운영체제론 실습 7주차

CPS LAB

ptrace (1) : 함수가 호출하는 시스템 콜 추적

CHAPTER 1.

ptrace

- ptrace 소개 및 Request 종류
- ptrace 사용법
- ptrace 예제

ptrace (= process trace)

- 한 프로세스(tracer)가 다른 **프로세스(tracee)의 실행을 관찰 및 제어**할 수 있게 하는 시스템 콜
- 피추적자(tracee)의 **메모리**와 **레지스터**를 검사 및 변경할 수 있는 방법 제공

```
#include <sys/ptrace.h>

long ptrace(enum __ptrace_request request, pid_t pid, void *addr, void *data);
```

- enum __ptrace_request request → 요청
- pid_t pid → tracee 프로세스 ID
- void *addr → tracee 내의 주소
- void *data → tracer 내의 주소
  - tracee에게 전달할 data 위치
  - tracee에게 받은 정보를 저장할 위치

사용할 Request 목록

```
long ptrace(enum __ptrace_request request, pid_t pid, void *addr, void *data);
```

Request	설 명
PTRACE_TRACEME	프로세스가 부모에 의해 추적될 것임을 나타냄
PTRACE_PEEKUSER	피추적자의 USER 영역의 오프셋 addr에서 word를 읽음 (레지스터, 프로세스에 대한 기타 정보 포함)
PTRACE_GETREGS	각각 피추적자의 범용 레지스터들이나 부동 소수점 레지스터들을 추적자 내의 주소 data로 복사함
PTRACE_SYSCALL	정지된 피추적 프로세스를 재시작 하되, 다음 번 시스템 호출 진입이나 퇴장에서 또는 한 인스트럭션 실행 후에 피추적자가 멈추도록 함

ptrace 사용법 (1)

- **PTRACE_TRACEME**

```
ptrace(PTRACE_TRACEME, 0, 0, 0);
```

▶ 이 프로세스가 부모에 의해 추적될 것임을 나타냄

- **PTRACE_PEEKUSER**

```
ptrace(PTRACE_PEEKUSER, pid, addr, 0);
```

▶ 피추적자 USER 영역의 오프셋 addr에서 word를 읽음

- **PTRACE_GETREGS**

```
ptrace(PTRACE_GETREGS, pid, 0, &regs);
```

▶ 피추적자의 범용 레지스터들이나 부동 소수점 레지스터들을 추적자 내의 주소 regs(data)에 복사함 register 에 대한 정보는 <sys/user.h> 참고

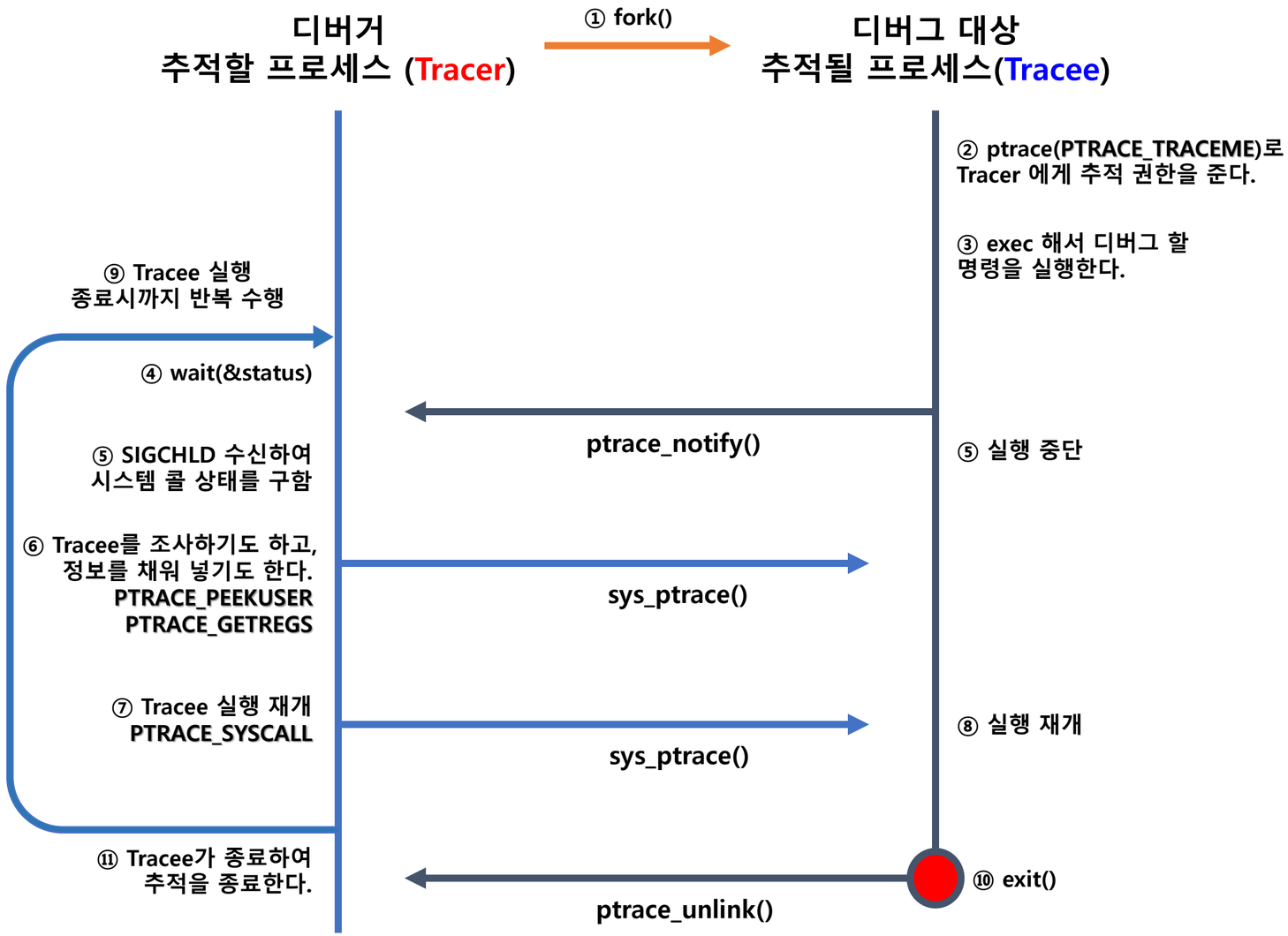
ptrace 사용법 (2)

- **PTRACE_SYSCALL**

```
ptrace(PTRACE_SYSCALL, pid, 0, sig);
```

- ▶ 매 시스템콜 진입/퇴장/인스트럭션 실행에 피추적자가 멈추도록 설정됨
- ▶ sig(data)가 0이 아니면 피추적자에게 보낼 시그널 번호로 해석
- ▶ 0 이면 시그널을 보내지 않음

tracer-tracee flow



예제 1 : 함수가 호출하는 system call 확인

week7/1_which_syscall/which_syscall.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/ptrace.h>
#include <sys/reg.h>
#include <sys/syscall.h>
#include <sys/types.h>
#include <sys/user.h>
#include <sys/wait.h>
#include <unistd.h>

int main() {
  long orig_rax;
  pid_t child;
  int status;
  int input;
  child = fork();
  if (child == 0) {
    ptrace(PTRACE_TRACEME, 0, NULL, NULL);
    raise(SIGSTOP);
    scanf("%d", &input);
    printf("INPUT: %d\n", input);
  } else {
    while (1) {
      wait(&status);
      if (WIFEXITED(status))
        break;
      orig_rax = ptrace(PTRACE_PEEKUSER, child, 8 * ORIG_RAX, NULL);
      printf("SYSCALL NO: %ld\n", orig_rax);
      ptrace(PTRACE_SYSCALL, child, NULL, NULL);
    }
  }
  return 0;
}
```

예제 1 : 실행 결과

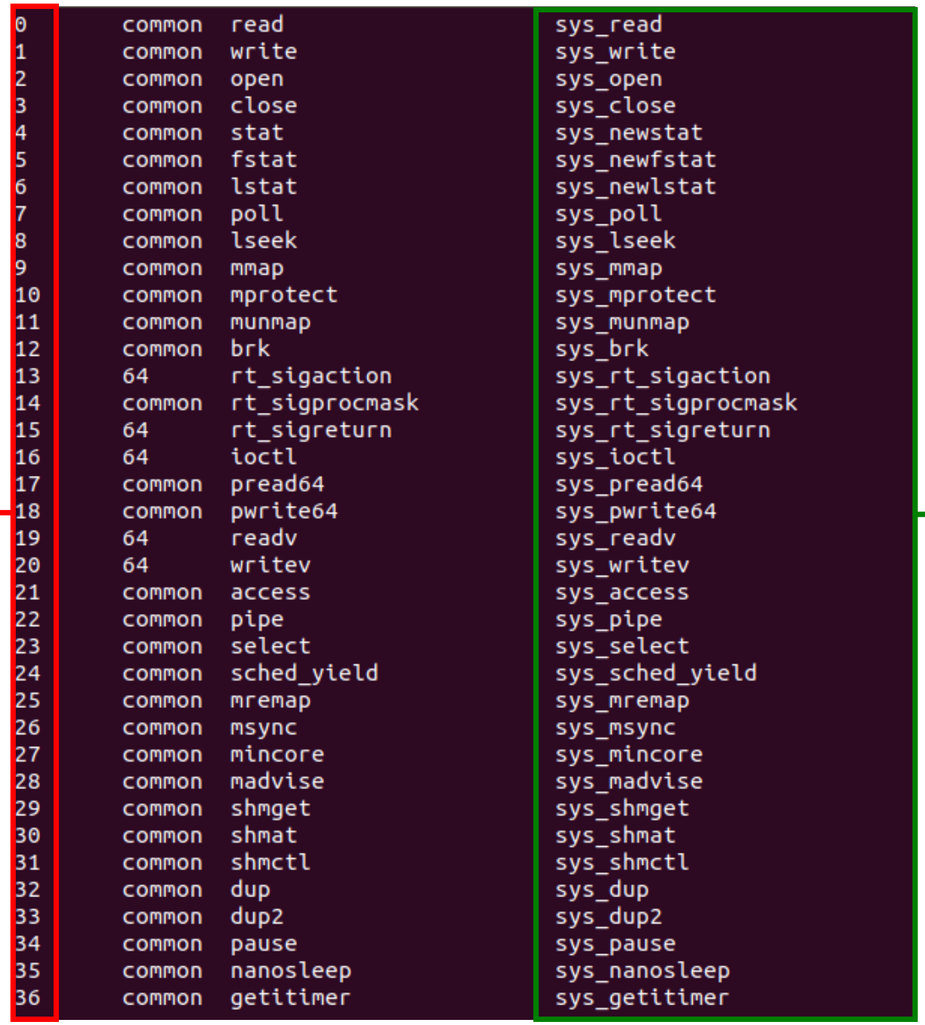
```
os@os-virtual-machine:~/Downloads/1_which_syscall$ ./which_syscall
SYSCALL NO: 234
SYSCALL NO: 14
SYSCALL NO: 14
SYSCALL NO: 5
SYSCALL NO: 5
SYSCALL NO: 12
SYSCALL NO: 12
SYSCALL NO: 12
SYSCALL NO: 12
SYSCALL NO: 0
0
SYSCALL NO: 0
SYSCALL NO: 5
SYSCALL NO: 5
SYSCALL NO: 1
INPUT: 0
SYSCALL NO: 1
SYSCALL NO: 8
SYSCALL NO: 8
SYSCALL NO: 231
```

scanf 함수가 호출하는 system call 중 하나

printf 함수가 호출하는 system call 중 하나

예제 1 : System call 등록 위치

```
$ vi /usr/src/linux-$(uname -r)/arch/x86/entry/syscalls/syscall_64.tbl
```



System call Number ←

→ **System call Name**

예제 1 : user_regs_struct 구조체

```
$ vi /usr/src/linux-$(uname -r)/arch/x86/ 이하 각 경로
```

entry/calling.h

```
#define R15             0*8
#define R14             1*8
#define R13             2*8
#define R12             3*8
#define RBP             4*8
#define RBX             5*8
/* These regs are callee-clobbered. Always saved on kernel entry. */
#define R11             6*8
#define R10             7*8
#define R9              8*8
#define R8              9*8
#define RAX             10*8
#define RCX             11*8
#define RDX             12*8
#define RSI             13*8
#define RDI             14*8
/*
 * On syscall entry, this is syscall#. On CPU exception, this is error code.
 * On hw interrupt, it's IRQ number:
 */
#define ORIG_RAX        15*8
/* Return frame for iretq */
#define RIP             16*8
#define CS              17*8
#define EFLAGS          18*8
#define RSP             19*8
#define SS              20*8
```

구조체 내에서 위치(offset)를 나타내는 매크로 상수

include/asm/user_64.h

0번째 →

15번째 →

```
struct user_regs_struct {
        unsigned long   r15;
        unsigned long   r14;
        unsigned long   r13;
        unsigned long   r12;
        unsigned long   bp;
        unsigned long   bx;
        unsigned long   r11;
        unsigned long   r10;
        unsigned long   r9;
        unsigned long   r8;
        unsigned long   ax;
        unsigned long   cx;
        unsigned long   dx;
        unsigned long   si;
        unsigned long   di;
        unsigned long   orig_ax;
        unsigned long   ip;
        unsigned long   cs;
        unsigned long   flags;
        unsigned long   sp;
        unsigned long   ss;
        unsigned long   fs_base;
        unsigned long   gs_base;
        unsigned long   ds;
        unsigned long   es;
        unsigned long   fs;
        unsigned long   gs;
};
```

참고 : https://sites.uclouvain.be/SystInfo/usr/include/sys/reg.h.html

예제 2 : 프로세스 레지스터 값 한번에 불러오기

week7/2_get_regs/ptrace_regs.c

```c
int main() {
  pid_t child;
  long orig_rax, rax;
  long params[3];
  int status;
  int insyscall = 0;
  struct user_regs_struct regs;
  child = fork();
  if (child == 0) {
    ptrace(PTRACE_TRACEME, 0, NULL, NULL);
    execl("/bin/ls", "ls", NULL);
  } else {
    while (1) {
      wait(&status);
      if (WIFEXITED(status))
        break;
      orig_rax = ptrace(PTRACE_PEEKUSER, child, 8 * ORIG_RAX, NULL);
      if (orig_rax == SYS_write) {
        if (insyscall == 0) {
          insyscall = 1;
          ptrace(PTRACE_GETREGS, child, NULL, &regs);
          printf("Write called with "
                 "%lld %lld %lld\n",
                 regs.rdi, regs.rsi, regs.rdx);
        } else {
          rax = ptrace(PTRACE_PEEKUSER, child, 8 * RAX, NULL);
          printf("Write returned "
                 "with %ld\n",
                 rax);
          insyscall = 0;
        }
      }
      ptrace(PTRACE_SYSCALL, child, NULL, NULL);
    }
  }
  return 0;
}
```

예제 2 : 실행 결과

```
#include <unistd.h>

ssize_t write(int fd, const void *buf, size_t count);
```

```
os@os-virtual-machine:~/Downloads/2_get_regs$ make
gcc -g --save-temps -o ptrace_regs ptrace_regs.c
os@os-virtual-machine:~/Downloads/2_get_regs$ ./ptrace_regs
Write called with 1 94552481731856 82
Makefile  ptrace_regs  ptrace_regs.c  ptrace_regs.i  ptrace_regs.o  ptrace_regs.s
Write returned with 82
```

쓰여진 문자열 길이가 82

쓰여진 바이트 단위
길이가 반환됨

예제 3 : strace를 통해 system call 확인

week7/3_strace/print_something.c

- **strace**는 진단, 디버깅 및 교육용 사용자 공간 유틸리티
- **system call, signal** 및 프로세스 상태 변경 및 프로세스와 Linux 커널 간의 상호 작용을 모니터링하고 변경하는 데 사용
- strace의 작동은 **ptrace**로 알려진 커널 기능으로 가능

```c
#include <stdio.h>
void main(){
        char c = 'A';
        putchar(c);
        return;
}
```

예제 3 : 실행 결과

```
$ strace ./print_something
```

호출되는 system call 목록

```
os@os-virtual-machine:~/Downloads/3_strace$ make
gcc -g --save-temps -o print_something print_something.c
os@os-virtual-machine:~/Downloads/3_strace$ strace ./print_something
execve("./print_something", ["./print_something"], 0x7ffe5d78b340 /* 61 vars */) = 0
brk(NULL)                               = 0x561faac15000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=73868, ...}) = 0
mmap(NULL, 73868, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fb9c3b81000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\240\35\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030928, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fb9c3b7f000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fb9c357a000
mprotect(0x7fb9c3761000, 2097152, PROT_NONE) = 0
mmap(0x7fb9c3961000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x
7fb9c3961000
mmap(0x7fb9c3967000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fb9c3
967000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7fb9c3b804c0) = 0
mprotect(0x7fb9c3961000, 16384, PROT_READ) = 0
mprotect(0x561faac00000, 4096, PROT_READ) = 0
mprotect(0x7fb9c3b94000, 4096, PROT_READ) = 0
munmap(0x7fb9c3b81000, 73868)           = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
brk(NULL)                               = 0x561faac15000
brk(0x561faac36000)                     = 0x561faac36000
write(1, "A", 1A)                        = 1
exit_group(65)                          = ?
+++ exited with 65 +++
```

CHAPTER 2.

Project

- 함수가 호출하는 system call 추적

프로젝트 설명

week7/4_ptrace/my_ptrace.c

- strace와 유사한 기능 구현
- 코드를 실행하며 호출되는 모든 system call 들의 이름을 출력하도록 구현
  - System call 번호를 읽어, 번호에 해당되는 system call 이름을 모두 출력하도록 함

```c
#define syscode_case(x)
  case x:
    return #x

const char *get_syscode(long);

struct user_regs_struct regs;

int main() {
  pid_t child = fork();
  if (child == 0) {
    /* TODO 1 : 자식 프로세스를 추적 가능하도록 설정 */
    execl("/bin/ls", "ls", NULL);
  } else if (child < 0) {
    printf("Fork failed.\n");
  } else {
    int status;
    while (waitpid(child, &status, 0) && !WIFEXITED(status)) {
      /* TODO 2 : 전체 레지스터 값을 가져오기 */
      fprintf(stderr, "[SYSCALL]:%-20s\t%5lld\n", get_syscode(regs.orig_rax),
              regs.orig_rax);
      /* TODO 3 : 매 시스템 콜 마다 실행을 중단하게끔 설정하기 */
    }
  }
  return 0;
}
```

프로젝트 실행 결과

```
[SYSCALL]:SYS_execve              59
[SYSCALL]:SYS_brk                 12
[SYSCALL]:SYS_brk                 12
[SYSCALL]:SYS_access              21
[SYSCALL]:SYS_access              21
[SYSCALL]:SYS_access              21
[SYSCALL]:SYS_access              21
[SYSCALL]:SYS_openat             257
[SYSCALL]:SYS_openat             257
[SYSCALL]:SYS_fstat                5
[SYSCALL]:SYS_fstat                5
[SYSCALL]:SYS_mmap                 9
[SYSCALL]:SYS_mmap                 9
[SYSCALL]:SYS_close                3
[SYSCALL]:SYS_close                3
[SYSCALL]:SYS_access              21
[SYSCALL]:SYS_access              21
[SYSCALL]:SYS_openat             257
[SYSCALL]:SYS_openat             257
[SYSCALL]:SYS_read                 0
[SYSCALL]:SYS_read                 0
[SYSCALL]:SYS_fstat                5
[SYSCALL]:SYS_fstat                5
[SYSCALL]:SYS_mmap                 9
[SYSCALL]:SYS_mmap                 9
[SYSCALL]:SYS_mmap                 9
[SYSCALL]:SYS_mmap                 9
[SYSCALL]:SYS_mprotect            10
[SYSCALL]:SYS_mprotect            10
[SYSCALL]:SYS_mmap                 9
[SYSCALL]:SYS_mmap                 9
[SYSCALL]:SYS_mmap                 9
[SYSCALL]:SYS_mmap                 9
```

중략

• • •

```
[SYSCALL]:SYS_openat              257
[SYSCALL]:SYS_openat              257
[SYSCALL]:SYS_fstat                 5
[SYSCALL]:SYS_fstat                 5
[SYSCALL]:SYS_mmap                  9
[SYSCALL]:SYS_mmap                  9
[SYSCALL]:SYS_close                 3
[SYSCALL]:SYS_close                 3
[SYSCALL]:SYS_ioctl                16
[SYSCALL]:SYS_ioctl                16
[SYSCALL]:SYS_ioctl                16
[SYSCALL]:SYS_ioctl                16
[SYSCALL]:SYS_openat              257
[SYSCALL]:SYS_openat              257
[SYSCALL]:SYS_fstat                 5
[SYSCALL]:SYS_fstat                 5
[SYSCALL]:SYS_getdents             78
[SYSCALL]:SYS_getdents             78
[SYSCALL]:SYS_getdents             78
[SYSCALL]:SYS_getdents             78
[SYSCALL]:SYS_close                 3
[SYSCALL]:SYS_close                 3
[SYSCALL]:SYS_fstat                 5
[SYSCALL]:SYS_fstat                 5
[SYSCALL]:SYS_write                 1
Makefile  my_ptrace  my_ptrace.c  my_ptrace.i  my_ptrace.o  my_ptrace.s
[SYSCALL]:SYS_write                 1
[SYSCALL]:SYS_close                 3
[SYSCALL]:SYS_close                 3
[SYSCALL]:SYS_close                 3
[SYSCALL]:SYS_close                 3
[SYSCALL]:SYS_exit_group          231
```

감사합니다.

CPS LAB