TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 – GJ (박현우)
uc820@naver.com

1. module_init (syscall_hooking_init)을 파헤치기

```
module_init(syscall_hooking_init);
module_exit(syscall_hooking_cleanup);
MODULE_LICENSE("GPL");
```

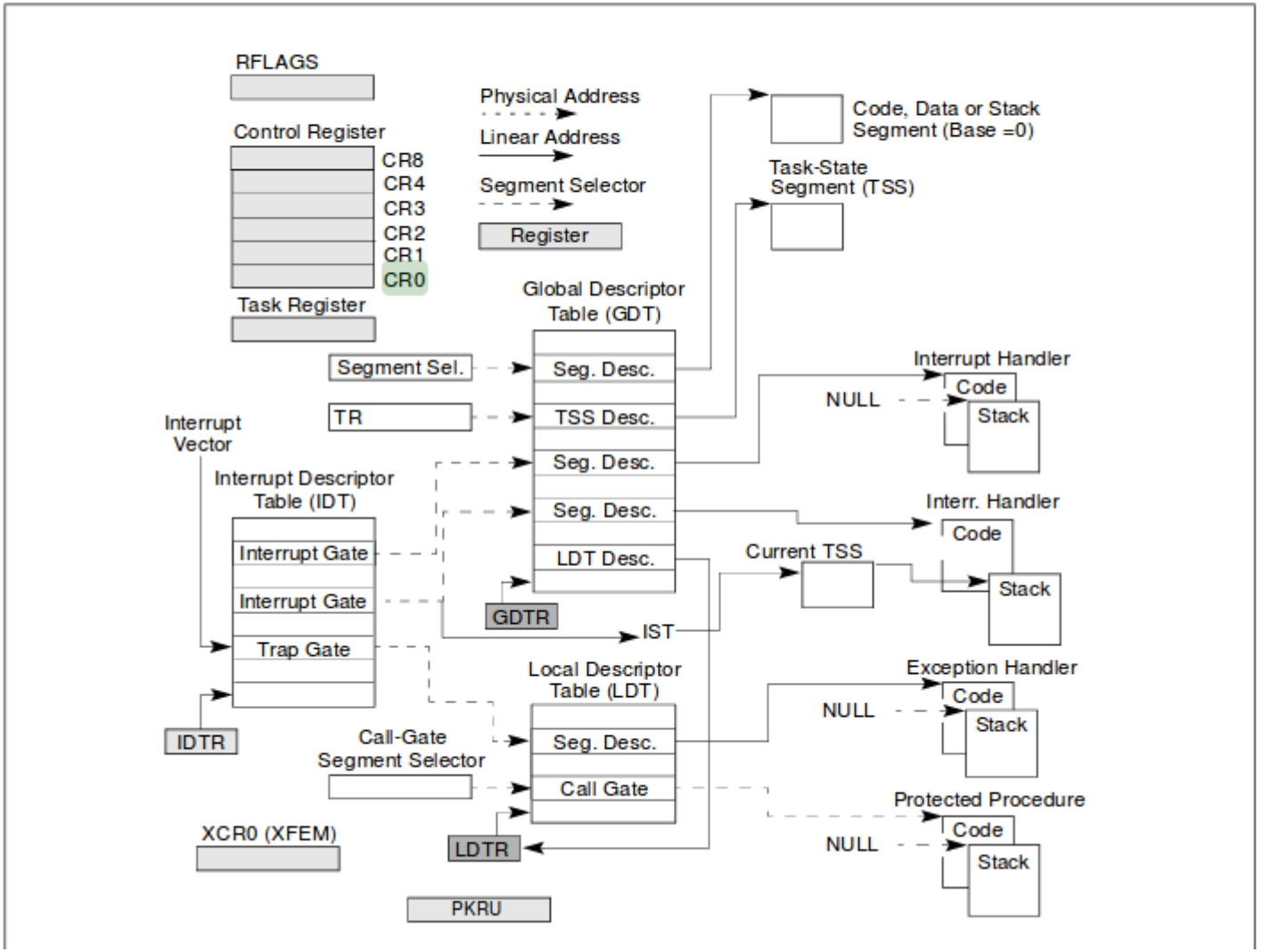
```
int syscall_hooking_init(void)
{
    unsigned long cr0;

    if((sys_call_table = locate_sys_call_table()) == NULL)
    {
        printk("<0>Can't find sys_call_table\n");
        return -1;
    }

    printk("<0>sys_call_table is at[%p]\n", sys_call_table);

    // CR0 레지스터를 읽어옴
    cr0 = read_cr0();
    // Page 쓰기를 허용함
    write_cr0(cr0 & ~0x00010000);

    /* set_memory_rw 라는 심볼을 찾아와서 fixed_set_memory_rw 에 설정함 */
    fixed_set_memory_rw = (void *)kallsyms_lookup_name("set_memory_rw");
    if(!fixed_set_memory_rw)
    {
        printk("<0>Unable to find set_memory_rw symbol\n");
        return 0;
    }

    /* 시스템 콜 테이블이 위치한 물리 메모리에 읽고 쓰기 권한 주기 */
    fixed_set_memory_rw(PAGE_ALIGN((unsigned long)sys_call_table) - PAGE_SIZE, 3);

    orig_call = (void *)sys_call_table[__NR_open];
    sys_call_table[__NR_open] = (void *)sys_our_open;
    write_cr0(cr0);
    printk("<0>Hooking Success!\n");
    return 0;
}
```

2. cro0 는 뭘까?

```
int syscall_hooking_init(void)
{
    unsigned long cr0;

    if((sys_call_table = locate_sys_call_table()) == NULL)
    {
        printk("<0>Can't find sys_call_table\n");
        return -1;
    }

    printk("<0>sys_call_table is at[%p]\n", sys_call_table);

    // CR0 레지스터를 읽어옴
    cr0 = read_cr0();
    // Page 쓰기를 허용함
    write_cr0(cr0 & ~0x00010000);

    /* set_memory_rw 라는 심볼을 찾아와서 fixed_set_memory_rw 에 설정함 */
    fixed_set_memory_rw = (void *)kallsyms_lookup_name("set_memory_rw");
    if(!fixed_set_memory_rw)
    {
        printk("<0>Unable to find set_memory_rw symbol\n");
        return 0;
    }

    /* 시스템 콜 테이블이 위치한 물리 메모리에 읽고 쓰기 권한 주기 */
    fixed_set_memory_rw(PAGE_ALIGN((unsigned long)sys_call_table) - PAGE_SIZE, 3);

    orig_call = (void *)sys_call_table[__NR_open];
    sys_call_table[__NR_open] = (void *)sys_our_open;
    write_cr0(cr0);
    printk("<0>Hooking Success!\n");
    return 0;
}
```

x86(intel)에서 cr0는 시스템 구조의 Control Regitster다.

SYSTEM ARCHITECTURE OVERVIEW



- **CR0** — Contains system control flags that control operating mode and states of the processor.

CR0는 operating mode와 프로세스의 상태를 제어하는 system control flags를 포함한다.

Write Protect (bit 16 of CR0) — When set, inhibits supervisor-level procedures from writing into read-only pages; when clear, allows supervisor-level procedures to write into read-only pages (regardless of the U/S bit setting; see Section 4.1.3 and Section 4.6). This flag facilitates implementation of the copy-on-write method of creating a new process (forking) used by operating systems such as UNIX.

1로 셋팅이 되어 있으면 페이지 쓰기 권한이 막혀 있고 0으로 셋팅되면 페이지 쓰기 권한을 허용한다.

```
// Page 쓰기를 허용함
write_cr0(cr0 & ~0x00010000);
```

를 0으로 셋팅 시켜 page 쓰기 권한을 허용한다..

```
static inline void write_cr0(unsigned long x)
{
    native_write_cr0(x);
}
```

x = cr0의 16번째 비트가 0으로 된 값임.

```
static inline void native_write_cr0(unsigned long val)
{
    asm volatile("mov %0,%%cr0": : "r" (val), "m" (__force_order));
}
```

바뀐 cr0값을 cr0에 다시 셋팅한다.

3. cr0를 셋팅 했으니, kallsyms_lookup_name을 분석해보자.

```
/* Lookup the address for this symbol. Returns 0 if not found. */
/* 심볼에 대한 주소를 검색한다. 발견되지 않으면 리턴 0*/
/* name = set_memory_rw*/
unsigned long kallsyms_lookup_name(const char *name)
{
    char namebuf[KSYM_NAME_LEN];    // KSYM_NAME_LEN 128
    unsigned long i;
    unsigned int off;

    for (i = 0, off = 0; i < kallsyms_num_syms; i++) {
        /*
         0,  nambuf, 128
         */
        off = kallsyms_expand_symbol(off, namebuf, ARRAY_SIZE(namebuf));

        if (strcmp(namebuf, name) == 0)
            return kallsyms_addresses[i];
    }
    return module_kallsyms_lookup_name(name);
}
```

Namebuf, I, off 변수를 선언하고

kayllsyms_num_syms 수 만큼 for문을 돌리면서

kall_sysm_expand_symbol을 사용하여 심볼의 offset 번호를 리턴 받고 namebuf에 찾은 심볼을 저장한

if문으로 name과 심볼이 같으면 kallsyms_addresses를 리턴한다.
심볼을 찾지 못하면 module_kallsyms_lookup_name 모듈에서 해당 심볼을 찾는다.

4. lookup_name을 살 봤으니, kallsyms_expand_symbol함수를 알아보자. (1)

```
/*
 * Expand a compressed symbol data into the resulting uncompressed string,
 * if uncompressed string is too long (>= maxlen), it will be truncated,
 * given the offset to where the symbol is in the compressed stream.
 */
// 압축된 심볼 데이터를 string결과 값으로 바꿨는데, 만약 길이가 maxlen보다 길면 뒷 부분 짤라진다.
/*
 첫번째 인자 0,  nambuf, 128
 */
static unsigned int kallsyms_expand_symbol(unsigned int off,
                       char *result, size_t maxlen)
{
    int len, skipped_first = 0;
    const u8 *tptr, *data;

    /* Get the compressed symbol length from the first symbol byte. */

    /*
       찾으려는 압축된 심볼의 길이를 첫 심볼의 byte수로 부터 얻어온다.
    */

    data = &kallsyms_names[off];
    len = *data;
    data++;

    /*
     * Update the offset to return the offset for the next symbol on
     * the compressed stream.
     */
    /*
     다음 심볼을 찾기 위해서 오프 셋을 갱신하라.
     */
    off += len + 1;
```

4. lookup_name을 살 봤으니, kallsyms_expand_symbol함수를 알아보자. (2)

```
    /*
     * For every byte on the compressed symbol data, copy the table
     * entry for that byte.
     */
    /*
       압축된 심볼 데이터에는 모든 바이트가 올라와 있으니, 해당 비트에 대한 테이블 엔트리를 복
사하라.
     */

    /*
        len 0번 Offset에 대한 길이
        *data =  len +1;

        kayllsyms_token_table 이랑 kayllsyms_token_index는 다른 라이브러리에 있어서 못 찾음.
     */
    while (len) {
        tptr = &kallsyms_token_table[kallsyms_token_index[*data]];
        data++;
        len--;

        while (*tptr) {
            if (skipped_first) {   // 처음에 skipped_first = 0
                if (maxlen <= 1)   // maxlen 128    --> malen이 1이 되면 tail로 이동
                    goto tail;
                *result = *tptr;
                result++;
                maxlen--;
            } else
                skipped_first = 1;
            tptr++;
        }
    }

tail:
    if (maxlen)
        *result = '\0';
    /*
       tail로 왔다는 건 해당 심볼을 못 찾음. 찾은 결과 값을 null로 초기화.

     */
    /* Return to offset to the next symbol.
        다음 심볼을 위한 오프셋을 리턴한다.
     */
    return off;
}
```