운영체제 1차 과제

<Kernel System call의 구현과 이해>

2021320011 컴퓨터학과 유지연

제출일: 2025년 4월 15일 (화)

Free day 사용일수: 0일

1. 개발환경

- OS: Window 10 - Virtual Box 7.0.20
- Ububtu 18.04.2 - Linux 4.20.11

```
jiyeon011@jiyeon011-VirtualBox:~$ uname -r
4.20.11.jiyeon
```

2. 리눅스의 시스템 콜

CPU는 사용자 프로그램이 하드웨어에 직접 접근하는 것을 막기 위해 user mode, kernel mode 두 가지의 모드를 제공한다. user mode에서는 하드웨어를 직접 제어할 수 없기 때문에 kernel에 접근 요청이 필요한데 이 때 사용되는 개념이 '시스템 콜'이다.

시스템 콜은 일반적으로 시스템 콜 번호와 시스템 콜에서 필요한 인자로 구성된다. 시스템 콜 번호는 커널 내의 시스템 콜 테이블에서 해당하는 커널 함수를 찾는데 사용되며, 시스템 콜 테이블에는 모든 시스템 콜의 커널 함수에 대한 정보가 담겨있다.

프로세스가 원하는 시스템 콜을 수행하면 해당 시스템 콜이 프로세스를 대신하여 커널에 진입한다. 커널이 진입한 시스템 콜은 시스템 콜 테이블에서 해당하는 커널 함수를 찾아 실행하고 그 결과를 프로세스에 돌려준다. 이 과정에서 user mode와 kernel mode간의 mode switching이 발생하지만 이 전환은 프로세스는 인지하지 못한다. 이러한 일련의 과정들 속에서 사용자 프로그램에서는 간접적으로 하드웨어에 접근할 수 있게 된다.

3. 코드 설명

3-1. syscall_64.tbl

```
#jiyeon add system call
335     common  jiyeon_enqueue          __x64_sys_jiyeon_enqueue
336     common  jiyeon_dequeue          __x64_sys_jiyeon_dequeue
```

syscall_64.tbl 에는 시스템 콜과 그에 해당하는 시스템 콜 번호가 저장되어 있다. 이번 과제에서 구현할 시스템 콜 함수는 enqueue, dequeue 두 가지이므로 각각에 335, 336번의 시스템 콜 번호를 할당하였다.

3-2. syscalls.h

새롭게 정의할 시스템 콜 함수의 prototype을 정의하는 파일이다. enqueue 함수의 경우 push할 int 값 한 개를 인자로 받고 반환값은 void이고, dequeue 함수의 경우 인자는 없고 반환값은 pop한 int값이 된다.

```
/*jiyeon*/
asmlinkage void sys_jiyeon_enqueue(int);
asmlinkage int sys_jiyeon_dequeue(void);
```

3-3. my_queue_syscall.c

queue 역할로서 int형 배열을 전역변수로 선언하고, hd, tl 이라는 변수로 queue의 front, rear index를 관리하였다. 이때 tl는 현재 queue의 rear값이 담긴 index의 다음 index로 설정하여 enqueue가 수행되었을 때 값이 저장되어야 할 index 값을 의미한다.

```
int queue[100];
int hd = 0; //index of queue's head(front)
int tl = 0; //index of queue's tail
```

3-3-1. enqueue

enqueue 함수에서 고려해야할 사항은 다음과 같다.

- queue는 int 변수를 저장함
- push 할 때 해당 값이 queue에 존재하는지 확인하고 이미 있다면 push 하지 않음

```
SYSCALL_DEFINE1(jiyeon_enqueue, int, a){
    bool check = true;
    int i;

    for (i = hd; i < tl; i++) {
        if (queue[i] == a) {
            check = false; //a already exits
            break;
        }
    }

    if (check) queue[tl++] = a;
    //enqueue(push) a and add one to tl (move tail index to the right

    //print result using printk
    printk("Queue Front---------------\n");
    for (i = hd; i < tl; i++) printk("%d\n", queue[i]);
    printk("Queue Rear---------------\n");

}
```

for문을 통해 현재 queue에 넣고자 하는 값이 이미 존재하는지를 체크하고(check) 존재하지 않는 경우에만 queue에 값을 넣어준다. 이후에는 printk문을 통해 현재 queue상태를 출력한다.

3-3-2. dequeue

dequeue함수에서 고려해야 할 것은 다음과 같다.

- 현재 queue의 원소 중 가장 처음에 들어온 것을 제거한 후 그 값을 return

```
SYSCALL_DEFINE0(jiyeon_dequeue){
    int head = -1;
    int i;

    if (hd != tl) head = queue[hd++]; //pop queue's head value

    //print result using printk
    printk("Queue Front---------------\n");
    for (i = hd; i < tl; i++) printk("%d\n", queue[i]);
    printk("Queue Rear---------------\n");

    return head;
}
```

hd와 tl의 값이 같은 경우는 empty queue라는 뜻이므로 pop이 진행되지 않는다. pop의 수행은 hd값을 1 증가시켜 주는 것으로 구현하였다. 이 후에는 enqueue함수와 동일한 queue 상태 출력 과정을 거친다.

3-4. call_my_queue.c

직접 선언한 시스템 콜을 호출하기 위한 application 코드이다. 각 함수에 해당하는 시스템 콜 번호인 335, 336을 #define을 통해 선언한 수 syscall함수를 통해 이를 호출한다.

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

#define jiyeon_enqueue_syscall 335
#define jiyeon_dequeue_syscall 336

int main(){
    int r;

    r = syscall(jiyeon_enqueue_syscall, 1);
    printf("Enqueue: %d\n", 1);
```

3-5. Makefile

kernel make시에 my_queue_syscall가 포함되도록 코드를 수정한다.

```
obj-y     = fork.o exec_domain.o panic.o \
            cpu.o exit.o softirq.o resource.o \
            sysctl.o sysctl_binary.o capability.o ptrace.o user.o \
            signal.o sys.o umh.o workqueue.o pid.o task_work.o \
            extable.o params.o \
            kthread.o sys_ni.o nsproxy.o \
            notifier.o ksysfs.o cred.o reboot.o \
            async.o range.o smpboot.o ucount.o my_queue_syscall.o
```

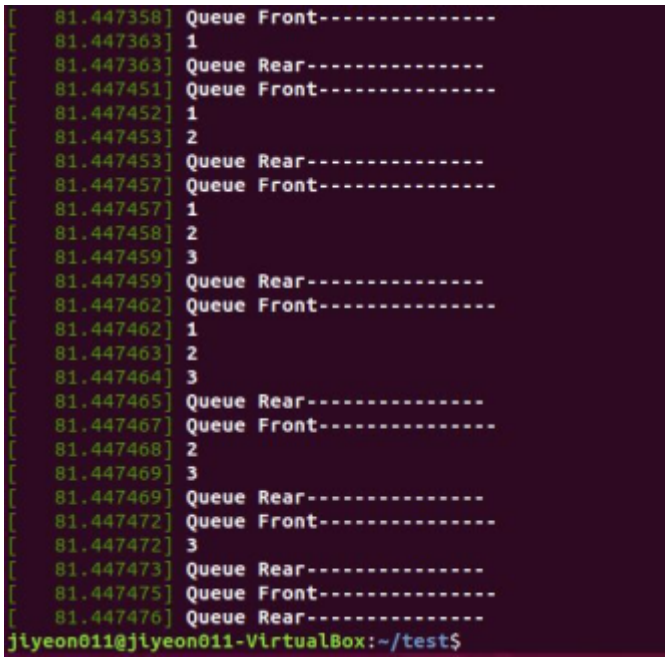
4. 실행 결과

4-1. 응용프로그램 출력

```
jiyeon011@jiyeon011-VirtualBox:~/test$ gcc call_my_queue.c -o call_my_queue
jiyeon011@jiyeon011-VirtualBox:~/test$ ./call_my_queue
Enqueue: 1
Enqueue : 2
Enqueue : 3
Enqueue : 3
Dequeue : 1
Dequeue : 2
Dequeue : 3
```

gcc 명령어를 통해 call_my_queue.c 파일을 컴파일 한 후 call_my_queue라는 이름의 실행파일을 만든다. 만들어진 실행파일은 ./call_my_queue 명령어를 통해 실행할 수 있으며 실행 결과는 위와 같이 나온다.

4-2. 커널 로그 출력 (dmesg)



dmesg 명령어를 통해 printk로 작성한 커널로그를 확인할 수 있다. 이는 enqueue, dequeue 수행 후 현재 queue의 상태를 출력하는 커널로그이다.

5. 문제점과 해결방법

my_queue_syscall.c를 작성하는 과정에서 아래와 같은 for문을 사용하였으나 오류가 발생하였다.

```
for (int i = hd; i< tl; i++);
error: 'for' loop initial declarations are only allowed in C99 or C11 mode
```

이는 make의 컴파일 버전에서는 지원하지 않는 반복문 문법이기 때문에 발생한 오류이다. 반복문의 조건 내부에서 변수를 선언하는 것이 지원되지 않으므로 아래와 같이 수정하였다.

```
int i; //for문 외부에서 변수 선언
for (i = hd; i < tl; i++);
```