리눅스 디버깅 방법

Application 디버깅

컴퓨터공학부 시스템프로그래밍
(wonlee@koreatech.ac.kr)

리눅스 디버깅 툴

- ftrace
- strace
- gdb

ftrace

- 커널 함수 콜을 추적
- 커널 디버깅 및 분석
- 옵션 설정을 통해 원하는 정보 필터링 가능 (원하는 프로세스만 추적 등)

strace

- system call 추적
- 어플리케이션 디버깅 및 분석에 활용

GDB

- GNU Debugger - Open Source
- https://www.sourceware.org/gdb/
- 텍스트 기반 디버거
- ddd - 오래된 GUI Front End
- Eclipse, Visual Studio 연동 가능

디버깅을 위한 컴파일 옵션

- 소스레벨 디버깅을 위해서는 디버깅 정보가 필요
- 디버깅 정보 : 소스코드 이름, 줄 번호, 변수 및 함수 심볼 정보, 구조체 정보 등
- ex) gcc -g -o app app.c
- ex) gcc -g -O0 -o app app.c

예제 코드

$ vim debug.c

```c
#include <stdio.h>

static int my_static(void)
{
    printf("static function\n");
}

int my_function(int arg)
{
    return (arg+1);
}

int main(int argc, char **argv)
{
    int i;

    for (i = 0; i < 10; i++);

    printf("hello, this is a message: %d\n", my_function(20));

    my_static();

    return 0;
}

```

컴파일

$ gcc debug.c -o debug

strace 사용

$ ./debug

$ strace ./debug

```
brk(NULL)                                       = 0x561531283000
brk(0x5615312a4000)                             = 0x5615312a4000
write(1, "hello, this is a message: 21\n", 29hello, this is a message: 21
) = 29
write(1, "static function\n", 16static function
)       = 16
exit_group(0)                                   = ?
+++ exited with 0 +++
```

gdb 사용

$ gdb ./debug

(gdb) b main

(gdb) r

(gdb) list main

(gdb) c

(gdb) q

```
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./debug...
(No debugging symbols found in ./debug)
(gdb) b main
Breakpoint 1 at 0x119e
(gdb) r
Starting program: /home/neo/lecture/sp/debug
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, 0x000055555555519e in main ()
(gdb) list main
(gdb) c
Continuing.
hello, this is a message: 21
static function
[Inferior 1 (process 96835) exited normally]
(gdb)
```

디버깅 정보 추가 후 gdb 실행

$ gcc debug.c -o debug -g

$ gdb -q ./debug

```
Reading symbols from ./debug...
(gdb) b main
Breakpoint 1 at 0x11a9: file debug.c, line 17.
(gdb) r
Starting program: /home/neo/lecture/sp/debug
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main (argc=1, argv=0x7fffffffde88) at debug.c:17
17          for (i = 0; i < 10; i++);
(gdb) list main
9       {
10          return (arg+1);
11      }
12
13      int main(int argc, char **argv)
14      {
15          int i;
16
17          for (i = 0; i < 10; i++);
18
(gdb)
19          printf("hello, this is a message: %d\n", my_function(20));
20
21          my_static();
22
23          return 0;
24      }
25
(gdb) c
Continuing.
hello, this is a message: 21
static function
[Inferior 1 (process 96884) exited normally]
(gdb)
```

gdb 사용법 - next, n, nexti, ni

- 다음 코드 실행 (함수 호출 시 함수내 코드는 skip)

(gdb) next

(gdb) n

- 다음 코드 실행 (어셈코드 단위)

(gdb) nexti

(gdb) ni

gdb 사용법 - step, s, stepi, si

- 다음 코드 실행 (함수 호출 시 함수 안의 코드로 점프)

(gdb) step

(gdb) s

- 다음 코드 실행 (어셈코드 단위)

(gdb) stepi

(gdb) si

gdb 사용법 - print, x

- 변수나 주소에 담겨진 값을 출력

```
Breakpoint 1, main (argc=1, argv=0x7fffffffde88) at debug.c:17
17          for (i = 0; i < 10; i++);
(gdb) s
19          printf("hello, this is a message: %d\n", my_function(20));
(gdb) print i
$1 = 10
(gdb) x i
0xa:    Cannot access memory at address 0xa
(gdb) s
my_function (arg=20) at debug.c:10
10          return (arg+1);
(gdb) s
11      }
(gdb) s
__printf (format=0x555555556014 "hello, this is a message: %d\n") at ./stdio-common/printf.c:28
28      ./stdio-common/printf.c: No such file or directory.
(gdb) x format
0x555555556014: 0x6c6c6568
(gdb) x/s format
0x555555556014: "hello, this is a message: %d\n"
(gdb) x/i $pc
=> 0x7ffff7c60770 <__printf>:   endbr64
(gdb) x/i main
   0x555555555196 <main>:       endbr64
```

gdb 사용법 - breakpoints

- breakpoint 설정

(gdb) b symbol

(gdb) b 파일명:줄번호

- breakpoint 확인

(gdb) info breakpoints

- breakpoint 삭제

(gdb) delete breakpoints 1

- breakpoint 켜기/끄기

(gdb) enable breakpoints 1

(gdb) disable breakpoints 1

```
Reading symbols from ./debug...
(gdb) b main
Breakpoint 1 at 0x11a9: file debug.c, line 17.
(gdb) r
Starting program: /home/neo/lecture/sp/debug
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main (argc=1, argv=0x7fffffffde88) at debug.c:17
17          for (i = 0; i < 10; i++);
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00005555555551a9 in main at debug.c:17
        breakpoint already hit 1 time
(gdb) b printf
Breakpoint 2 at 0x7ffff7c60770: file ./stdio-common/printf.c, line 28.
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00005555555551a9 in main at debug.c:17
        breakpoint already hit 1 time
2       breakpoint     keep y   0x00007ffff7c60770 in __printf at ./stdio-common/printf.c:28
(gdb) c
Continuing.

Breakpoint 2, __printf (format=0x555555556014 "hello, this is a message: %d\n") at ./stdio-common/printf.c:28
28      ./stdio-common/printf.c: No such file or directory.
(gdb) disable breakpoints 2
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00005555555551a9 in main at debug.c:17
        breakpoint already hit 1 time
2       breakpoint     keep n   0x00007ffff7c60770 in __printf at ./stdio-common/printf.c:28
        breakpoint already hit 1 time
(gdb) c
Continuing.
hello, this is a message: 21
static function
[Inferior 1 (process 96990) exited normally]
(gdb)
```

gdb 사용법 - backtrace (콜스택 보기)

(gdb) bt

```
Reading symbols from ./debug...
(gdb) b main
Breakpoint 1 at 0x11a9: file debug.c, line 17.
(gdb) r
Starting program: /home/neo/lecture/sp/debug
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main (argc=1, argv=0x7fffffffde88) at debug.c:17
17          for (i = 0; i < 10; i++);
(gdb) b puts
Breakpoint 2 at 0x7ffff7c80ed0: file ./libio/ioputs.c, line 33.
(gdb) c
Continuing.
hello, this is a message: 21

Breakpoint 2, __GI__IO_puts (str=0x555555556004 "static function") at ./libio/ioputs.c:33
33      ./libio/ioputs.c: No such file or directory.
(gdb) bt
#0  __GI__IO_puts (str=0x555555556004 "static function") at ./libio/ioputs.c:33
#1  0x0000555555555180 in my_static () at debug.c:5
#2  0x00005555555551e1 in main (argc=1, argv=0x7fffffffde88) at debug.c:21
(gdb) disas my_static
Dump of assembler code for function my_static:
   0x0000555555555169 <+0>:     endbr64
   0x000055555555516d <+4>:     push   %rbp
   0x000055555555516e <+5>:     mov    %rsp,%rbp
   0x0000555555555171 <+8>:     lea    0xe8c(%rip),%rax        # 0x555555556004
   0x0000555555555178 <+15>:    mov    %rax,%rdi
   0x000055555555517b <+18>:    call   0x555555555060 <puts@plt>
   0x0000555555555180 <+23>:    nop
   0x0000555555555181 <+24>:    pop    %rbp
   0x0000555555555182 <+25>:    ret
End of assembler dump.
(gdb)
```

gdb 사용법 - set (레지스터/변수/메모리 값 변경)

(gdb) set $eax=0x12345678

(gdb) set var i=50

(gdb) set *주소=값

```
(gdb) info reg
rax            0x555555555196      93824992235926
rbx            0x0                 0
rcx            0x555555557db8      93824992247224
rdx            0x7fffffffde98      140737488346776
rsi            0x7fffffffde88      140737488346760
rdi            0x1                 1
rbp            0x7fffffffdd70      0x7fffffffdd70
rsp            0x7fffffffdd50      0x7fffffffdd50
r8             0x7ffff7e1af10      140737352150800
r9             0x7ffff7fc9040      140737353912384
r10            0x7ffff7fc3908      140737353890056
r11            0x7ffff7fde680      140737354000000
r12            0x7fffffffde88      140737488346760
r13            0x555555555196      93824992235926
r14            0x555555557db8      93824992247224
r15            0x7ffff7ffd040      140737354125376
rip            0x5555555551bc      0x5555555551bc <main+38>
eflags         0x202               [ IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) print/x $rax
$3 = 0x555555555196
(gdb) print/x $eax
$4 = 0x55555196
(gdb) set $eax=0
(gdb) print/x $eax
$5 = 0x0
(gdb) print/x $rax
$6 = 0x555500000000
(gdb)
```

gdb 사용법 - display (실행 흐름이 바뀔 때 마다 출력하기)

(gdb) display 변수명

(gdb) display $레지스터

(gdb) display i

(gdb) display $rip

(gdb) info display

(gdb) delete display 1

```
Reading symbols from ./debug...
(gdb) b main
Breakpoint 1 at 0x11a9: file debug.c, line 17.
(gdb) r
Starting program: /home/neo/lecture/sp/debug
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main (argc=1, argv=0x7fffffffde88) at debug.c:17
17          for (i = 0; i < 10; i++);
(gdb) x/x &i
0x7fffffffdd6c: 0x00005555
(gdb) display i
1: i = 21845
(gdb) n
19          printf("hello, this is a message: %d\n", my_function(20));
1: i = 10
(gdb) set *0x7fffffffdd6c=20
(gdb) n
hello, this is a message: 21
21          my_static();
1: i = 20
(gdb) x/x &i
0x7fffffffdd6c: 0x00000014
(gdb)
```

퀴즈 1. 포인터 변경

“Good job~!” 출력하기

```c
#include <stdio.h>

void dont_call(void)
{
    printf("Good job~!\n");
}

void should_call(char *str)
{
    printf("%s\n", str);
}

int main(int argc, char **argv)
{
    void (*func)(char *);

    func = should_call;
    func("no way\n");

    return 0;
}
```

퀴즈 2. 쉘(/bin/sh) 실행하기

퀴즈 1의 바이너리 활용