**2020년 1학기 운영체제 1차 과제**

**학과** 컴퓨터학과

**학번** 2018320112

**이름** 김현아

**제출일** 2020.5.1

**Freeday 사용 일수** 1일

**목차**  개발환경

리눅스의 시스템 콜(호출 루틴 포함)

수정 및 작성한 부분과 설명 (이유)

실행 결과 스냅샷

숙제 수행 과정 중 발생한 문제점과 해결 방법

1. **개발환경**

Ubuntu 18.04.02 (64bit), Linux kernel 4.20.11 (가상환경 oracle VM virtualbox)

1. **리눅스의 시스템 콜(호출 루틴 포함)**

시스템 콜(system call)이란 user mode에서 실행중인 어플리케이션이 kernel mode의 권한이 필요한 서비스를 받기 위한 방법을 말한다. User application이 함수를 통해 system call interface를 실행시키면 system call table에서 해당 번호에 해당하는 함수를 찾아서 kernel이 system routine을 호출하게 되고, kernel mode에 접근 가능하게 해준다. 보다 자세히 설명하자면, user mode의 사용자가 시스템 콜을 호출한다고 가정하였을 때, C library에서 해당 시스템 콜의 고유번호를 레지스터에 저장하고, interrupt를 발생시킨다. 커널은 IDT에서 해당 주소에 있는 시스템 콜을 찾는다. 이 함수에서 호출된 시스템 콜 번호와 모든 register를 stack에 저장, 확인 후 system call table에서 시스템 콜 번호에 해당하는 함수를 호출한다. 서비스를 모두 처리한 후에 다시 user mode로 돌아오게 되며, kernel이 동작을 return 할 때까지 user application은 멈춰있는다.

1. **수정 및 작성한 부분과 설명 (이유)**
2. Syscall\_64.tbl



syscall\_64.tbl은 함수들의 이름에 대한 심볼 정보를 모아둔 파일이다. System call 335번에 push를, 336번에 pop을 지정해주었다.

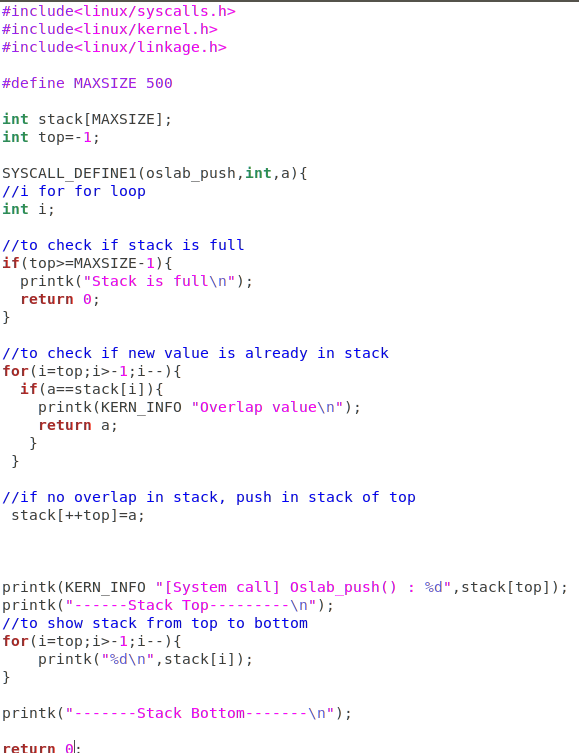
1. Syscalls.h

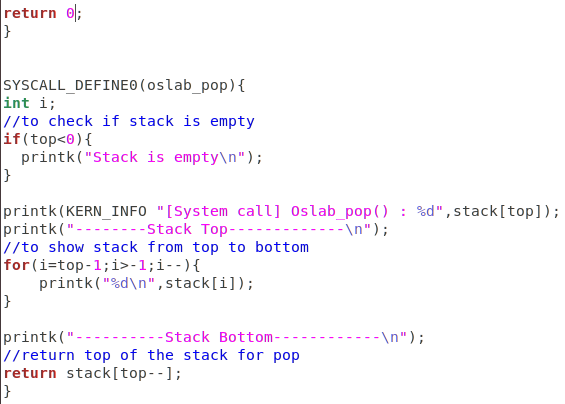


Syscalls.h에 추가한 함수의 프로토타입을 정의하였다. Asmlink를 사용한 이유는 시스템 콜 호출이 int 80 인터럽트 핸들러에서 호출되는데 이것은 어셈블리 코드로 작성된다. 하지만 oslab\_push와 oslab\_pop은 C언어로 작성되기 때문에 asmlinkage를 사용하여 어셈블리 코드(핸들러)에서도 C함수를 호출할 수 있게 한다.

또한 push함수는 stack에 int를 받아 추가하고, pop은 stack에서 위에 있는 순서대로 int를 빼기 때문에 각각 void sys\_oslab\_push(int), int sys\_oslab\_pop(void)로 작성하였다.

1. My\_stack\_syscall.c

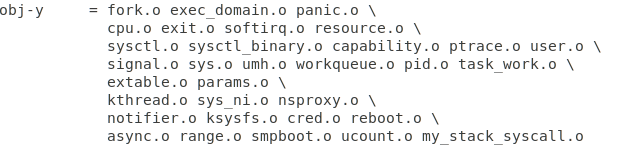




시스템 콜의 구현을 my\_stack\_syscall.c에 하였다. 자세한 내용은 주석을 통해 설명해 두었으며, SYSCALL\_DEFINE1은 파라미터의 개수가 1개인 시스템콜 구현을 위한 매크로이다. 마찬가지로 SYSCALL\_DEFINE0은 파라미터의 개수가 0개인 시스템콜 구현을 위한 매크로이다. Push, pop 함수 모두 맨 처음 stack이 full/empty인지 확인하였고, push함수의 경우 push함수를 통해 추가하려는 값이 이미 stack에 있는지 확인하고 그 값을 리턴하도록 for~if부를 구성하였다. 그 후 push, pop함수 모두 어떤 integer가 push/pop되는지 그리고 실행 이후의 stack상태를 출력하도록 하였다.

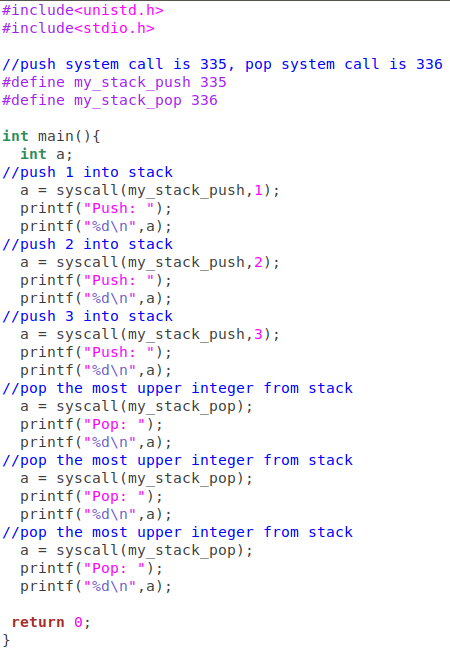
push함수의 경우 void push (int)이기에 return값을 주지 않았고, pop함수의 경우 int pop (void)이기 때문에 pop하는 수를 return하도록 하였다.

1. Makefile

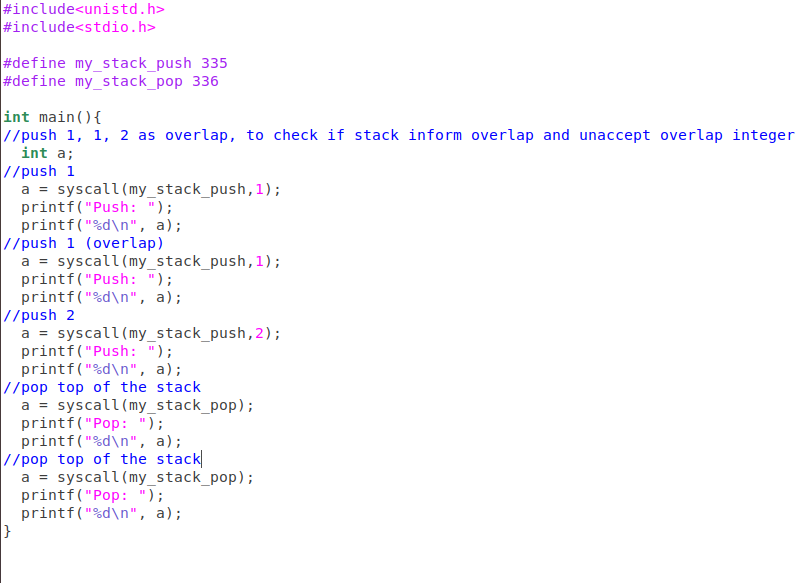


Kernel make시에 포함되도록 obj-y부분의 맨 뒤 부분에 my\_stack\_syscall.o를 작성하였다.

1. Call\_my\_stack/ex1



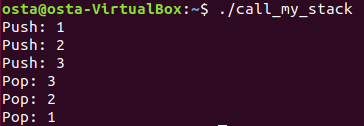
Call\_my\_stack.c는 stack의 기본 기능을 확인하기 위해 작성하였으며 먼저 1, 2, 3,을 순서로 push하고 그 후 세 번 pop을 진행하여 순서대로 3, 2, 1이 pop하는 것을 염두에 두고 만든 함수이다. 이것을 gcc로 compile하여 call\_my\_stack.o 파일이 생성되었다.



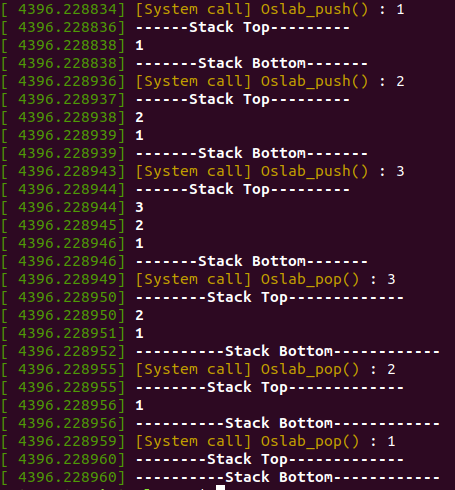
Ex1.c는 call\_my\_stack과 비슷하게 작성되었고, 프로그램의 조건인 push 함수를 통해 추가하려는 값이 이미 Stack에 있는 값과 같으면 Stack에 추가하지 않는다를 만족하는지 확인하기 위해 작성되었다. 1을 두 번 push하여 중복된 값이 push되는지 되지 않는지 확인한다. 마찬가지로 gcc로 compile하여 ex1.o 파일이 생성되었다.

1. **실행 결과 스냅샷**
2. 기본 : Call\_my\_stack 실행

<응용 프로그램 출력>

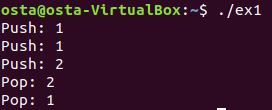


<커널 로그 출력 (dmesg 명령어)>

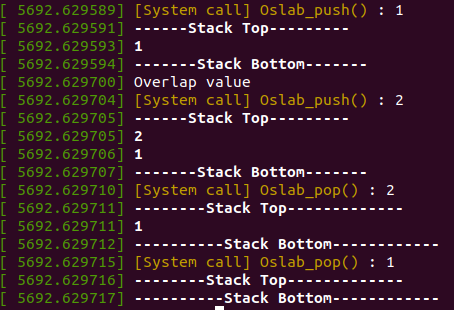


1. 조건 : ex1 실행[중복된 값 push했을 때 받아들이지 않도록]

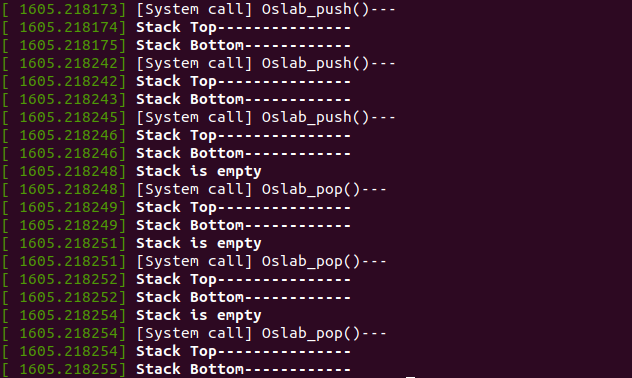
<응용 프로그램 출력>



<커널 로그 출력 (dmesg 명령어)> 두번째 1이 push될 때 overlap value라고 뜸



1. **숙제 수행 과정 중 발생한 문제점과 해결방법**
2. My\_stack\_system.c가 제대로 컴파일되지 않은 점



이 상태는 처음 tutorial을 보고 뼈대만 만든 것인데 이후 코드를 수정하고 실행할 때마다 아무리 코드를 수정해도 sudo dmesg 상태에서 실행 결과가 바뀌지 않았다. Reboot를 하고 새로 sudo make를 하니 해결되었다.

1. Error : drm:vmw\_host\_log [vmwgfx] Failed to send host log message

Application을 작성하고 sudo dmesg command를 입력한 후 이런 에러가 발생했다. 구글링을 통해 환경설정에서 display의 graphics controller를 VBOXxBGA로 바꾸니 실행되었다. 이 과정에서 환경설정을 바꾸기 위해서는 우선 ubuntu를 종료하여야 한다는 점을 배웠다.