<운영체제 1차과제 보고서>

2018320103 컴퓨터학과 김태윤

0423

목록

1. 개발환경

2. 시스템 콜 설명

3. 수정 및 작성한 부분

4. 실행 결과

5. 숙제 수행 중 발생한 문제점과 해결방법

1. 개발환경

VirtualBox를 이용한 가상머신 환경에서 구현하였다. 리눅스는 Ubuntu 18.04.2 LTS를 사용하였고, 커널은 4.20.11버전을 사용하였다.

1. 시스템 콜 설명

시스템 콜이란, kernel모드 권한이 필요한 서비스를 user모드에서 실행 중인 user application이 사용할 수 있게 하는 인터페이스다. user모드와 kernel모드 사이의 통로라고 볼 수 있다. 시스템 콜은 트랩의 일종이므로 트랩 핸들러가 처리한다.

시스템 콜의 기능은 process control, file management, device manipulation, information maintenance, communication, protection으로 나눌 수 있다. 리눅스는 300개 이상의 시스템 콜이 있다.

리눅스의 시스템 콜 호출 과정은 다음과 같다. 먼저, user application이 시스템 콜의 파라미터들을 레지스터에 저장하고, libc에서 제공하는 시스템 콜 함수를 호출한다. 그러면 트랩 핸들러가 그 시스템 콜 함수의 번호를 보고 해당하는 시스템 콜을 호출한다. 그리고 OS가 레지스터의 파라미터를 이용해서 시스템 콜에 해당하는 작업을 수행한다. 작업이 끝나면, 나온 결과를 user application에 return하고 user application 작업을 계속 수행한다. 이때는 다시 user모드로 전환된 상태이다.

1. 수정 및 작성한 부분

syscall\_64.tbl 파일에 새로 만든 시스템 콜의 고유 번호를 추가했다. ppt에 나온 대로 enqueue는 335번, dequeue는 336번으로 추가하였다. 시스템 콜이 호출되려면 이 파일 안에 해당 번호가 있어야 한다.

syscalls.h 파일에 새로 만든 시스템 콜 함수들의 prototype을 정의하였다. asmlinkage를 함수 앞에 적어서 assembly code에서도 c언어 함수 호출이 가능해지게 했다. enqueue함수는 enqueue에 필요한 int형 매개변수를 하나 갖고, int형을 return하게 했다. dequeue함수는 아무 매개변수도 없고, int형을 return하도록 정의했다.

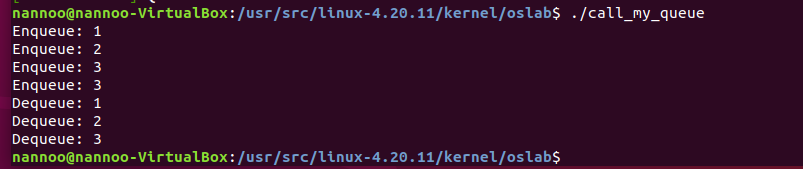
my\_queue\_syscall.c 파일을 새로 만들고, 이 안에 시스템 콜이 실제 할 일을 작성했다. 헤더는 <linux/syscalls.h>, <linux/kernel.h>, <linux/linkage.h>를 사용했다. 매크로 함수(SYSCALL\_DEFINE1, SYSCALL\_DEFINE0)를 이용해서 enqueue, dequeue함수를 구현했다. enqueue함수는 매개변수 a를 주어진 queue에 enqueue하는 역할을 한다. enqueue하기 전에 우선 queue가 full이면 error메시지를 커널 영역에 프린트하고 -2를 return한다. 그 다음에 중복인 값이 들어오면 error메시지를 프린트하고 a를 return한다. 앞의 두 가지 경우가 아니면 정상적으로 queue에 a를 enqueue하고 a를 return한다. 그리고 enqueue가 완료된 queue를 커널 영역에 프린트한다. dequeue함수는 queue의 맨 앞에 있는 값(가장 먼저 들어온 값)을 dequeue한다. 우선 queue가 비어 있는지 확인하고 비어 있으면 error메시지를 프린트하고 -2를 return한다. queue가 비어 있지 않으면, 가장 앞 요소를 dequeu하고 그 요소를 return한다. 그리고 dequeue가 완료된 queue를 커널 영역에 프린트한다.

Makefile에 kernel make할 때 포함되도록 obj-y 부분에 my\_queue\_syscall.o를 추가했다.

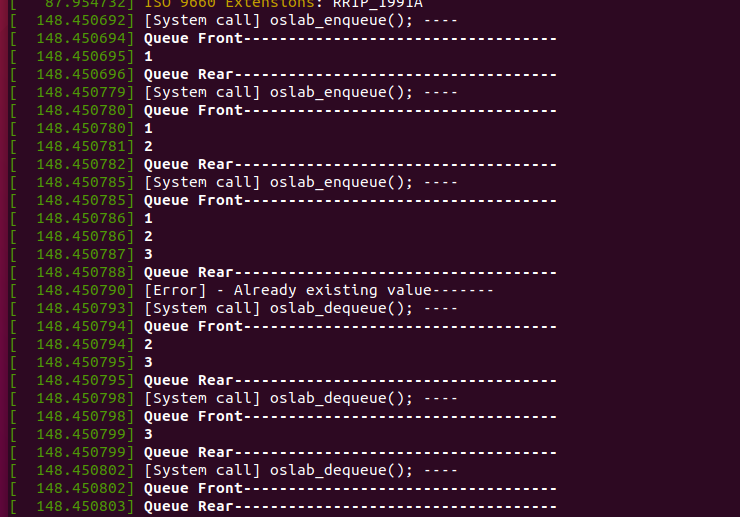
call\_my\_queue.c 파일을 생성해서 내용에 새로 만든 system call을 사용할 user application을 작성하였다. 헤더는 <stdio.h>, <unistd.h>를 사용하였다. syscall() 매크로 함수를 이용하여 시스템 콜을 호출했다. 시스템 콜 번호(335, 336)를 각각 #define으로 선언하고 사용했다. enqueue 호출은 syscall(my\_queue\_enqueue, 1) 과 같은 형식으로 호출하고 return 값은 a로 받았다. dequeue 호출은 syscall(my\_queue\_dequeue) 형식으로 호출하고 마찬가지로 return 값은 a로 받았다. 각각 1, 2, 3, 3을 enqueue하고 dequeue를 세 번 하도록 구현했다.

1. 실행 결과

call\_my\_queue 결과



kernel print 결과



1. 숙제 수행 과정 중 발생한 문제점과 해결방법

처음에 동영상과 ppt를 보면서 따라하고 call\_my\_queue.c를 실행했는데 결과값이 모두 -1이 나왔다. 검색해보니까 syscall()함수는 에러가 발생했을 때 -1을 default로 return한다고 해서 어느 곳에서 에러가 발생했는지 찾아보았다. my\_queue\_syscall.c에서 에러가 있었는데, 동영상을 그대로 따라하느라 dequeue, enqueue내용이 덜 채워져 있었다. 그래서 마저 구현하고 다시 make한 뒤 실행했는데, 또 -1이 print되었다. 이번에는 syscalls.h에 prototype을 잘못 설정했다는 것을 발견했다. ppt자료에는 enqueue가 void를 return 하도록 나와 있어서 그대로 작성했다가 에러가 난 것이었다. 그래서 int를 return하도록 수정했더니 에러 없이 잘 실행되었다.

또, my\_queue\_syscall.c의 printk부분을 수정하고 make한 후 call\_my\_queue.c를 실행했는데 수정한 내용이 반영이 안 될 때가 있었다. 이럴 때는 sudo reboot로 재시작을 했더니 적용이 되었다.