운영체제 실습 1 보고서

2017320208 컴퓨터학과
이연준

과제1-1.

시스템 콜은 커널에서 제공하는 서비스에 대해, 응용 프로그램의 요청에 따라 커널에 접근하기 위한 인터페이스이다. read함수는 C 라이브러리에 정의되어 있는 함수로, 리눅스에 구현된 C 라이브러리는 GNU C 라이브러리이다.

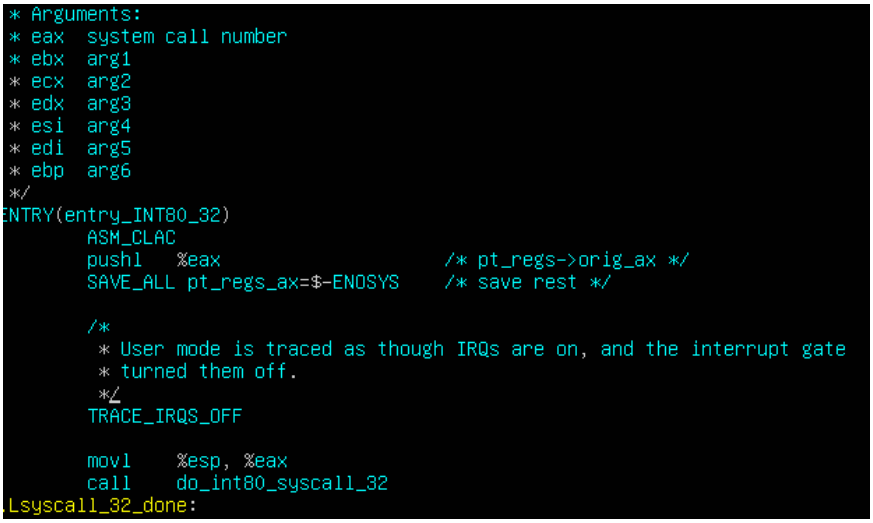
운영체제에서 다른 프로그램의 실행을 방해하거나 프로그램 간의 충돌을 일으키는 문제를 막기 위해, 하드웨어 적인 보안을 유지하기 위해 사용자 모드와 커널 모드를 제공하는데, 사용자 모드는 사용자에게 허용된 영역에만 접근할 수 있고 커널 모드는 모든 영역에 접근이 가능하다.

인터럽트는 다양한 예외상황이 발생하여 처리가 필요한 경우, 프로세서에게 알려 프로세서가 인지하여 처리할 수 있도록 하는 것이다. 인터럽트는 외부 인터럽트와 트랩으로 구분할 수 있는데, 외부 인터럽트는 키보드 입력과 같은 input signal, 즉 프로세서 외부에서 인터럽트를 발생시키는 종류이고 트랩은 소프트웨어에서 발생시키는 종류이다.

C 프로그램을 실행시키는 중에 read함수가 발생되었다면 현재 실행되는 프로그램이 사용자의 프로그램이므로 사용자 모드에서 실행이 된다.

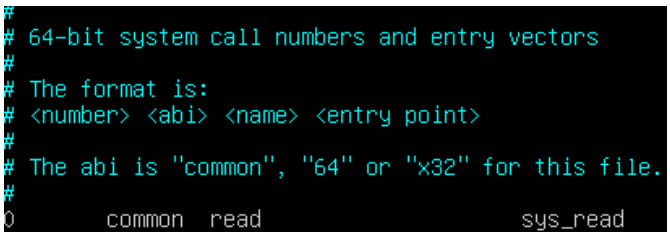
read 함수가 호출되었을 때 바로 시스템 콜을 호출하는 것이 아닌, c언어에서 사용하는 GNU C library 안에 있는 read 함수의 코드를 호출하여 실행한다. 이때, 0x80(decimal로 128)번의 인터럽트를 발생시키는데 커널의 메모리 안에 있는 IDT(Interrupt Descriptor Table)에서 128번 index에 있는 , 즉 0x80 offset을 가지는 인터럽트가 무엇인지 참조하게 된다. IDT는 커널 메모리 안에 있는 인터럽트 벡터 테이블을 구현하기 위해 사용되는 데이터 구조체로 프로세서가 인터럽트와 예외에 대한 정확한 반응을 결정하기 위해 사용된다. 여기서 IDT가 커널 영역의 메모리에 있기 때문에 여기에 접근하기 위해 CPU는 커널 모드를 사용하게 된다. IDT의 0x80에는 시스템 콜에 해당하는 syscall이 위치해있다. 따라서 system call에 대해서 다룬다는 것을 알 수 있다.

IDT의 0x80번 index에 syscall이 있는 것을 확인할 수 있다



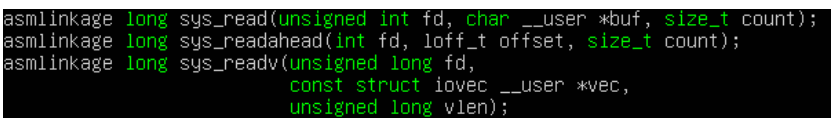
eax에 system call number가 매개변수로 들어가는 것을 확인할 수 있다

위의 그림에서 매개변수들이 정리되어 있는데 eax에 system call number가 매개변수로 전달되는 것을 볼 수 있다. read함수 내부적으로 eax에 전달되는 값인 0이 system call number가 된다. system call number(시스템 콜 번호)는 system call table(시스템 콜 테이블)에서 각 시스템 콜에 부여된 번호이다. 시스템 콜 테이블은 시스템 콜 함수들의 목록이 있는 테이블로, 형식은 <number> <abi> <name> <entry point>로 되어 있는데 number는 시스템 콜 번호가, abi에는 64, x32, common 중에 하나가 들어가고, name에는 함수의 이름, entry point는 실제로 작동하는 함수의 이름이 들어간다. 여기서 시스템 콜 번호를 이용해 시스템 콜 테이블을 참조하여 실제로 작동하는 함수가 무엇인지 확인할 수 있다.



system call table의 0번에 sys_read가 있는 모습

위 그림에서 시스템 콜 번호 0번에 sys_read가 있는 것을 확인할 수 있고 sys_read에 대한 함수의 정의는 헤더파일에서 찾아볼 수 있다.



syscall.h 안에 sys_read에 대한 헤더 파일이 들어있는 모습

요약하면 read함수가 호출 되면 IDT를 참조하여 그 함수가 system call이라는 것을 알 수 있고 시스템 콜 번호를 이용하여 시스템 콜 테이블에서 0번 인덱스를 참조하여 sys_read함수를 실행함으로써 시스템 콜 과정이 끝난다.

과제1-2.

과정1. 시스템 콜 번호 할당

```
333     common  print_student_id        sys_print_student_id
```

syscall_64.tbl 파일에서 위와 같이 333번에 시스템 콜 번호를 할당하였다.

과정2. 시스템 콜 함수 구현

```
#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 2017320208\n");
}
EXPORT_SYMBOL_GPL(sys_print_student_id);
```

new_syscall.c 파일에 학번을 출력하는 sys_print_student_id라는 시스템 콜 함수를 구현하였다.

과정3. 시스템 콜 함수 선언

```
asmlinkage void sys_print_student_id(void);
```

syscalls.h 헤더파일에 시스템 콜 함수를 선언하였다.

과정4. 사용자 영역 프로그램 작성

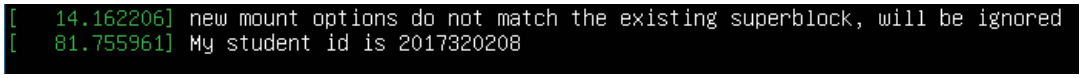
```
#include <linux/unistd.h>

int main(void)
{
        // TODO: Write your code here
        syscall(333);
        return 0;
}
```

/home/guest에 있는 사용자 영역 프로그램 assignment.c을
sys_print_student_id가 333번 systemcall이므로 위와 같이 작성하였다.

과정5. 작성한 프로그램 실행

cd /home/guest 명령어를 통해 해당 경로로 이동한 뒤,
gcc -o assignment assignment.c 명령어로 assignment.c를 컴파일 했다.
./assignment 명령어를 통해 컴파일된 프로그램을 실행하고 dmesg 명령어로
출력된 내용을 확인할 수 있다.

과정6. dmesg 명령어로 결과 확인



과제 2부터 다음페이지에 정리

과제2-1.

퀴즈01

```c
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;
    }
```

퀴즈 1 코드

```
os-practice:  ~/.../quiz/01
→ gcc -o quiz1 main.c

os-practice:  ~/.../quiz/01
→ ./quiz1
My pid is 3872 (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 3871 (parent)
The answer to the ultimate question of life the universe and everything is 42.
But Arthur replied that it was 54.
```

퀴즈 1 실행 결과

퀴즈02

```c
    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;
```

퀴즈 2 코드

```
os-practice:  ~/.../quiz/02
→ ./quiz2
The value is 1
The value is 2 in parent.

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

퀴즈 2 실행 결과

퀴즈03

```c
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;
}
```

퀴즈 3 코드

```
os-practice:  ~/.../quiz/03
→ gcc -o quiz3 main.c

os-practice:  ~/.../quiz/03
→ ./quiz3
Before executing `ls -l`.
total 16
-rw-rw-r-- 1 guest guest  607 Apr 16 12:07 main.c
-rwxrwxr-x 1 guest guest 8344 Apr 16 12:07 quiz3
```

퀴즈 3 실행 결과

퀴즈04

```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;
```

퀴즈 4 코드

```
os-practice:  ~/.../quiz/04
→ gcc -o quiz4 main.c

os-practice:  ~/.../quiz/04
→ ./quiz4
I'm your father.
I'm sorry, but I'm not Luke. I'm...guest
```

퀴즈 4 실행 결과

퀴즈05

```c
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;
```

퀴즈 5 코드

```
os-practice:  ~/.../quiz/05
→ gcc -o quiz5 main.c

os-practice:  ~/.../quiz/05
→ ./quiz5
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!
```

퀴즈 5 실행 결과

과제2-2.

퀴즈01

```
    // 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;
}
```

퀴즈 1 코드

```
os-practice:  ~/.../quiz/01
→ gcc -o quiz1 main.c -pthread

os-practice:  ~/.../quiz/01
→ ./quiz1
Knock knock.
Who's there? - from ninja
Knuc...kles.
```

퀴즈 1 실행 결과

퀴즈02

```c
    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_exit(NULL);
    }
```

퀴즈 2 코드

```
os-practice:  ~/.../quiz/02
→ gcc -o quiz2 main.c -pthread

os-practice:  ~/.../quiz/02
→ ./quiz2
global          main            thread          thread-static
0x56546ff4a014  0x56546fd4984f  (nil)    (nil)
0x56546ff4a014  0x56546ff4a01c  0x7f4344e7bee4  0x56546ff4a018
0x56546ff4a014  0x56546ff4a01c  0x7f434467aee4  0x56546ff4a018
0x56546ff4a014  0x56546ff4a01c  0x7f4343e79ee4  0x56546ff4a018
```

퀴즈 2 실행 결과

퀴즈03

```
    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;
```

퀴즈 3 코드

```
os-practice:  ~/.../quiz/03
→ gcc -o quiz3 main.c -pthread
main.c: In function ‘worker’:
main.c:24:31: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
     stick_this_thread_to_core((int)arg);
                               ^
main.c:31:18: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
     pthread_exit((void*)progress);
                  ^


os-practice:  ~/.../quiz/03
→ ./quiz3
792514
expectd: 10000000
result: 6190298

os-practice:  ~/.../quiz/03
→ ./quiz3
9999999
expectd: 10000000
result: 10000000
```

퀴즈 3 실행 결과