운영체제 assignment1

2018320212 김상엽

1-1. 시스템 콜 과정 이해하기

CPU가 사용자 모드에서 C언어 프로그램을 실행 중에, C라이브러리 내의 read함수가 발생되었다면 이 함수는 interrupt를 발생시킨다. Interrupt가 발생하면 CPU는 커널모드에 들어가게 되고, 커널 영역의 메모리에 있는 IDT를 읽어 이를 통해 read() 함수가 발생시킨 interrupt가 system call임을 확인 할 수 있고, (IA32_SYSCALL_VECTOR = 0x80) 처리해야 할 일에 대해 알 수 있다. (entry_INT_32)

os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox

파일 머신 보기 입력 장치 도움말

```
        INTG(X86_TRAP_OLD_MF,           coprocessor_segment_overrun),
        INTG(X86_TRAP_TS,               invalid_TSS),
        INTG(X86_TRAP_NP,               segment_not_present),
        INTG(X86_TRAP_SS,               stack_segment),
        INTG(X86_TRAP_GP,               general_protection),
        INTG(X86_TRAP_SPURIOUS,         spurious_interrupt_bug),
        INTG(X86_TRAP_MF,               coprocessor_error),
        INTG(X86_TRAP_AC,               alignment_check),
        INTG(X86_TRAP_XF,               simd_coprocessor_error),

#ifdef CONFIG_X86_32
        TSKG(X86_TRAP_DF,               GDT_ENTRY_DOUBLEFAULT_TSS),
#else
        INTG(X86_TRAP_DF,               double_fault),
#endif
        INTG(X86_TRAP_DB,               debug),

#ifdef CONFIG_X86_MCE
        INTG(X86_TRAP_MC,               &machine_check),
#endif

        SYSG(X86_TRAP_OF,               overflow),
#if defined(CONFIG_IA32_EMULATION)
        SYSG(IA32_SYSCALL_VECTOR,       entry_INT80_compat),
#elif defined(CONFIG_X86_32)
        SYSG(IA32_SYSCALL_VECTOR,       entry_INT80_32),
#endif
};

/*
 * The APIC and SMP idt entries
 */
static const __initconst struct idt_data apic_idts[] = {
#ifdef CONFIG_SMP
        INTG(RESCHEDULE_VECTOR,         reschedule_interrupt),
        INTG(CALL_FUNCTION_VECTOR,      call_function_interrupt),
                                                                        86,1-8        23%
```

System call interrupt임을 알게 되었으니, system call table에서 read의 system call 번호인 0번째 index에 있는 sys_read()를 호출 하게된다.

os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox

파일 머신 보기 입력 장치 도움말

```
#
# 64-bit system call numbers and entry vectors
#
# The format is:
# <number> <abi> <name> <entry point>
#
# The abi is "common", "64" or "x32" for this file.
#
0       common  read                    sys_read
1       common  write                   sys_write
2       common  open                    sys_open
3       common  close                   sys_close
4       common  stat                    sys_newstat
5       common  fstat                   sys_newfstat
```

이후 syscall 헤더파일에 정의되어 있는 system call 함수가 호출이되어 커널 소스코드를 실행함으로써 시스템 콜 호출이 완료가 된다.

os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox

파일 머신 보기 입력 장치 도움말

```
asmlinkage long sys_utime(char __user *filename,
                          struct utimbuf __user *times);
asmlinkage long sys_utimes(char __user *filename,
                          struct timeval __user *utimes);
asmlinkage long sys_lseek(unsigned int fd, off_t offset,
                          unsigned int whence);
asmlinkage long sys_llseek(unsigned int fd, unsigned long offset_high,
                          unsigned long offset_low, loff_t __user *result,
                          unsigned int whence);
asmlinkage long sys_read(unsigned int fd, char __user *buf, size_t count);
asmlinkage long sys_readahead(int fd, loff_t offset, size_t count);
asmlinkage long sys_readv(unsigned long fd,
                          const struct iovec __user *vec,
```

1-2. 새로운 시스템 콜 추가하기

시스템 콜 번호 할당 (/usr/src/linux/arch/x86/entry/syscalls/syscall_64.tbl)

os-practice1-template-2022 2 [실행 중] - Oracle VM VirtualBox

파일 머신 보기 입력 장치 도움말

```
316     common  renameat2               sys_renameat2
317     common  seccomp                 sys_seccomp
318     common  getrandom               sys_getrandom
319     common  memfd_create            sys_memfd_create
320     common  kexec_file_load         sys_kexec_file_load
321     common  bpf                     sys_bpf
322     64      execveat                sys_execveat/ptregs
323     common  userfaultfd             sys_userfaultfd
324     common  membarrier              sys_membarrier
325     common  mlock2                  sys_mlock2
326     common  copy_file_range         sys_copy_file_range
327     64      preadv2                 sys_preadv2
328     64      pwritev2                sys_pwritev2
329     common  pkey_mprotect           sys_pkey_mprotect
330     common  pkey_alloc              sys_pkey_alloc
331     common  pkey_free               sys_pkey_free
332     common  statx                   sys_statx
333     common  print_stduent_id        sys_print_student_id
#
# x32-specific system call numbers start at 512 to avoid cache impact
# for native 64-bit operation.
#
512     x32     rt_sigaction            compat_sys_rt_sigaction
```

시스템 콜 함수 구현 (/usr/src/linux/include/linux/new_syscall.c)

os-practice1-template-2022 2 [실행 중] - Oracle VM VirtualBox

파일 머신 보기 입력 장치 도움말

```
#include <linux/unistd.h>
#include <linux/errno.h>
#include <linux/kernel.h>
#include <linux/sched.h>

asmlinkage void sys_print_student_id(void)
{
        // TODO

        printk("My student id is 2018320212");
        return 1;
}
EXPORT_SYMBOL_GPL(sys_print_student_id);

~
~
~
~
~
~
```

시스템 콜 함수 선언 (usr/src/linux/kerenel/syscalls.h)

```
asmlinkage long sys_utime(char __user *filename,
                          struct utimbuf __user *times);
asmlinkage long sys_utimes(char __user *filename,
                          struct timeval __user *utimes);
asmlinkage long sys_lseek(unsigned int fd, off_t offset,
                          unsigned int whence);
asmlinkage long sys_llseek(unsigned int fd, unsigned long offset_high,
                          unsigned long offset_low, loff_t __user *result,
                          unsigned int whence);
asmlinkage void sys_print_student_id(void);
asmlinkage long sys_read(unsigned int fd, char __user *buf, size_t count);
asmlinkage long sys_readahead(int fd, loff_t offset, size_t count);
asmlinkage long sys_readv(unsigned long fd,
                          const struct iovec __user *vec,
                          unsigned long vlen);
asmlinkage long sys_write(unsigned int fd, const char __user *buf,
                          size_t count);
```

사용자 영역 프로그램 작성

os-practice1-template-2022 2 [실행 중] - Oracle VM VirtualBox

파일 머신 보기 입력 장치 도움말

```
#include <linux/unistd.h>

int main(void)
{
        // TODO: Write your code here
        syscall(333);

        return 0;
}
~
~
~
~
~
```

작성한 프로그램 실행 후 dmesg 명령어로 결과 확인

os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox

파일 머신 보기 입력 장치 도움말

```
[   12.931138] EXT4-fs (dm-0): re-mounted. Opts: (null)
[   12.975793] Loading iSCSI transport class v2.0-870.
[   13.143683] iscsi: registered transport (tcp)
[   13.243274] systemd-journald[413]: Received request to flush runtime journal from PID 1
[   13.709752] random: crng init done
[   13.902696] iscsi: registered transport (iser)
[   14.583144] snd_intel8x0 0000:00:05.0: white list rate for 1028:0177 is 48000
[   15.332125] EXT4-fs (sda2): mounted filesystem with ordered data mode. Opts: (null)
[   15.614570] audit: type=1400 audit(1649939819.624:2): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="/usr/bin/lxc-start" pid=779 comm="apparmor_parser"
[   15.640076] audit: type=1400 audit(1649939819.648:3): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="/sbin/dhclient" pid=778 comm="apparmor_parser"
[   15.640079] audit: type=1400 audit(1649939819.648:4): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="/usr/lib/NetworkManager/nm-dhcp-client.action" pid=778 comm="apparmor_par
ser"
[   15.640081] audit: type=1400 audit(1649939819.648:5): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="/usr/lib/NetworkManager/nm-dhcp-helper" pid=778 comm="apparmor_parser"
[   15.640082] audit: type=1400 audit(1649939819.648:6): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="/usr/lib/connman/scripts/dhclient-script" pid=778 comm="apparmor_parser"
[   15.657382] audit: type=1400 audit(1649939819.668:7): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="/usr/sbin/tcpdump" pid=783 comm="apparmor_parser"
[   15.658802] audit: type=1400 audit(1649939819.668:8): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="/usr/bin/man" pid=780 comm="apparmor_parser"
[   15.658805] audit: type=1400 audit(1649939819.668:9): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="man_filter" pid=780 comm="apparmor_parser"
[   15.658806] audit: type=1400 audit(1649939819.668:10): apparmor="STATUS" operation="profile_load"
 profile="unconfined" name="man_groff" pid=780 comm="apparmor_parser"
[   15.664648] audit: type=1400 audit(1649939819.676:11): apparmor="STATUS" operation="profile_load"
 profile="unconfined" name="lxc-container-default" pid=777 comm="apparmor_parser"
[   16.647959] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[   16.648903] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
[   16.650649] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[   19.252922] new mount options do not match the existing superblock, will be ignored
[   47.754080] My student id is 2018320212

os-practice:  ~
→
```

Right Control

2-1. 프로세스 퀴즈

Quiz01.

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

const int SEVEN_AND_A_HALF_MILLION_YEARS = 3;
const int A_DAY = 1;

// Allocated in data segment.
static int the_answer = 0;

int main(int argc, char* argv[]){
    // Allocated in stack segment.
    int arthur = 0;

    pid_t pid;

    switch(pid = fork()){
        default :
            // HINT: The parent process should fall into this scope.
            the_answer = 42;
            arthur = 6 * 9;
            sleep(SEVEN_AND_A_HALF_MILLION_YEARS);
            break;
        case  0:
            // HINT: The child process should fall into this scope.
            sleep(A_DAY * 2);
            break;
        case -1:
            printf("WTF?");
            return -1;
            break;
    }

    printf("My pid is %ld (%s)\n", (long)getpid(), pid == 0 ? "child" : "parent");
    printf("The answer to the ultimate question of life the universe and everything is %d.\n", the_a
nswer);
"main.c" 64L, 1561C                                                    41,13         34%
```

```
os-practice:  ~/.../quiz/01
→ ./main
My pid is 3944 (child)
The answer to the ultimate question of life the universe and everything is 0.
But Arthur replied that it was 0.

My pid is 3943 (parent)
The answer to the ultimate question of life the universe and everything is 42.
But Arthur replied that it was 54.
```

Quiz02.

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(int argc, char* argv[]){
    pid_t pid;
    int val = 1;

    printf("The value is %d\n", val);

    pid = fork();

    if(pid > 0){
        // HINT: The parent process should fall into this scope.
        val++;
    } else if(pid ==  0) {
        // HINT: The child process should fall into this scope.
        sleep(1);
        val--;
    } else {
        printf("WTF?");
        return -1;
    }

    printf("The value is %d in %s.\n", val, (pid==0) ? "child" : "parent");

    return 0;
}
```

```
os-practice:  ~/.../quiz/02
→ ./main
The value is 1
The value is 2 in parent.

os-practice:  ~/.../quiz/02
→ The value is 0 in child.
```

Quiz03.

```c
#include <stdio.h>
#include <unistd.h>

int main(int argc, char* argv[]){
    printf("%s executing `ls -l`.\n", "Before");

    // HINT: The /bin/ls -l should be executed.
    execl("/bin/ls", "ls", "-l", NULL);

    printf("%s executing `ls -l`.\n", "After");

    return 0;
}
```

```
os-practice:  ~/.../quiz/03
→ ./main
Before executing `ls -l`.
total 16
-rwxrwxr-x 1 guest guest 8344 Apr 14 00:45 main
-rw-rw-r-- 1 guest guest  608 Apr 14 00:45 main.c
```

Quiz04.

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(int argc, char* argv[]){
    pid_t pid = fork();

    switch (pid)
    {
        default :
            // HINT: The parent process should fall into this scope.
            printf("I'm your father.\n");
            sleep(3);
            break;

        case 0:
            sleep(1);
            // HINT: The child process should fall into this scope.
            printf("I'm sorry, but I'm not Luke. I'm...");
            fflush(stdout);

            sleep(1); // for dramatic effect

            // HINT: The /usr/bin/whoami should be executed.
            execl("/usr/bin/whoami", "whoami", NULL);


        case -1:
            printf("WTF?");
            return -1;
            break;
    }
}
```

```
                                                                    13,8          46%
```

```
os-practice:  ~/.../quiz/04
→ ./main
I'm your father.
I'm sorry, but I'm not Luke. I'm...guest
```

Quiz05.

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(int argc, char* argv[]){
    pid_t pid;
    int status;

    printf("It breaks my heart to see my fellow zealots suffer on the battlefield.\n");
    printf("But what if we dragoons went to their rescue?\n");

    printf("Duh! ");
    fflush(stdout);

    pid = fork();

    if(pid > 0){
        // HINT: The parent process should fall into this scope.
        wait(&status);
        printf("Goon!\n");
    } else if(pid == 0){
        // HINT: The child process should fall into this scope.
        printf("Ra! ");
    } else {
        printf("WTF?");
        return -1;
    }

    return 0;
}
```

```
os-practice:  ~/.../quiz/05
→ ./main
It breaks my heart to see my fellow zealots suffer on the battlefield.
But what if we dragoons went to their rescue?
Duh! Ra! Goon!
```

2-2. 스레드 퀴즈

Quiz01.

```c
#include <unistd.h>
#include <pthread.h>
#include <sys/wait.h>

void* ninja(void* arg){
    printf("Who's there?");
    fflush(stdout);

    pthread_exit("ninja");
}


int main(int argc, char* argv[]){
    pthread_t tid;
    char* from = "";

    printf("Knock knock.\n");

    // HINT: The thread that runs `ninja` should be created.
    int status = pthread_create(&tid, NULL, ninja, NULL);

    if(status != 0){
        printf("WTF?");
        return -1;
    }

    // HINT: The main thread should not be exited until `ninja` has finished.
    pthread_join(tid, (void**) &from);

    // HINT: The variable `from` should not be empty.
    printf(" - from %s\n", from);

    printf("Knuc...kles.\n");

    return 0;
}
                                                                    39,18          47%
```

```
os-practice:  ~/.../quiz/01
→ ./main
Knock knock.
Who's there? - from ninja
Knuc...kles.
```

Quiz02.

```c
int main(int argc, char* argv[]){
    static int main_static;

    pthread_t tids[NUM_THREADS];
    int status;

    printf("global\t\tmain\t\tthread\t\tthread-static\n");
    print_addr(&global, &main, 0, 0);


    for(int i = 0; i < NUM_THREADS; i++){
        // HINT: The thread that runs `worker` should be created.
        // HINT: The address of variable `main_static` should be passed
        //       when thread created.
        // HINT: Each thread descriptor should be stored appropriately.
        status = pthread_create(&tids[i], NULL, worker, (void*)&(main_static));

        if(status != 0){
            printf("WTF?");
            return -1;
        }
    }

    // HINT: The main thread should not be exited until all `worker`s have finished.
    for(int i=0 ;i < NUM_THREADS; i++){
        pthread_join(tids[i], NULL);
    }

    return 0;
}
```

```
os-practice:  ~/.../quiz/02
→ ./main
global          main            thread          thread-static
0x55f46b88b014  0x55f46b68a89f  (nil)   (nil)
0x55f46b88b014  0x55f46b88b01c  0x7f3257928ee4  0x55f46b88b018
0x55f46b88b014  0x55f46b88b01c  0x7f3257127ee4  0x55f46b88b018
0x55f46b88b014  0x55f46b88b01c  0x7f3256926ee4  0x55f46b88b018
```

Quiz03.

```c
int main(int argc, char* argv[]){
    pthread_t tids[NUM_THREADS];
    int status;
    int progress = 0;

    for(int i = 0; i < NUM_THREADS; i++){
        // HINT: The thread that runs `worker` should be created.
        // HINT: The address of variable `i` should be passed when thread created.
        // HINT: Each thread descriptor should be stored appropriately.
        status = pthread_create(&tids[i], NULL, worker, (void*) &i);

        if(status != 0){
            printf("WTF?");
            return -1;
        }
    }

    // HINT: The main thread should not be exited until all `worker`s have finished.
    for(int i = 0 ; i < NUM_THREADS; i++){
        pthread_join(tids[i], (void**) &progress);
        // HINT: The variable `progress` should not be 0.
        printf("\r%d ", progress);

        fflush(stdout);
        usleep(10*1000); // 10ms
    }

    printf("\nexpectd: %d\n", NUM_THREADS * NUM_TASKS);
    printf("result: %d\n", cnt);

    return 0;
}
```

```
os-practice:  ~/.../quiz/03
→ ./main
9999999
expectd: 10000000
result: 10000000

os-practice:  ~/.../quiz/03
```