하과: 컴퓨터학과

학번: 2022320072

이름: 무함마드 하피주딘 아스라프

제출 날짜: 4/16/2025

Freeday 사용 일수: 1/5, 나머지 4

개발환경: Ubuntu 18.04.02 (64 bit), Linux kernel 4.20.11 (가상머신 환경)

리녹스의 시스템 콜:

시스템 콜은 사용자 모드에서 실행 중인 프로그램이 커널 모드로 진입하여 커널의 서비스를 요청할 수 있도록 해주는 인터페이스입니다. 일반적으로 사용자 프로그램은 먼저 C 라이브러리의 래퍼(wrapper) 함수를 호출하는데, 이 함수는 운영 체제의 호출 규약에 따라 시스템 콜 번호와 매개변수를 레지스터나 스택에 저장한 후, syscall 이나 int 0x80 과 같은 트랩 명령어를 실행하여 커널 모드로 전환합니다. 커널 모드에서는 시스템 콜 핸들러가 시스템 콜 번호를 가져와, 이를 인덱스로 사용해 해당 번호에 대응하는 커널 함수를 찾는 테이블을 참조합니다. 해당 커널 함수는 전달받은 매개변수를 이용해 요청을 처리하고, 그 결과값이나 오류 코드를 레지스터를 통해 사용자 프로그램에 다시 전달함으로써 시스템 콜 사이클을 마칩니다.

수정 및 작성한 부분:

제 코드에서는 사용자 정의 시스템 콜을 정의했습니다:

1) my_enqueue_syscall
2) my_dequeue_syscall

그리고 syscall 번호를 지정하여 syscall 테이블(syscall_64.tbl)에 추가했습니다:

1) MY_ENQUEUE_SYSCALL (335)
2) MY_DEQUEUE_SYSCALL (336)

실행 시 커널 모드로 전환된 후, 커널은 전달된 시스템 콜 번호를 기반으로 해당 함수를 찾아 실행하게 됩니다.

syscalls.h 파일에 시스템 콜을 선언했습니다:

1) asmlinkage int sys_oslab_enqueue(int);
2) asmlinkage int sys_oslab_dequeue(void);

이 두 줄의 선언은 커널에 새로 추가된 함수들의 존재를 인식시키기 위함이며, 시스템 콜 핸들러가 정의된 매개변수 전달 방식에 따라 해당 함수들을 정확하게 호출할 수 있도록 도와줍니다.

my_queue_syscall 함수의 코드는 함수의 동작 로직을 정의하며, 주로 값을 큐에 삽입하고, dequeue 시스템 콜이 호출될 경우 해당 값을 제거하는 방식으로 동작합니다:

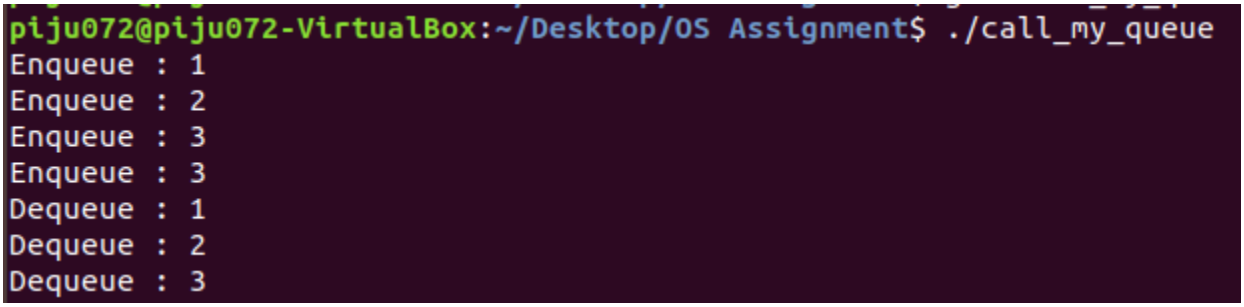
```c
SYSCALL_DEFINE1(oslab_enqueue, int, a)
{
    int i;

    for (i = 0; i < count; i++) {
        if (my_queue[(front + i) % QUEUE_SIZE] == a) {
            printk(KERN_INFO "[Error] - Already existing value");
            return -1;
        }
    }

    if (count == QUEUE_SIZE) {
        printk(KERN_INFO "[Error] - Queue is full\n");
        return -1;
    }

    my_queue[rear] = a;
    rear = (rear + 1) % QUEUE_SIZE;
    count++;

    printk(KERN_INFO "[System call] oslab_enqueue(); -----\n");
    printk(KERN_INFO "Queue Front----------------------\n");

    for (i = 0; i < count; i++) {
        int idx = (front + i) % QUEUE_SIZE;
        printk(KERN_INFO "%d\n", my_queue[idx]);
    }

    printk(KERN_INFO "Queue Rear-----------------------\n");

    return 0;
}
```

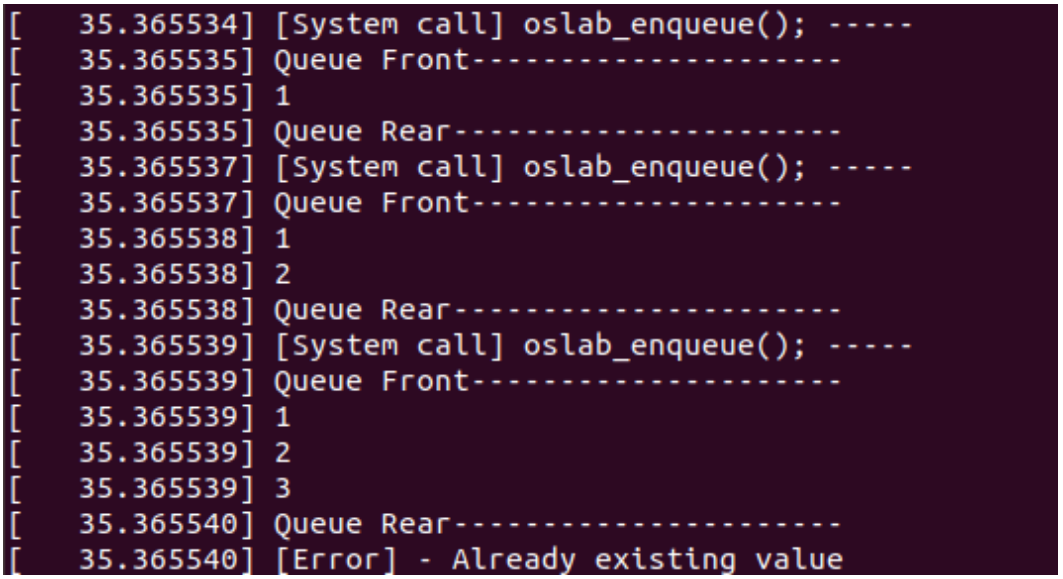
```c
SYSCALL_DEFINE0(oslab_dequeue)
{
    int val, i;

    if (count == 0) {
        printk(KERN_INFO "[Error] - Queue is empty\n");
        return -1;
    }

    val = my_queue[front];
    front = (front + 1) % QUEUE_SIZE;
    count--;

    printk(KERN_INFO "[System call] oslab_dequeue(); -----\n");
    printk(KERN_INFO "Queue Front----------------------\n");
    for (i = 0; i < count; i++) {
        int idx = (front + i) % QUEUE_SIZE;
        printk(KERN_INFO "%d\n", my_queue[idx]);
    }
    printk(KERN_INFO "Queue Rear-----------------------\n");

    return val;
}
```

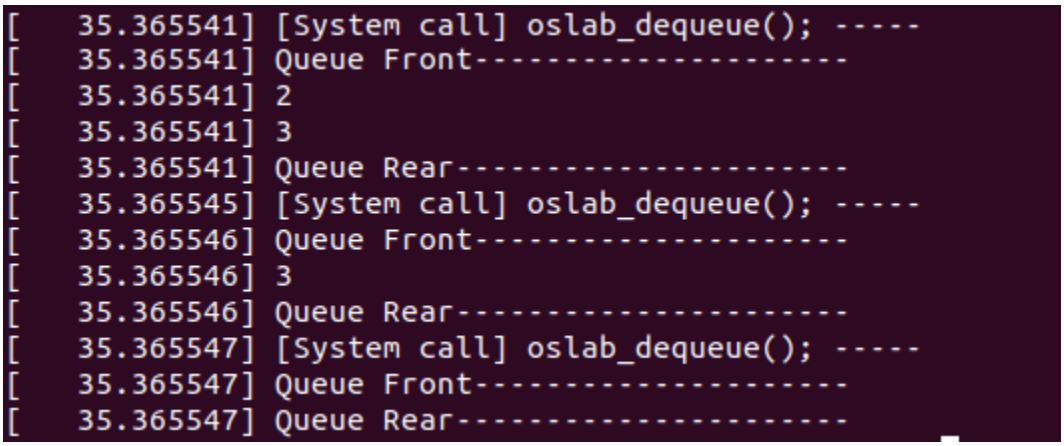
Makefile 에 my_queue_syscall.o 를 추가한 것은 커널 빌드 시스템이 해당 C 소스 파일을 컴파일하고 링크하여 최종 커널 이미지에 포함시키기 위함입니다. 이 과정이 누락되면, 시스템 콜은 커널의 일부로 인식되지 않아 작동하지 않습니다.

사용자 모드에서 작성한 소스 코드는 제가 커널에 구현한 시스템 콜 함수를 호출하며, 출력 결과를 통해 해당 함수가 커널 내에서 올바르게 구현되고 정상적으로 작동함을 확인할 수 있습니다.

실행 결과 스냅샷:

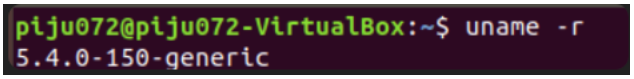






발생한 문제점과 해결방법:

운영체제가 계속해서 잘못된 OS 로 부팅되어, 과제를 수행하는 데 있어 많은 시간을 불필요한 문제를 해결하는 데 소비하게 되었습니다:



GRUB 설정 파일을 간단히 수정한 이후, 시스템이 정상적으로 원하는 OS 로 부팅되도록 설정할 수 있었습니다.

외국인이라 한국어 표현이 다소 어색하거나 틀릴 수 있는 점 양해 부탁드립니다. 그래도 최대한 성의껏 작성하려 노력했습니다.