[COSE341] 운영체제

Assignment #1

정보대학 데이터과학과 2021320307 김은진

2023-10-13 (freeday 1일 사용)

1. **Development environment**

Windows 11 intel core i5 노트북, Virtual Box, Ubuntu 18.04.2 LTS, Linux 4.20.11 Kernel

2. **Explanation of system calls on Linux (including call routines)**

시스템 콜은 유저 모드에서 커널 모드로 전환하기 위한 인터페이스로, 커널에 의해 보호되고 있는 서비스를 사용하기 위해서 필요하다. 프로그램이 시스템 콜을 바로 호출하지 않고, high level APIs를 사용한다.

리눅스의 시스템콜은 C 라이브러리 함수인 glibc를 통해 제공된다. 유저 모드에서 동작하고 있는 프로세스에서 시스템 콜을 호출하면 해당 시스템콜이 저장된 메모리 주소에서 trap instruction이 발생한다. 그러면 발생한 trap에 대응되는 trap handler가 동작하게 되고 이 handler가 커널 모드의 시스템 콜 테이블에서 호출된 시스템 콜의 고유번호를 확인하고 실행한다. Trap handler가 완료된 후에는 원래 실행하고 있던 task를 수행하기 위해 유저 모드의 프로세스로 돌아온다.

3. **Implementation: modified and written codes with descriptions**

1. /arch/x86/entry/syscalls/syscall_64.tbl에 시스템 콜 추가

   Syscall_64.tbl은 System call handler의 이름 정보와 번호를 가지고 있는 파일이다. Linux kernel의 코드 디렉토리에 있는 system call 함수의 주소가 저장된 테이블을 포함하고 있는데, 이 테이블에 과제를 위한 os2023_push와 os2023_pop 함수와 각각의 번호를 추가해 주었다.

2. /include/linux/syscalls.h에 시스템 콜 함수 정의

   Syscalls.h는 시스템 콜 함수의 프로토타입을 정의하고 필요한 테이블을 등록하는 파일이다. asmlinkage int sys_os2023_push(int); asmlinkage int sys_os2023_pop(void); 를 추가해주었다. 추가한 system call이 trap instruction에서 호출되는데 trap을 처리하는 루틴은 주로 어셈블리 코드로 작성되기 때문에 C 함수가 trap루틴에서 안전하게 호출되도록 asmlinkage 매크로를 사용하였다.

3. /usr/src/linux-4.20.11/kernel/oslab_my_stack.c에 시스템 콜 함수 구현

추가하려는 system call handler가 동작하기 위한 실제 소스 코드가 담겨 있는 파일이다. 정수가 저장될 stack 변수와 index를 저장하기 위한 top 변수를 전역 변수로 선언하였다. Push함수는 스택에 넣을 정수를 인자로 받고, pop함수는 인자를 받지 않기 때문에 각 system call을 구현할 때 SYSCALL_DEFINE1, SYSCALL_DEFINE0 매크로를 활용하였다.

1) os2023_push

for loop에서 활용할 i 변수와 이미 스택에 들어있는 숫자인지 판단하기 위한 already 변수를 선언하였다. Top변수 값이 스택의 최대 용량인 STACK_SIZE(100으로 설정) 이상이라면 스택이 꽉 찼음을 커널에 프린트하고 종료된다. 비어 있다면 already를 0으로 초기화 후 for loop를 통해 이미 있는 정수인지 판단한다. 있었던 수라면 already를 1로 바꿔준다. Already가 0인 경우에만 top에 1을 더한 후 stack의 top번째 index에 정수를 추가한다. 이후에는 커널에 스택을 출력한 후 처음 받았던 정수를 반환하며 종료된다.

2) os2023_pop

for loop에서 활용할 i, j 변수와 버블 정렬에서 활용할 temp 변수, 최댓값을 반환하기 위한 max변수를 선언하였다. Top이 -1 이하라면 스택이 비어있음을 커널에 출력하고 종료된다. 아니라면 버블정렬을 통해 스택을 오름차순으로 정렬하고 stack의 top번째 원소를 max변수에 저장한다. 이후 정렬되고 pop된 스택을 커널에 출력하고 최댓값을 반환하며 종료된다.

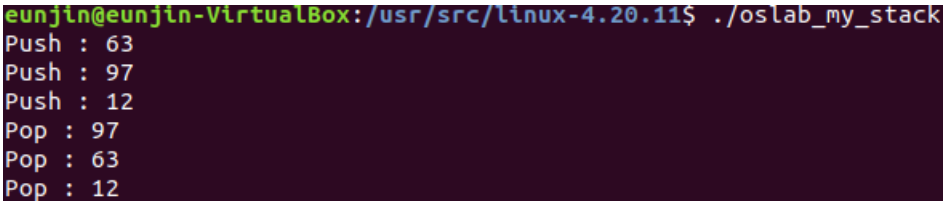
4. /usr/src/linux-4.20.11/kernel/Makefile에 object file 추가

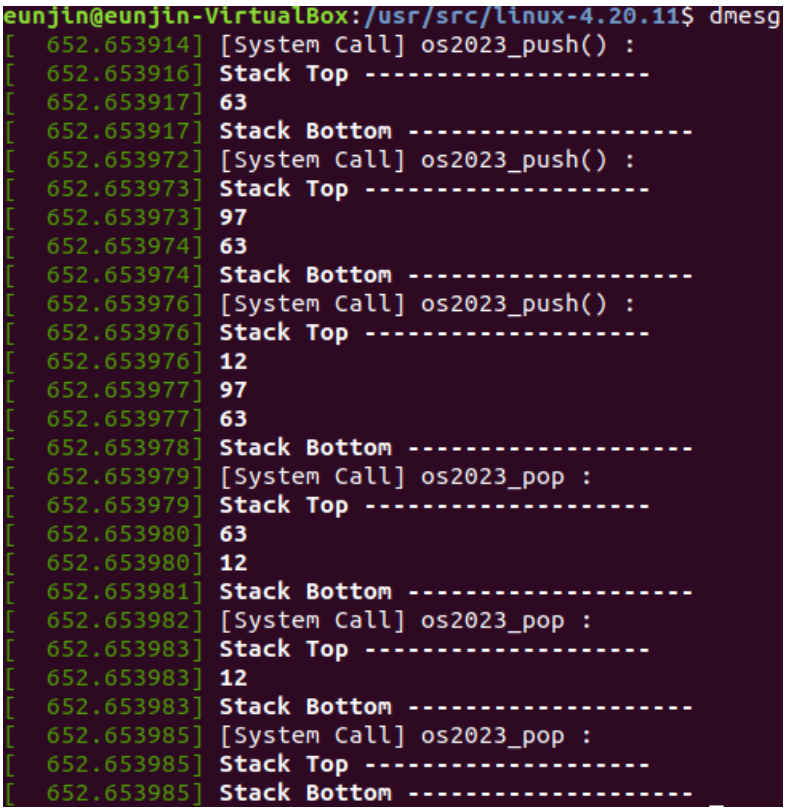
소스코드를 컴파일 할 때 추가한 system call handler도 컴파일 되도록 oby-j 부분에 oslab_my_stack.o를 추가하였다.

5. /usr/src/linux-4.20.11/oslab_my_stack.c (제출파일에서는 call_my_stack.c)

   추가한 system call을 호출하는 user application 파일이다. 랜덤으로 정수를 생성하기 위해 time.h와 stdlib.h 라이브러리를 include하였고 syscall() 매크로를 쓰기 위해 unistd.h 라이브러리를 include하였다. 정수를 랜덤으로 생성하여 push로 스택에 세 개의 정수를 추가하는데, 이전 정수와 똑같은 정수가 생성되었다면 한번 더 정수를 생성해주는 방식으로 개수를 보장하였다. 이후에는 pop을 통해 스택에 넣었던 세 개의 정수를 세 번에 걸쳐 내림차순으로 스택에서 꺼낸다. Push나 pop이 실행될 때마다 'Push : [정수]', 'Pop : [정수]' 형태로 출력하도록 하였다.

4. Snapshot of project execution (both from the user application and the kern el)





5. Difficulties and your efforts (maybe solutions) encountered during this proje ct

1. Kernel 디렉토리의 파일을 편집할 때 permission denied

   파일이 readonly로 열려 편집하는 데 권한 오류가 생겼다. 'sudo chmod -R 777 [파일명]' 명령어를 이용하여 권한을 부여하여서 해결하였다.

2. 초기 /usr/src/linux-4.20.11/kernel/oslab_my_stack.c 컴파일 오류

   C90에서는 for loop 안에서 변수를 초기 선언할 수 없기 때문에 for loop 밖에서 미리 선언하도록 수정하였다. Already변수를 선언하지 않았거나 STACK_SIZE를 STACKSIZE로 작성한 실수도 수정하였다.

3. ./oslab_my_stack 실행했을 때 시스템 콜 미작동

   오류가 없는데도 시스템 콜이 작동하지 않아 'uname -r'로 kernel의 버전을 확

인했더니 4.20.11.oslab이 아닌 다른 버전으로 되어있어서 reboot후 왼쪽 shift 버튼을 누르고 4.20.11.oslab 버전을 실행하여 해결하였다.