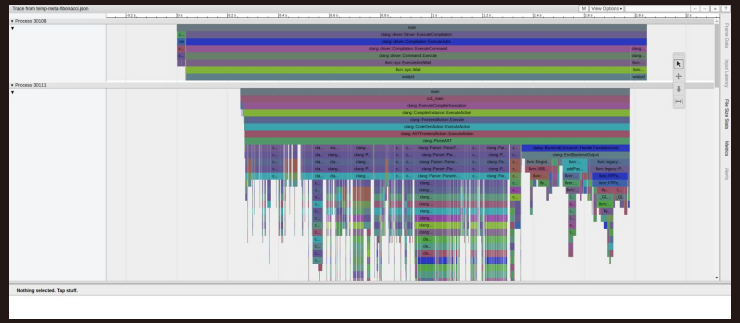
프로젝트 소개

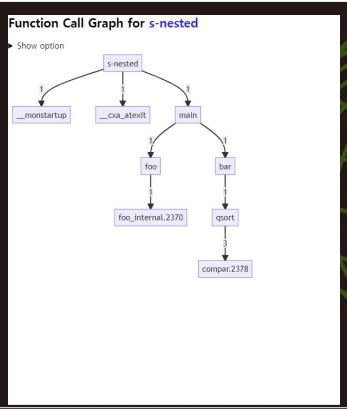
프로젝트 상세 소개 내용

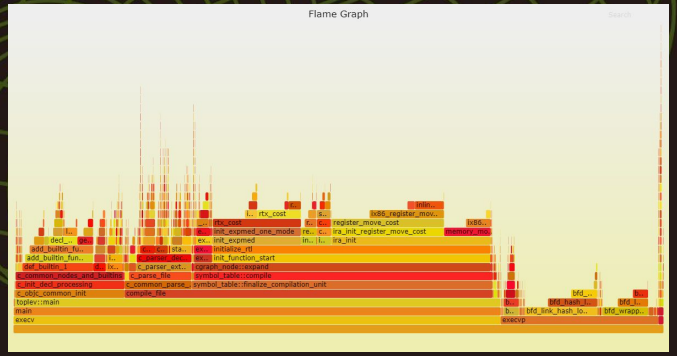
2 프로젝트 소개

주요내용 1

uftrace는 C/C++/Rust로 작성된 프로그램을 코드 수정 없이 함수 단위로 성능을 측정하고 실행흐름을 추적(trace)할 수 있는 분석 도구입니다. 이러한 분석을 위해 어떠한 코드 수정도 필요 없이 단지 기존에 존재하는 컴파일러 옵션의 도움으로 분석이 가능합니다.Node.js, Chromium, MySQL, GCC, Clang/LLVM, CPython 등 C, C++, Rust, Python 언어로 구현된 다양한 오픈소스 프로젝트의 내부 동작을 분석, 디버깅 및 성능 분석 레포트 시각화로 개선점 분석에 활용합니다.



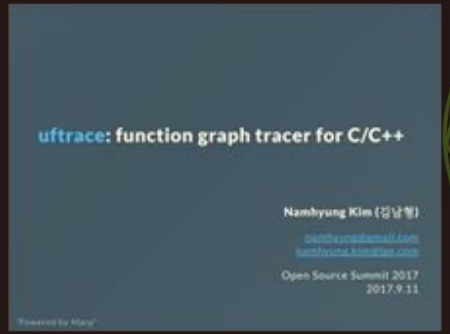




2 프로젝트 소개

주요내용 2

2016년 github에 공개된 이후 Linux Tracing Summit, CppCon 등 해외 컨퍼런스에 소개되며 사용자 층을 넓혀가고 있으며, 2019 공개 SW 컨트리뷰톤 대상 및 2022 수상작입니다.



2 프로젝트 소개

주요내용 3

uftrace는 개발자를 위한 범용성 높은 도구입니다.

어플리케이션의 유저 뿐만 아니라 커널 측면에서 어떤 동작을 하는 지 분석이 가능합니다.

Full dynamic tracing 기법의 도입으로 컴파일러의 도움 없이도 바이너리 분석 및 조작을 통해서도 분석이 가능합니다.

특히 시스템, 임베디드 프로그램 및 정보보안 분야등에서 사용될 수 있으며 C/C++/Rust/Python을 사용하는 개발자 누구나 성능 개선점을 찾거나 소프트웨어 분석을 할 때 유용하게 사용할 수 있습니다.

```
$ sudo uftrace -K 2 a.out
Hello World!
# DURATION     TID     FUNCTION
            [162954] | main() {
            [162954] |   printf() {
            [162954] |     sys_newfstat() {
   3.074 us [162954] |       vfs_fstat();
   1.200 us [162954] |       cp_new_stat();
   6.914 us [162954] |     } /* sys_newfstat */
            [162954] |     __do_page_fault() {
   0.643 us [162954] |       down_read_trylock();
   0.630 us [162954] |       _cond_resched();
   2.440 us [162954] |       find_vma();
   9.697 us [162954] |       handle_mm_fault();
   0.690 us [162954] |       up_read();
  21.813 us [162954] |     } /* __do_page_fault */
  42.064 us [162954] |   } /* printf */
            [162954] |   fflush() {
            [162954] |     sys_write() {
   0.933 us [162954] |       __fdget_pos();
  18.310 us [162954] |       vfs_write();
  27.117 us [162954] |     } /* sys_write */
  30.673 us [162954] |   } /* fflush */
  74.223 us [162954] | } /* main */
```

Dynamic tracing

- uftrace patches only NOPs to mcount call at runtime.
  - `main` doesn't have NOPs so won't show up in the result.

```
$ gcc foobar.c
<bar>:
  call <mcount@plt>
  ret

<foo>:
  call <mcount@plt>
  call <bar>
  ret

<main>:
  call <foo>
  call <bar>
  ret
```

```
$ uftrace -P . a.out
# DURATION     TID     FUNCTION
            [ 14813] | foo() {
   0.507 us [ 14813] |   bar();
   4.130 us [ 14813] | } /* foo */
   0.240 us [ 14813] | bar();
```

Full Dynamic Tracing

- uftrace patches mcount call at runtime without compiler support.

```
$ gcc foobar.c
<bar>:
  call <mcount@plt>
  ret

<foo>:
  call <mcount@plt>
  call <bar>
  ret

<main>:
  call <mcount@plt>
  call <foo>
  call <bar>
  ret
```

```
$ uftrace -P . a.out
# DURATION     TID     FUNCTION
            [ 14830] | main() {
            [ 14830] |   foo() {
   0.303 us [ 14830] |     bar();
   2.433 us [ 14830] |   } /* foo */
   0.176 us [ 14830] |   bar();
   4.917 us [ 14830] | } /* main */
```

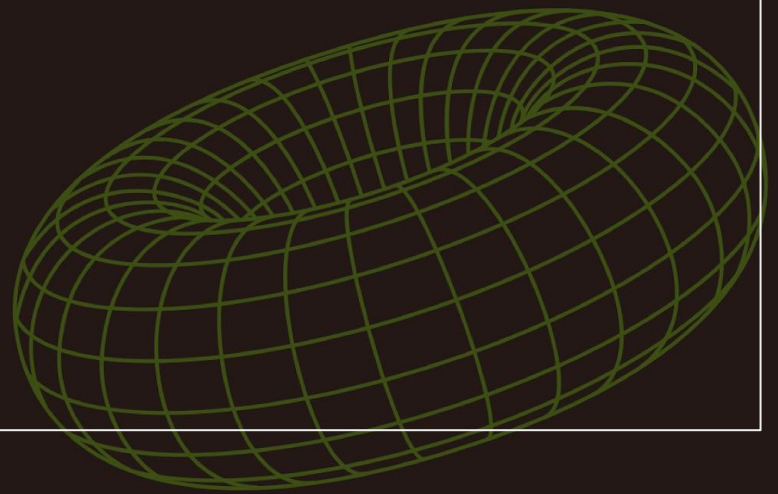
3 컨트리뷰션 가이드

단계별 컨트리뷰션 커리큘럼

3 컨트리뷰션 가이드

컨트리뷰션 코스 1 Git/Github을 활용한 협업 훈련

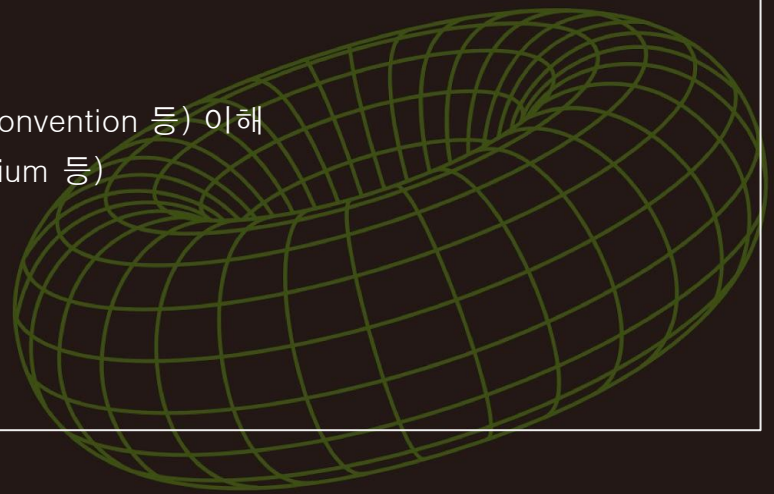
- Git/Github의 기본 실습
- 오픈소스 개발 방식의 이해
- 협업 시 발생하는 주요 이슈 및 해결 방법의 이해
- CONTRIBUTING.md 숙지
- README.md 숙지



3 컨트리뷰션 가이드

컨트리뷰션 코스 2 프로젝트 배경지식 습득

- 프로젝트에 대한 이해 (사용 분야 및 프로젝트 정의)
- GDB 및 debugpy 를 활용한 디버깅 습득
- 유닛 테스트 인프라 활용하기
- 도커를 활용한 빌드 및 자동화 테스트
- Adress Sanitizer 및 정적분석 도구 활용하기
- 바이너리 구조 분석을 위한 필수 지식 (ELF, PLT/GOT, calling convention 등) 이해
- 주요 오픈소스 프로젝트 분석 사례 공유 (node.js, clang, chromium 등)
    - https://github.com/namhyung/uftrace/wiki
    - https://uftrace.github.io/

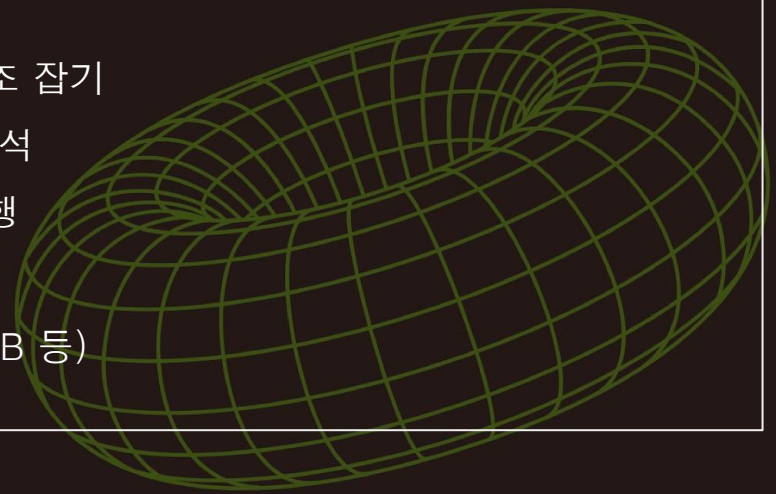


3 컨트리뷰션 가이드

컨트리뷰션 코스 3 프로젝트 개발 환경 구성

- 개발 및 테스트 환경 구성: 리눅스 환경
  - X64, Arm64 라즈베리 파이 또는 가상화 환경
  - VS Code 또는 Vim, 디버깅 툴 등 세팅
- GIt 개발 환경 구성: local, remote repo, upstream 구조 잡기
- 프로젝트 컴파일, 실행: 기본적인 예제 코드 실행 및 분석
- 프로젝트 테스트 방법의 이해: 테스트 코드 분석 및 실행
- 개인적으로 분석하고 싶은 프로젝트 분석 시도

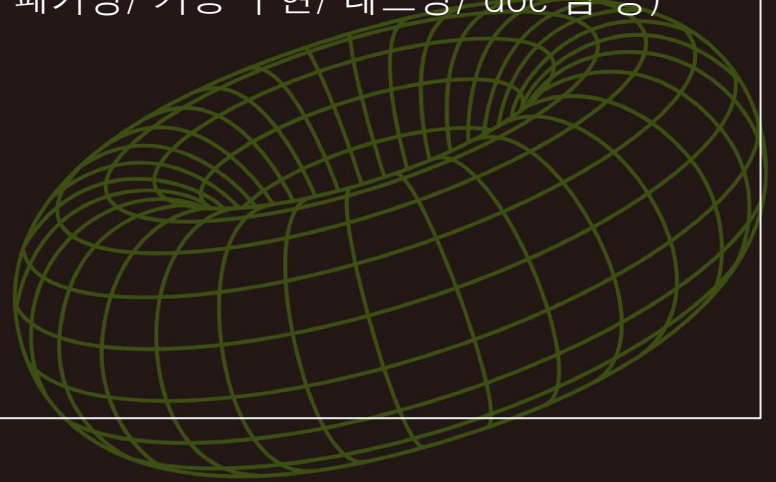
  (ex. node.js Engine, CPython internal, MYSQL InnoDB 등)



3 컨트리뷰션 가이드

컨트리뷰션 코스 4 프로젝트 분석

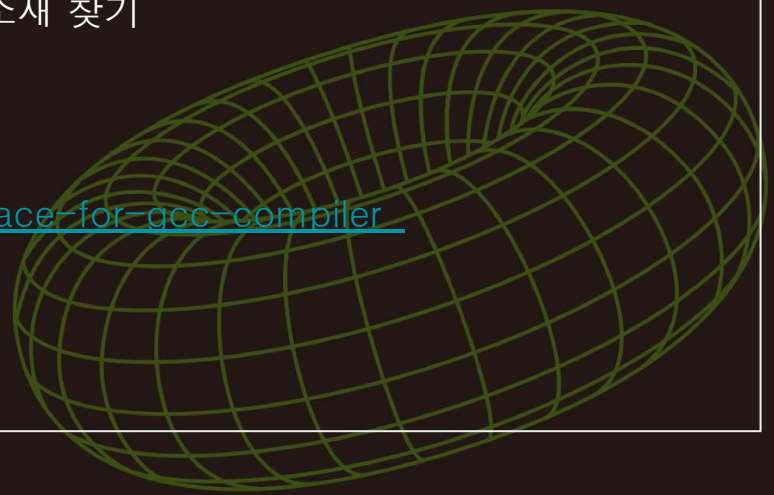
- uftrace 주요 기능에 대한 소스 분석
- 분석 진행 과정/ 결과 등을 공유하고 함께 고민
- 3~4개 조 결성, 멘티 당 한 조 이상 편성 (ex. UX 개선/ 패키징/ 기능 구현/ 테스팅/ doc 팀 등)
- 조별 목표 선정 후 프로젝트 분석 및 협업



3 컨트리뷰션 가이드

컨트리뷰션 코스 5 프로젝트 기여

- 조별/전체 온/오프라인 모임 참여
- 코스 4에서의 분석을 바탕으로 조별 미션 수행
- 버그/리팩토링/Minor feature/문서화 등 Trivial Patch 소재 찾기
- Pull Request를 통한 프로젝트 기여
- 다양한 오픈소스 프로젝트 usecase 문서화
  - https://github.com/namhyung/uftrace/wiki/uftrace-for-gcc-compiler
- 분석 및 기여 경험을 바탕으로 기술 발표 (선택 사항)
  - SOSCON 등



4

컨트리뷰션 운영 방안

단계별 컨트리뷰션 커리큘럼

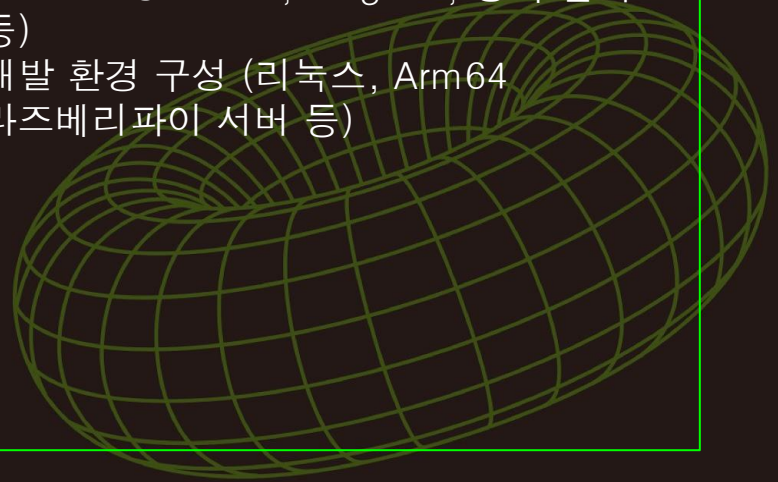
4 컨트리뷰션 운영 방안

1st Week

- 발대식
- 아이스 브레이킹
- uftrace 소개
- 핵심 목표 소개
  - 오픈소스 초심자도 uftrace 에 컨트리뷰션할 수 있는 가이드 작성
  - uftrace Github 홈페이지 및 매뉴얼 사이트 구성하기
  - uftrace Wiki 페이지 보완 작업
  - D3.js 를 활용한 데이터 시각화가 가능하도록 작업
  - 기존에 지원하는 기능들에 대한 유닛 테스트 통과 등 꾸준하게 개선
  - 병렬 태스크에 대한 측정 개선
  - RISC-V 64환경에서 uftrace PoC
  - 새로 추가된 python tracing 기능 테스트 및 개선
- git/github 협업 훈련

2nd Week

- 프로젝트 지식 습득
- 프로젝트 분석 툴 사용법 숙지 (GDB, Address Sanitizer, Valgrind, 정적 분석 등)
- 개발 환경 구성 (리눅스, Arm64 라즈베리파이 서버 등)



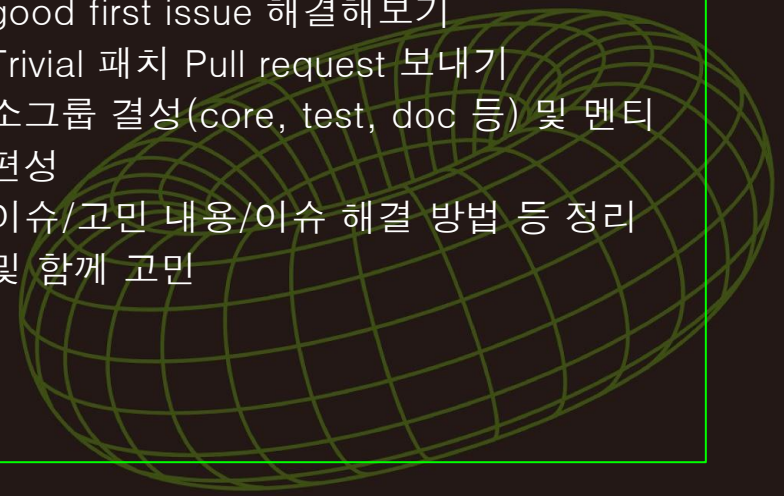
4 컨트리뷰션 운영 방안

3rd Week

- [멘티 교육] 기여하고 싶은 분야 소개
- 기본 예제 코드 실행 및 분석
- 테스트 코드 실행 및 분석
- 분석하고 싶은 프로젝트 분석 시도
- 프로젝트 usecase 작성

4th Week

- [멘티 교육] 프로젝트 분석 1 및 필요 기초 세션
- good first issue 해결해보기
- Trivial 패치 Pull request 보내기
- 소그룹 결성(core, test, doc 등) 및 멘티 편성
- 이슈/고민 내용/이슈 해결 방법 등 정리 및 함께 고민



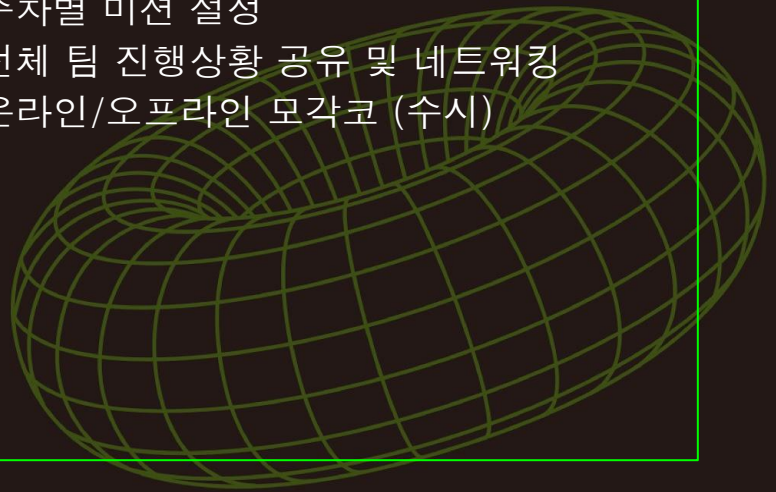
4 컨트리뷰션 운영 방안

5th Week

- [멘티 교육] 프로젝트 분석 2 및 필요 기초 세션 진행
- good first issue 해결해보기
- Trivial 패치 Pull Request 보내기
- 소그룹별 미션 설정 (doc, wiki, 시각화, test 등)
- 소그룹별 미션 수행

6th Week

- [멘티 교육] QnA 및 필요 세션 진행
- 소그룹별 미션 수행
- 주차별 미션 설정
- 전체 팀 진행상황 공유 및 네트워킹
- 온라인/오프라인 모각코 (수시)



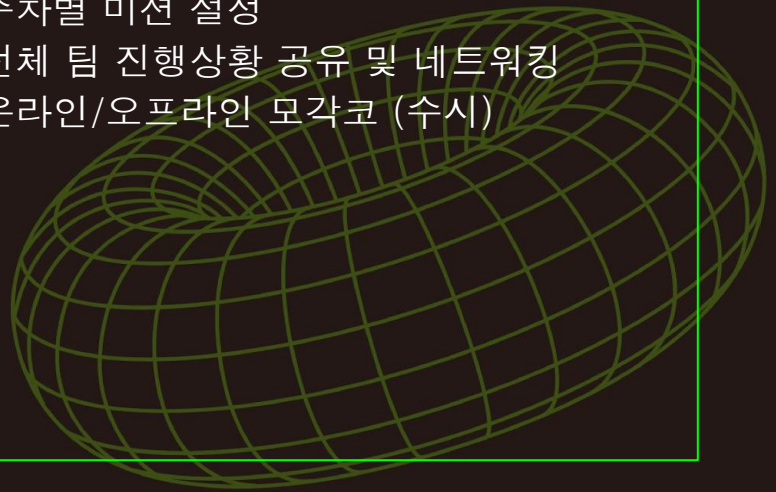
4 컨트리뷰션 운영 방안

7th Week

- [멘티 교육] QnA 및 필요 세션 진행
- 소그룹별 미션 수행
- 주차별 미션 설정
- 전체 팀 진행상황 공유 및 네트워킹
- 온라인/오프라인 모각코 (수시)

8th Week

- [멘티 교육] QnA 및 필요 세션 진행
- 소그룹별 미션 수행
- 주차별 미션 설정
- 전체 팀 진행상황 공유 및 네트워킹
- 온라인/오프라인 모각코 (수시)



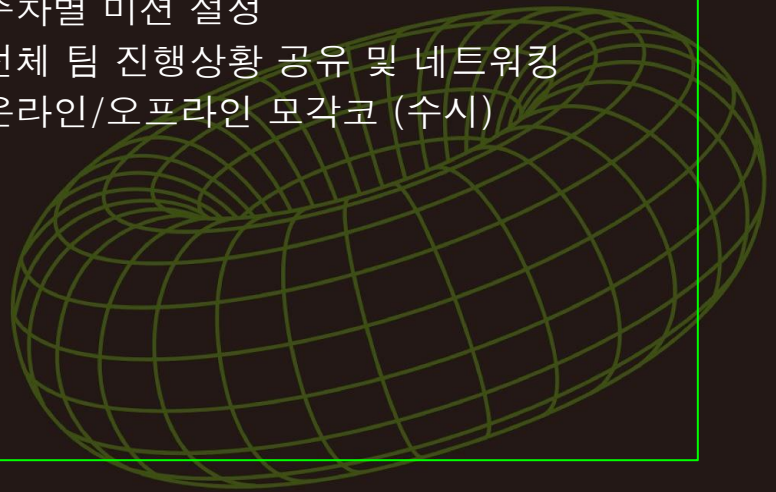
4 컨트리뷰션 운영 방안

9th Week

- [멘티 교육] QnA 및 필요 세션 진행
- 중간공유회 준비 (issue, PR 등 집계 및 발표 멘티 선정)
- 중간 보고서 작성

10th Week

- [멘티 교육] QnA 및 필요 세션 진행
- 소그룹별 미션 수행
- 주차별 미션 설정
- 전체 팀 진행상황 공유 및 네트워킹
- 온라인/오프라인 모각코 (수시)



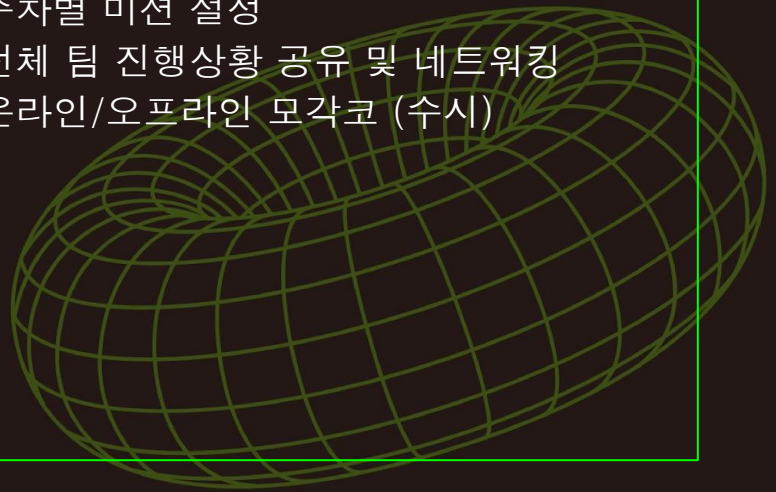
4 컨트리뷰션 운영 방안

11th Week

- [멘티 교육] QnA 및 필요 세션 진행
- 소그룹별 미션 수행
- 주차별 미션 설정
- 전체 팀 진행상황 공유 및 네트워킹
- 온라인/오프라인 모각코 (수시)

12th Week

- [멘티 교육] QnA 및 필요 세션 진행
- 소그룹별 미션 수행
- 주차별 미션 설정
- 전체 팀 진행상황 공유 및 네트워킹
- 온라인/오프라인 모각코 (수시)



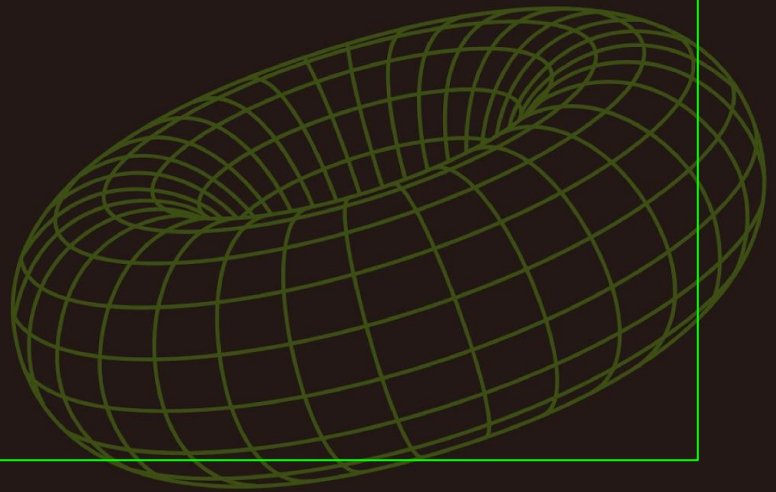
4 컨트리뷰션 운영 방안

13th Week

- 최종 평가 준비
- 미완성 목표 달성
- 최종 보고서 작성

마무리

- 최종 평가 준비
- 활동 기간 별 기여 목록 정리



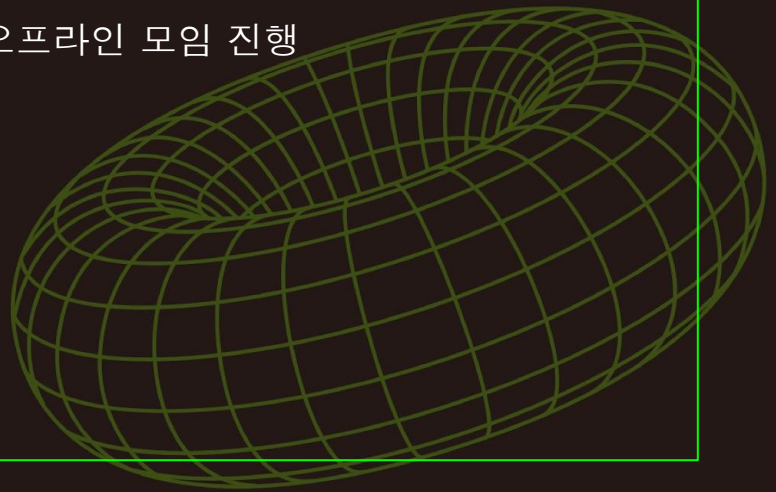
4 컨트리뷰션 운영 방안

◎ ONLINE

- 디스코드(Discord)를 활용하여 수시로 소통 (필요에 따라 화면 공유와 음성 통화를 사용함)
- 구글 공유 문서를 활용하여 멘토/멘티별 매주 진행 상황 공유

◎ OFFLINE

- 가능한 시간을 사전 조사하여 주기적인 시간을 정하고 OpenUp 서초 센터에서 오프라인 모임 진행



5

멘토 소개

컨트리뷰션 프로젝트팀 멘토단 소개

5 멘토 소개

- **김윤성**
- 오픈 소스 컨트리뷰터
- 2022 OSSCA uftrace 멘티
- 2022 한국 리눅스 커널 개발자 모임 audit 세미나
- 2021 공개 SW 개발자 대회 동상

- **진성희**
- 2022 OSSCA uftrace 멘티
- 2021 OSSCA linux kernel networking stack 멘티

2023 오픈소스 컨트리뷰션 아카데미

Open Source Contribution Academy

uftrace

컨트리뷰션에 도전해 보세요!

📁 THANK YOU 💻