# 2023년 1학기 운영체제 1차 과제

**지도 교수님** 양경식 (ksyang@os.korea.ac.kr)

**조교** 고의진, 정승우 (osta@os.korea.ac.kr)

**주제** 시스템 콜 추가 및 이해

**출제일** **2023. 3. 21. (화)**
**제출일** **2023. 4. 11. (화) PM 11:59**

**환 경** Ubuntu 18.04.02 (64bit), Linux kernel 4.20.11 (가상머신 환경)

**목 차**

## 1. 목적

이번 과제의 목적은 일반 배포판 리눅스에 **새로운 Kernel을 컴파일하고 수정해서 새로운 System Call을 추가하는 것**이다. System Call은 Kernel이 시스템이나 하드웨어에 특권이 있어야만 하는 동작들을 지정해둔 함수로써 User-Space에서 Kernel-Space로 전환하기 위해서 특별히 지정된 **Trap Instruction**을 거쳐야 한다. 본 과제에서는 Kernel을 직접 컴파일하고, 이후 **기존의 Kernel에 새로운 System Call을 추가한 후에 이를 호출할 수 있는 User-Application까지를 구현**하도록 한다.

새로 추가한 system call은 내부적으로 Stack 자료구조를 갖도록 한다. 그리고 이 Stack에 integer값을 Push하거나 Pop하는 역할을 수행하도록 작성한다. 단, 중복된 integer값을 저장할 수 없다. 그 밖에 정해진 Stack의 구현 방법은 없으며 자유롭게 구현할 수 있다. 기본적인 Push와 Pop가 가능하면 된다. Stack에 저장할 수 있는 entry의 개수 또한 고정적으로 하여도 무관하다.

## 2. 과제 목표

**A.** Kernel

- VirtualBox를 이용한 가상 머신 환경에 구현하도록 한다.
- 리눅스는 Ubuntu 18.04.2 LTS 배포판으로 한다.
  i. MacOS의 M1, M2 CPU의 경우 해당 버전의 Ubuntu 배포판으로 실행이 불가 할 수 있음. 다른 버전의 Ubuntu Desktop 사용시 보고서에 명시한다
- 새로 컴파일하는 Kernel Version은 4.20.11 로 한다.
- 커널 상에 Stack를 구현한다.
- System Call시 전달받은 정수형 인자를 printk로 출력하도록 한다.

**B.** User Process

- syscall를 이용해서 새로 작성한 System Call을 호출하도록 한다.
- Push와 Pop가 제대로 동작하는지 확인한다.
- System Call의 반환 값을 출력하도록 한다.

## 3. 세부사항

### A. System call

앞에서도 간략하게 설명했지만 **System Call이란 리눅스 커널에 의해 제공되는 Kernel-level 서비스**이다. 예를 들어 프로그래머가 파일을 읽는 서비스를 이용하고 싶다면 그에 해당하는 System Call을 이용해야 한다. C를 이용해서 프로그래밍할 경우 대부분의 System Call은 libc를 통한 Wrapper Function 형태로 제공받을 수 있다. 예를 들어 User Application에서 파일을 읽는 read 함수가 호출되었다고 하면 Figure 1과 같은 호출이 일어나게 된다.

Figure 1과 같이 User Process에서 read를 호출하면 Wrapper Function을 통해서 0x80의 Trap Instruction이 발생한다. 0x80번째 Trap은 System Call로 예약이 되어 있고 이 Trap Handler는 Sys_call_table의 번호를 확인해서 3번째 Table에 저장되어 있는 sys_read를 호출하게 된다. 이렇게 커널에서의 서비스가 끝나면 다시 User Process로 결과값과 함께 return하게 된다. 이번 과제에서는 sys_call_table에 2개의 Entry를 더 추가해서 sys_os2023_push, sys_os2023_pop을 User Process에서 호출하고 dmesg를 통해서 printk로 출력한 메시지를 확인하도록 한다.

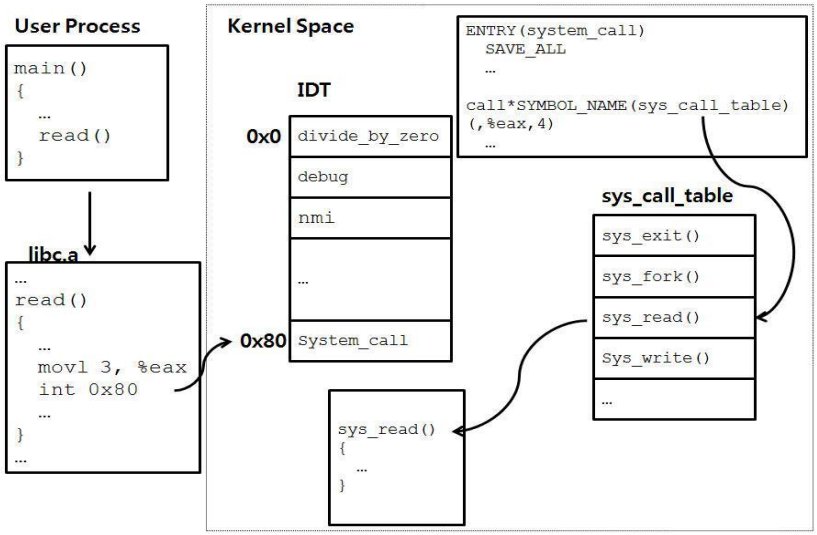



**Figure 1. syscall example**

### B. Kernel에서 문자열 출력 (printk 함수)

Kernel 영역에서는 기존에 사용하던 printf를 사용하지 못한다. 커널 영역에서 메시지를 출력하려면 printk를 사용해야 하며, 함수의 사용법은 기본적으로 printf와 같다. printk는 특별히 중요도를 지정해서 출력할 수도 있지만 이번 과제에서는 사용하지 않도록 한다. System Call에서 printk를 통해 메시지를 출력했다면 일반 Terminal 화면으로는 볼 수 없고, 최근에 발생한 커널 메시지를 확인하는 dmesg 명령어를 통해서 확인할 수 있다.

## 4. 커널에서 수정해야 하는 주요 파일 목록

A. (linux)/arch/x86/entry/syscalls/syscall_64.tbl (64비트 머신)
→ 시스템 콜 함수들의 이름에 대한 심볼 정보를 모아 놓은 파일

B. (kernel)/include/linux/syscalls.h
→ 추가한 시스템 콜 함수들의 prototype 정의

C. (kernel)/oslab_my_stack.c
→ 새로 추가할 시스템 콜의 소스

D. (kernel)/Makefile
→ oslab_my_stack.o 오브젝트 추가

### 5. 추가한 System Call을 호출하는 User Application

System Call도 함수로 되어 있고 User Application에서 호출해야 실행된다. System Call을 호출하는 방법에도 여러 가지가 있는데 그 중 하나가 syscall 매크로를 사용하는 것이다. syscall을 사용하는 방법은 syscall(시스템콜 번호, args...) 이다. 예를 들어 새로 추가한 시스템 콜이 335번째 시스템 콜로 define이 되어 있다고 가정하면 User Application에서의 구현은 다음과 같다.

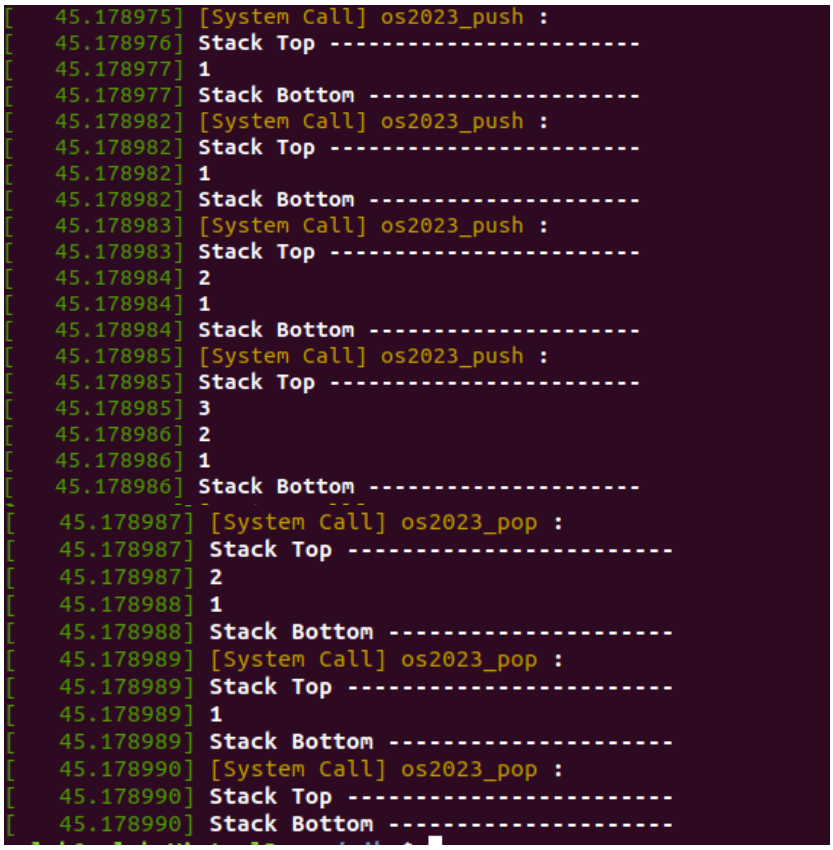
```
#include<unistd.h>
...
        int r;
        r = syscall( 335, 123 );            // System Call 호출
```

**또는**

```
#include<unistd.h>
#define my_stack_syscall 335 // 커널의 헤더 파일과 동일한 번호로 정의된
                                        System Call을 User에서도 지정
...
        int r;
        r = syscall(my_stack_syscall, 123);      // System Call 호출
```

### 6. 과제 결과 출력 형식

```
oslab@oslab-VirtualBox:~/ejko$ ./oslab_call_stack
Push 1
Push 1
Push 2
Push 3
Pop 3
Pop 2
Pop 1
```




**A. 제한사항**

- 리눅스 배포판 및 커널 버전은 과제에서 명시된 버전을 사용한다.
  i. MacOS의 M1, M2 칩을 사용하는 운영체제의 경우 등 다른 버전을 사용할 경우 보고서에 이를 명시한다.

## 7. 제출 방법

**A. Blackboard를 통하여 파일을 제출한다.**

**B.** 보고서에 아래 내용을 반드시 **포함**하도록 한다. (보고서는 표지 제외 5페이지를 넘지 않도록 한다)

- 학과, 학번, 이름, 제출 날짜, Freeday 사용 일수 (표지에 기재)
- 개발환경
- 리눅스의 시스템 콜 (호출 루틴 포함)에 대한 설명
  i. **주의:** 각 자료를 그대로 복사, 붙여넣기 금지
  시스템 콜과 관련된 자료는 매우 많으나, 반드시 자료를 이해 후 자신의 글로 작성할 것
- 수정 및 작성한 부분과 설명 (이유)
  i. 소스를 그대로 첨부하지 말고, 전체적인 흐름과 수행한 작업을 설명한다.
- 실행 결과 스냅샷
- 숙제 수행 과정 중 발생한 문제점과 해결방법

**C. 제출할 파일 목록**

- 보고서
- 직접 작성한 소스 파일 전체
  i. 소스 코드에는 중요 변수와 함수의 역할에 대한 주석을 작성해야 한다.
- 실행 결과 파일 (result.txt)
  i. 실행 결과(6. 과제 결과 출력 형식)를 text 파일로 제출

**D. 파일 제출 방법**

1) **새로 작성하거나, 수정한 모든 소스 파일, Makefile**을 Sources라는 폴더에 모으고, **보고서, 결과 파일을 더하여 "os1_학번_이름.zip"**으로 제출한다.
2) **소스파일은 절대로 리눅스 커널 소스 전체를 제출하지 않도록 한다.**
3) 동일한 환경에서 컴파일과 실행이 가능해야 한다.

**8. 비고**

**A.** 프리데이를 초과한 지각제출의 경우, 지각제출 1일당 **과목 전체 점수에서 1점씩 감점**됨

**B.** 2차과제 마감일 1주일 이후부터는 모든 과제제출이 불가함