

2024 오픈소스 컨트리뷰션 아카데미

Open Source Contribution Academy



uftrace

 Project Guide

2024 Open Source Contribution Academy 2024 Open Source Contribution Academy 2024 Open Source Contribution Academy 2024 Open Source Contribution Academy

1

프로젝트 개요

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

1 프로젝트 개요

프로젝트명 : uftrace

프로젝트 분야 : Tracing, Profiling, Program Instrumentation,
Binary Analysis

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

활용 언어 : C/C++, Python, Rust

프로젝트 난이도 : 중 ~ 상

1 프로젝트 개요

참가자 모집 유형

- Low-level 시스템 프로그래밍 / ELF 바이너리 구조 분석 / DWARF Debugging Info 에 관심 있으신 분
 - 리눅스 기반 함수 호출 추적 및 성능 프로파일링 도구 개발 에 관심 있으신 분
 - RISC-V 아키텍처 포팅 작업 에 관심 있으신 분
 - 기술 지식을 문서화 하는 작업 에 관심 있으신 분
- ★ 위에 있는 내용이 뭔지 하나도 모르겠지만 관심과 열정을 가지고 해보실 분

우대 사항 ★

- [사전과제 링크](#)로 주어진 사전 과제를 수행하여 제출하신 분
- C/C++, Rust, Python, Shell Script 언어 중 하나에 익숙하신 분
- git 과 github을 활용한 버전 관리에 익숙하고 협업이 가능하신 분
- 리눅스 기반 성능 프로파일링 기법(ex. ftrace, mcount func, gcc의 instrument-function 등) 에 익숙하신 분



2

프로젝트 소개

프로젝트 상세 소개 내용

2 프로젝트 소개

uftrace 개요 및 사용 분야

[uftrace 개요]

- uftrace는 코드 수정 없이 C/C++, Rust, Python 언어로 구성된 프로그램의 내부 함수 호출 흐름을 추적하고 각 함수 호출 시 소모되는 성능을 측정하는 분석 도구입니다.
- 기본적인 함수 추적 기능 외에도 함수의 인자 및 반환 값을 확인하는 기능, 라이브러리 함수 호출 추적 기능, 필터링 및 시각화 기능, 컴파일러의 도움 없이 추적할 수 있는 Full Dynamic Tracing 기능 등 다양한 기능을 제공합니다.

C/C++ (user) Function Tracing

```
• uftrace live
  ◦ Trace functions both doing record and replay

$ gcc -pg foobar.c

$ uftrace live a.out
# DURATION   TID      FUNCTION
[112643] | main() {
[112643] |   foo() {
0.190 us [112643] |   bar();
1.083 us [112643] | } /* foo */
0.127 us [112643] | bar();
2.050 us [112643] | } /* main */
```

Detect function types using debug info with -a/--auto-args

```
$ gcc -pg -g fibonacci.c
$ uftrace -o a.out 5
fib(5) = 5
# DURATION   TID      FUNCTION
1.478 us [31321] | main(2, 0x7fffd62e92a18) {
[31321] |   atoi();
[31321] |   fib(2) {
[31321] |     fib(1) {
0.155 us [31321] |       fib(0) = 1;
0.123 us [31321] |       fib(1) = 1;
0.883 us [31321] |     } = 2; /* fib */
0.125 us [31321] |     fib(2) = 1;
1.483 us [31321] |   } = 3; /* fib */
[31321] |   fib(3) {
0.125 us [31321] |     fib(2) = 1;
0.125 us [31321] |     fib(1) = 1;
0.774 us [31321] |   } = 2; /* fib */
2.716 us [31321] |   } = 5; /* fib */
4.382 us [31321] |   printf("fib(%d) = %d\n" = 11;
9.456 us [31321] | } = 0; /* main */
```

Library Function Tracing

```
• Library Function Tracing works via PLT hooking
  ◦ PLT contains trampoline code for calling shared functions

$ gcc -pg foobar.c

void bar() {
  getpid();
}

void foo() {
  bar();
}

int main() {
  foo();
  bar();
}

<bar>:
call <__count@plt>
call <getpid@plt>
ret

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

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

Full Dynamic Tracing

```
• uftrace patches mcount call at runtime without compiler support.

$ gcc foobar.c
$ uftrace -P a.out
# DURATION   TID      FUNCTION
<bar>:
call <__count@plt>
ret
0.303 us [14830] | 14830 | main() {
[14830] | 14830 |   foo() {
[14830] | 14830 |     bar();
2.433 us [14830] | } /* foo */
0.176 us [14830] | 14830 | bar();
4.917 us [14830] | } /* main */

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

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

2 프로젝트 소개

uftrace 개요 및 사용 분야

[uftrace 사용 분야]

- uftrace는 범용성을 추구하기 때문에 성능 추적 및 분석 도구를 활용하기 위한 목적이라면 대부분 사용 가능하지만, 그 중 아래와 같은 분야에서 유용하게 사용이 가능합니다.
- 특정 아키텍처에 따라 기능에 일부 제한이 있지만, 아래와 같이 다양한 아키텍처를 지원하기 때문에 시스템 및 임베디드 프로그램 개발 시 사용할 수 있습니다.
 - Intel 및 AMD 계열 CPU : x86, x86_64
 - ARM 계열 CPU : ARMv6 및 ARMv7, AArch64
 - RISC-V 계열 CPU : riscv64 (Dynamic Tracing 기능 미지원)
- 정보보안 분야에서 버그를 찾거나 취약점을 분석하는 등의 목적으로 사용될 수 있으며, 실제 사용된 사례는 아래와 같습니다.
 - Ref URL 1 : <https://www.youtube.com/watch?v=mI4OXEwhUF8>
 - Ref URL 2 : <https://github.com/eProsima/Fast-DDS/security/advisories/GHSA-v5r6-8mvh-cp98>

2 프로젝트 소개

uftrace 오픈소스 프로젝트 이력

국내외 다수 컨퍼런스에 소개되어 국내 뿐 만 아니라 외국의 컨트리뷰터들도 해당 프로젝트에 기여하고 있으며 2019년도 공개 SW 컨트리뷰톤 및 2022, 2023년도 오픈소스 컨트리뷰션 아카데미 수상작입니다.



uftrace: function graph tracer for C/C++

Namhyung Kim (김남형)
namhyung@kernel.org
namhyung.kim@ae.com
Open Source Summit 2017
2017.9.11



2019 공개 SW 컨트리뷰톤 프로젝트 수상팀 발표

대상	과학기술정보통신부장관상	uftrace
최우수상	정보통신산업진흥원장상	YOLK(You Only Look Keras) RustPython/gzython
우수상	정보통신산업진흥원장상	Armeria 구름 입체기 Flutter Moum
장려상	정보통신산업진흥원장상	mocha 자바스크립트 튜토리얼 한글화 React Native Tutorial



2022 오픈소스 컨트리뷰션 아카데미
Open Source Contribution Academy

Congratulations!

Contribute • Marathon • Contribution!

- 대상 과학기술정보통신부장관상 RustPython
- 최우수상 정보통신산업진흥원장상 GlueSQL
- 우수상 정보통신산업진흥원장상
- 장려상 정보통신산업진흥원장상
- uftrace



2023 오픈소스 컨트리뷰션 아카데미
Open Source Contribution Academy

Congratulations!

Contribute • Marathon • Contribution!

- 대상 과학기술정보통신부장관상 Argo Workflows
- 최우수상 정보통신산업진흥원장상 uftrace
- 우수상 정보통신산업진흥원장상 python-mysql-replication
- 장려상 정보통신산업진흥원장상 GlueSQL
- 특별상 한국IT비즈니스진흥원장상 FOSSLight Hub
Githru-vscode-ext
FOSSLight Hub
OpenStack

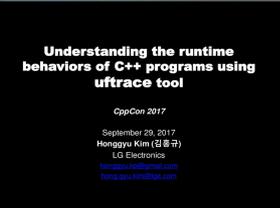


cppcon | 2017

HONGGYU KIM

Understanding the runtime behaviors of C++ programs using uftrace tool

CppCon.org



Understanding the runtime behaviors of C++ programs using uftrace tool

September 29, 2017
Honggyu Kim (김홍규)
LG Electronics
hongyu.kim@lge.com
hong.gyu@lge.com

2 프로젝트 소개

관련 자료

[uftrace 튜토리얼 슬라이드]

- URL : <https://uftrace.github.io/slide/>

[CppCon 2017 발표 영상]

- URL : <https://www.youtube.com/watch?v=s0B8hV2O8ps>

[2019 한국소프트웨어종합학술대회 튜토리얼 발표 자료]

- URL : https://uftrace.github.io/talks/uftrace_KSC_tutorial.pdf



3

컨트리뷰션 운영 방안

프로젝트 월별 활동 계획

3 컨트리뷰션 운영 방안

7월 (Challenges 7.13~8.9)

[첫째 주]

- 선발된 멘티들에게 uftace 개발 환경 구축 방법 사전 공유

[둘째 주]

- 발대식 참석 및 아이스 브레이킹

[셋째 주]

- 오픈소스 첫 기여자를 위한 git 사용법과 좋은 커밋 메시지 작성을 위한 커밋 컨벤션 소개
- uftace 프로젝트의 기여 방법 소개
- 이미 경험이 있는 기여자를 위한 Good First Issue 소개

[넷째 주]

- uftace가 프로그램을 추적하는 방식을 이해하기 위한 배경지식 습득
- 간단한 예제 코드를 활용해 습득한 배경지식 이해

8월 (Masters 8.10~11.2)

[첫째 주]

- uftace 코드 저장소 구조 및 프로그램을 추적하는 부분에 대한 코드 분석
- Python Tracing, Rust Tracing, C/C++ Tracing, Visualization, RISC-V Arch 포팅 등 기여하고 싶은 주제 별 소그룹 빌딩

[둘째 주]

- 소그룹 기여 방향성 설정 및 개인별 기여 주제 선정

[셋째 주]

- 각 소그룹 및 개인별 진행사항 공유와 어렵거나 궁금한 점에 대해 QnA 진행
- 온라인 및 오프라인 모각코

[넷째 주]

- 각 소그룹 및 개인별 진행사항 공유와 어렵거나 궁금한 점에 대해 QnA 진행
- 온라인 및 오프라인 모각코

3 컨트리뷰션 운영 방안

9월 (Masters 8.10~11.2)

[첫째 주]

- 각 소그룹 및 개인별 진행사항 공유와 어렵거나 궁금한 점에 대해 QnA 진행
- 온라인 및 오프라인 모각코

[둘째 주]

- 각 소그룹 및 개인별 진행사항 공유와 어렵거나 궁금한 점에 대해 QnA 진행
- 온라인 및 오프라인 모각코

[셋째 주]

- 각 소그룹 및 개인별 진행사항 공유와 어렵거나 궁금한 점에 대해 QnA 진행
- 온라인 및 오프라인 모각코

[넷째 주]

- 각 소그룹 및 개인별 진행사항 공유와 어렵거나 궁금한 점에 대해 QnA 진행
- 온라인 및 오프라인 모각코

10월 (Masters 8.10~11.2)

[첫째 주]

- 각 소그룹 및 개인별 진행사항을 마무리 하는데 집중
- 온라인 및 오프라인 모각코

[둘째 주]

- 서면평가 자료 작성을 위한 기여 목록 정리
- 각 소그룹 및 개인별 진행사항을 마무리 하는데 집중
- 온라인 및 오프라인 모각코
- 서면평가 자료 작성을 위한 기여 목록 최종 확인 및 자료 제출

[셋째 주]

- 최종 성과공유회(발표 평가) 준비

[넷째 주]

- 최종 성과공유회(발표 평가) 준비

3 컨트리뷰션 운영 방안

◎ ONLINE 모임

- 오프라인 위주로 운영이 진행될 예정이지만, 참석이 어려운 멘티들을 위해 오프라인 모임 시 디스코드를 활용해 소통 채널 제공
- 디스코드 채널을 활용해 멘티들이 궁금한 점을 바로바로 질문할 수 있도록 함
- 필요 시 지정된 오프라인 모임 시간 외에 디스코드를 활용해 온라인으로 추가적인 모임 진행 예정

◎ OFFLINE 모임

- 오프라인 모임이 가능한 시간을 사전 조사한 뒤 매 주 마다 오프라인 모임 진행

4

컨트리뷰션 가이드

단계별 컨트리뷰션 커리큘럼



4 컨트리뷰션 가이드

1. 컨트리뷰션을 시작하기 전 준비 과정

[개발환경 구축]

- 가상머신 또는 클라우드 서버 환경을 활용해 Linux 기반의 개발환경 구축

[git 사용법 습득]

- 기본적인 Git 사용법과 관련한 실습
- 좋은 Git 커밋 메시지 작성을 위한 커밋 컨벤션 소개
- 처음 기여하면서 발생할 수 있는 주요 이슈 및 해결 방법 소개
 - 커밋 메시지 작성 시 50/72 룰을 지켰는지 확인
 - 코드 리뷰를 반영한 코드를 다시 PR(Pull Request)에 Push하기

[오픈소스 프로젝트에 기여하는 방법 확인]

- 프로젝트에 기여하기 위해 지켜야 할 사항을 안내하는 CONTRIBUTING.md 확인

4 컨트리뷰션 가이드

2. ufttrace를 이해하기 위한 배경 지식 습득

[ufttrace가 프로그램을 추적하는 방식에 대한 배경지식 습득]

- ufttrace에서 프로그램을 추적하는데 사용하는 기법인 mcount(), PLT Hooking, Dynamic Tracing에 대해 이론적인 내용 분석과 간단한 예제 코드를 사용해 배경지식을 습득할 수 있도록 함
 - mcount() 기반의 Simple Tracer 따라해보기
 - PLT Hooking 기법을 사용해 라이브러리 함수 호출 시 호출이 되었다는 로그를 출력해보기
 - Dynamic Tracing에서 사용하는 함수 패치 원리를 이해하기 위한 예제

4 컨트리뷰션 가이드

3. ufttrace 코드 저장소 및 내부 구조 분석

[ufttrace 코드 저장소 분석]

- ufttrace의 코드 저장소 구조를 분석해 어느 위치에 어떤 파일이 있는지 대략적으로 파악할 수 있도록 함

[ufttrace가 프로그램을 추적하는 부분에 대한 코드 분석]

- mcount() 를 이용한 함수 호출 및 인자 값 추적 부분 분석
- PLT Hooking 을 이용한 라이브러리 함수 호출 추적 부분 분석
- 특정 부분의 코드가 각 아키텍처마다 다르게 구현되어야 하는 이유와 해당 부분의 코드 구현 분석
 - RISC-V 아키텍처 기준으로 분석 예정

4 컨트리뷰션 가이드

4. ufttrace 프로젝트에 기여

[ufttrace 프로젝트에 기여하기 위한 소그룹 빌딩]

- 개인별 관심사에 따라 사전에 지정된 소그룹에 참여하고, 소그룹에 할당된 방향성을 토대로 각자 기여하고자 하는 이슈 선정
 - Python Tracing, Rust Tracing, C/C++ Tracing, Visualization, RISC-V Arch로 소그룹 구성 예정



The Rust
Programming
Language



4 컨트리뷰션 가이드

4. ufttrace 프로젝트에 기여

[각 소그룹 별 방향성]

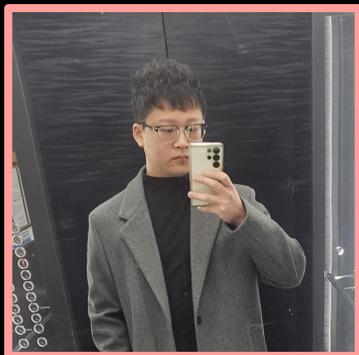
- Python Tracing, Rust Tracing, C/C++ Tracing 팀
 - 각 언어별 Tracing 코드 분석 및 기존 코드 고도화
 - 새로운 TestCase 코드 작성 또는 기존 코드 고도화
 - 각 언어별 Tracing과 관련한 문서 작성 및 기존 문서 고도화
- Visualization 팀
 - Visualization 관련 기존 코드 고도화
- RISC-V Arch 팀
 - Dynamic Tracing 기능이 RISC-V 아키텍처에서 동작할 수 있도록 포팅 및 테스트



멘토 소개

컨트리뷰션 프로젝트팀 멘토단 소개

5 멘토 소개



- **최기철 ★리드**
- (현) 리눅스 시스템 보안 제품 개발자
- (전) 2023 컨트리뷰션 아카데미 uftace 멘티



- **권석민**
- (현) OS 박사과정 재학
- (전) 2023 컨트리뷰션 아카데미 uftace 멘티

2024 오픈소스 컨트리뷰션 아카데미

Open Source Contribution Academy



uftrace

컨트리뷰션에 도전해 보세요!

 THANK YOU 

2024 Open Source Contribution Academy 2024 Open Source Contribution Academy 2024 Open Source Contribution Academy 2024 Open Source Contribution Academy