시스템 콜 추가 및 이해

2017320114 컴퓨터학과 최재원

2021. 4. 28. (Freeday 1일 사용)

I. 개발환경

1차 과제를 하기 위하여 가상머신(Virtual Machine)을 Host Operating System, 즉 나의 Mac OS 위에 설치를 했다. Virtual Machine은 Oracle에서 제공해주는 VirtualBox의 19년 02월 기준의 6.0.4 버전을 설치했다. Virtual Machine의 기본 스펙은 메모리(RAM) 크기는 4096MB(4GB)로 할당해 주었고, 하드 디스크는 60GB로 설정해 주었다. 만들어진 Virtual Machine 위에 설치되는 Guest OS으로는 리눅스를 사용하였다. 여러 가지 리눅스 버전 중에서 1차 과제를 위한 리눅스는 Ubuntu 18.04.2를 사용하였다.

II. 리눅스의 시스템 콜

리눅스의 시스템 콜은 사용자 응용 프로그램에서 운영체제의 기능을 수행해달라고 부탁하는 커널의 서비스이다. 만약 사용자 레벨에서 fork() 시스템 콜을 호출한다고 하면, 미리 제시되었던 파일을 읽는 read 함수와 비슷하게, C 라이브러리에서 fork() 시스템 콜의 고유번호를 찾고 그 번호를 레지스터에 저장한 뒤 0x80 트랩을 발생시킨다. 보통 0x80은 커널 공간 안에 있는 Descriptor table에 System_call로 되어있고 이로 인해 시스템 콜이 실행되게 되는 것이다. System_call 함수에서는 호출된 system call 번호와 즉 여기서는 fork()의 번호와, 모든 레지스터를 스택에 저장하고 올바른 시스템 콜 번호인지 검사 후, 트랩 핸들러가 Sys_call_table의 번호를 호출해서 fork()의 고유번호의 table에 저장되어 있는 것을 호출을 하게 된다. 이러한 커널 서비스가 마치게 되면 커널 내의 return 함수에 의해 사용자 프로세서로 돌아가게 된다.

III. 수정 및 작성한 부분과 이유

커널 소스코드의 수정을 위해서는 5개의 파일 수정 및 작성을 필요로 했다. 각각의 파일들을 설명을 하자면, 새로 추가할 시스템 콜 번호 할당을 위한 syscall_64.tbl 파일 수정(ex. 335번, 336번에 enqueue dequeue 추가), 추가한 시스템 콜 함수들의 정의 및 테이블 등록을 위한 syscalls.h 파일 수정(ex. Int void 선언, c 함수 호출을 가능케 하는 asmlinkage 선언), 새로 추가할 시스템 콜 작성을 위한 my_queue_syscall.c 파일 작성, 소스파일(.c 파일)을 오브젝트 파일 즉 실행파일로 변환시킬 수 있게 하는 my_queue_syscall.o 오브젝트를 makefile에 추가해 주고, 마지막으로 추가된 시스템 콜을 사용하는 application 소스 파일 call_my_queue.c 작성 및 실행 파일로 변환해서 추가된 시스템 콜들이 잘 출력이 되는지 확인을 했다.

my_queue_syscall.c 코드 작성에 대해서 조금 더 구체적으로 설명을 하자면, enqueue,

dequeue 구현에 있어서, 선형 큐 방식으로 구현을 했다. 즉, 가장 먼저 들어온 데이터를 가장 먼저 내보내는 FIFO 형태이다. Enqueue는 front는 고정을 한 상태에서 만약 인자로 들어온 데이터 a가 rear 전까지 같은 데이터를 가지는 값이 없고, input data로 0이 들어오고 그전에 0이 없으면 초기화 값이랑 겹치지 않게 하고, 마지막으로 rear가 MAXSIZE, 즉 큐가 차 있지 않으면 현재 queue[rear]의 자리에 데이터 a를 넣어주고 rear의 값을 증가 시켜 다음 자리로 옮기게 해주는 방식으로 구현을 했다. 반면에 dequeue의 경우에는 front의 인덱스를 증가시켜서 데이터를 내보내는 기능을 하게 구현을 했다. 즉 res = queue[front] 그리고 front++의 형식으로 했다. Enqueue, Dequeue 모두 기본적으로 결과를 프린트 할 때 기본 배열의 for문을 이용해서 현재 front에서 rear까지의 값이 출력되도록 했다.

IV. 실행 결과 스냅샷

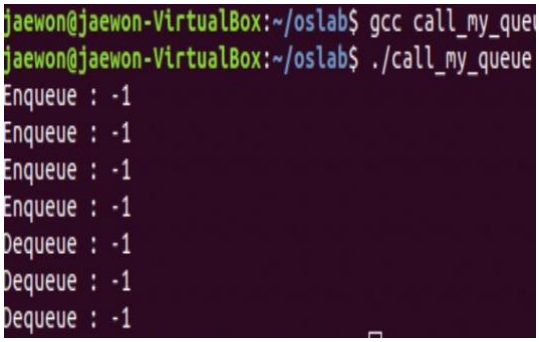
```
jaewon@jaewon-VirtualBox:~/oslab$ gcc call_my_queue.c -o call_my_queue
jaewon@jaewon-VirtualBox:~/oslab$ ./call_my_queue
Enqueue : 1
Enqueue : 2
Enqueue : 3
Enqueue : -1
Dequeue : 1
Dequeue : 2
Dequeue : 3
```

<./call_my_queue 실행 결과>

```
[ 1670.196396] [System call] oslab_enqueue(); -----
[ 1670.196397] Queue Front-----------------------
[ 1670.196398] 1
[ 1670.196398] Queue Rear------------------------
[ 1670.196465] [System call] oslab_enqueue(); -----
[ 1670.196466] Queue Front-----------------------
[ 1670.196466] 1
[ 1670.196467] 2
[ 1670.196467] Queue Rear------------------------
[ 1670.196469] [System call] oslab_enqueue(); -----
[ 1670.196470] Queue Front-----------------------
[ 1670.196470] 1
[ 1670.196471] 2
[ 1670.196471] 3
[ 1670.196472] Queue Rear------------------------
[ 1670.196474] [Error] - Already existing value
[ 1670.196476] [System call] oslab_dequeue(); -------
[ 1670.196477] Queue Front--------------------
[ 1670.196477] 2
[ 1670.196478] 3
[ 1670.196478] Queue Rear--------------------
[ 1670.196480] [System call] oslab_dequeue(); -------
[ 1670.196480] Queue Front--------------------
[ 1670.196481] 3
[ 1670.196481] Queue Rear--------------------
[ 1670.196483] [System call] oslab_dequeue(); -------
[ 1670.196484] Queue Front--------------------
[ 1670.196484] Queue Rear--------------------
```

<dmesg 실행 결과>

V. 과제 수행 과정 중 발생한 문제점과 해결방법



1차 과제를 수행하던 중 처음에 조금 헤매고 시간을 많이 소모했던 문제점은 크게 두 가지였다. 처음에는 직면한 문제점은 바로 my_queue_syscall.c를 작성하고 나서 커널 컴파일에 들어가는 순간 커널이 오랜 시간 동안 새로운 드라이버 설치 및 컴파일을 마치고 나서, 사용자 응용 프로그램을 통해서 enqueue dequeue를 실행하는 순간 왼쪽의 사진과 같이 전부 -1이 출력되었다. 당황스러워서 구글을 통해 여러 자료들을 찾아보는 와중에, 가장 기본적인 실수를 했다는 것을 깨달았다. 새로운 커널 시스템콜을 수정하고 설치를 하였는데 재부팅을 안하고 그대로 수행을 한 것이었다. 따라서 VM을 재부팅 하니까 결과적으로 잘 실행되었다.

또 다른 문제점은 위에 정상적으로 실행된 결과에서 1 -> 2-> 3-> 그리고 다음 3에서 출력 예시처럼 3이 출력이 안되고 -1이 출력이 되는걸 확인할 수 있었다. 처음에는 이것이 출력 예시랑 달라서 문제가 있다고 생각을 하였고, 검색을 하던 와중에 페이스북 운영체제 FAQ란에서 이와 비슷한 질문의 조교의 답변을 보게 되었다. 결과적으로 겹치는 input data가 들어왔을 때 겹치는 값이 있으면 내 코드에서는 -2 값을 return 하게 해 주었고, syscall 함수는 음수 반환 값을 에러로 간주한다. Errno은 global variable에 할당되고 결과적으로 syscall은 -1을 반환하게 되는 것이다. 결과적으로 -1이 나오는 것이 옳은 결과라는 것을 알게 되었다.