운영체제 1차과제 보고서

학 과: 컴퓨터학과

학 번: 2017320215

이 름: 임준상

제출일: 2019.04.19

Free Day 사용 일수: 0

**<목 차>**

I**.** 개발 환경과 시스템 콜에 대한 설명

II.수정 및 작성한 부분과 설명

III.실행 결과 스냅샷

IV.문제점과 해결방법

I**.** 개발 환경과 시스템 콜에 대한 설명

**개발 환경:**

System: Ubuntu 18.04.02 (64bit)

Kernel Version: Linux kernel 4.20.11 (**가상머신 환경**)

Compiler: GNU C Compiler

Text Editor: Text Editor, Notepad++

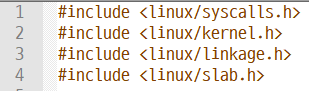
**시스템 콜에 대한 설명:**

현재 쓰고 있는 시스템은 보통 커널에 있는 데이터와 기능을 사용할 수 없다. 왜냐하면 컴퓨터에서 중요한 데이터와 자원이 매우 많다. 이런 데이터와 자원들이 마음대로 사용되면 오류를 발생해서 system crash가 될 것이다. 시스템이 application가 실행할 때 system crash가 발생하지 않기 위해 모두 권한을 application에게 주지 않고 마음대로 데이터와 자원들을 사용할 수 없다. 시스템은 application과 커널을 분리해서 다른 모드로 실행한다. Application의 모드는 유저 모드이고 커널의 모드는 커널 모드이다. 리눅스 시스템 속에서 application은 시스템 콜을 호출해서 커널의 자원과 기능을 쓸 수 있다. 예를 들면 프로세스에서 다른 프로세스를 생성, 메모리 공간을 요청 등이 있다. 그러므로 system call는 유저 모드와 커널 모드 사이의 소통 방법이다. 시스템 콜은 interrupt로 실행하며 유저 모드에서 커널 모드로 바꾼다. 커널 모드에서 system call number로 시스템 콜 루틴을 찾아서 실행한다. 실행되어 나오는 값은 return해서 유저 모드의 코드를 얻는다.

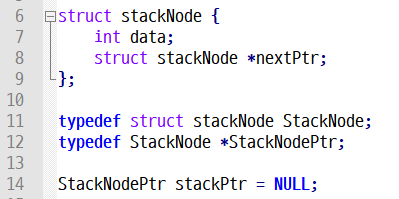
II.수정 및 작성한 부분과 설명

my\_stack\_syscall.c는 추가한 시스템 콜 소스이고 call\_my\_stack.c는 외부의 시스템 콜을 사용하는 application이다.

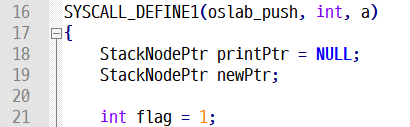
먼저 my\_stack\_syscall.c에 대해 설명한다. 처음에은 헤더 파일을 선언한다. kmalloc과 kfree를 사용하기 위해 #include <linux/slab.h>를 추가하였다.



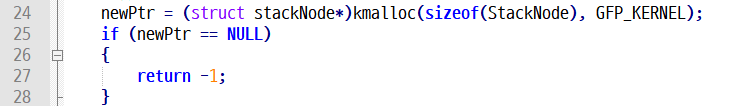
다음은 stack의 구조체를 만들었으며 typedef 함수로 stack의 노드와 노드의 포인트를 정의하였다. 그리고 뒤에 pop 함수가 메모리를 풀 수 있기 위해 전역 stack 포인트를 정의하였다.



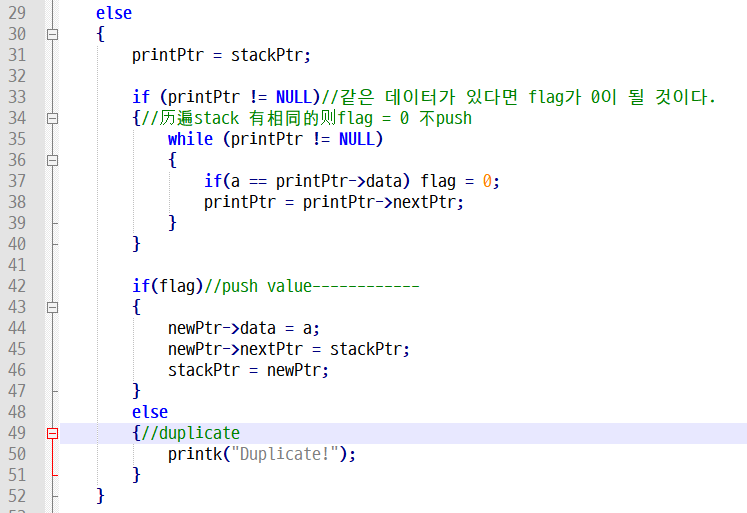
다음은 SYSCALL\_DEFINE1(oslab\_push, int, a) 함수 안에의 코드이다. 여기는 printPtr, newPtr와 flag을 선언하였다. printPtr는 전체 stack를 살펴봐서 같은 숫자가 있는지 없는지를 확인하고 stack를 print하는 두 가지 작용이 있다. newPtr는 메모리 공간을 분배하고 노드를 저장하는 작용이다. int 형 숫자 flag는 input 데이터와 stack 안에 있는 데어터가 비교하는 결과를 기록하는 작용이다.



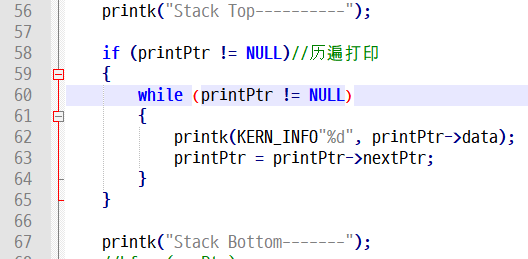
다음에는 메모리를 분배하는 코드이다. 실패되면 -1를 return한다.



메모리 공간 분배가 된다면 printPtr=stackPtr, 전체 stack를 살펴봐서 같은 value가 있다면 flag가 0이 될 것이다(line 37). 다음에는 push value이다(line 42). flag가 0이면 if 함수 안에 있는 push 코드가 실행되지 않고 “Duplicate!”를 print한다.



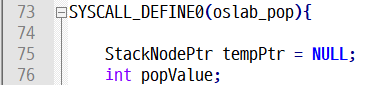
다시 printPtr=stackPtr, 전체 stack를 살펴봐서 print한다.



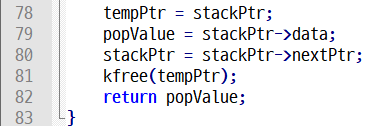
마지막으로 flag의 값은 1이다면 숫자 1를 return하고 종복없이 push가 된 것을 의미한다. flag의 값은 0이다면 숫자 0를 return하고 종복이 있다는 뜻을 의미한다. SYSCALL\_DEFINE1(oslab\_push, int, a) 함수의 설명은 여기까지 마쳤다.



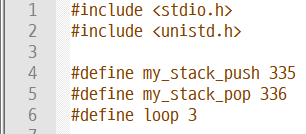
다음에는 SYSCALL\_DEFINE0(oslab\_pop) 에 대해 설명이다. 먼저 tempPtr을 만들고 int 형 숫자 popValue를 만들었다.



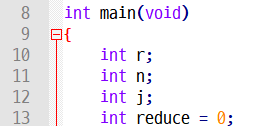
일단 tempPtr=stackPtr, 그리고 popValue 값은 현재 node의 숫자가 된다. 또한, stackPtr가 다음의 노드에 가며 tempPtr(현재 노드)을 free한다. 마지막으로 popValue(free된 데이터)를 return한다.



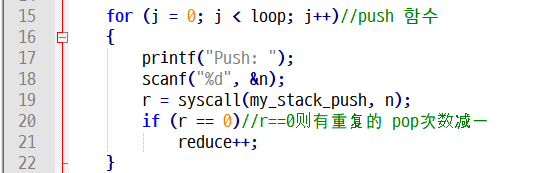
이상으로 추가한 시스템 콜 소스에 대한 설명이다. 다음에는 시스템 콜을 사용하는 application이다. 일단 헤더 파일을 선언하고 my\_stack\_push, my\_stack\_pop 그리고 loop의 값을 정의하였다. loop는 push 사이클의 횟수이다.



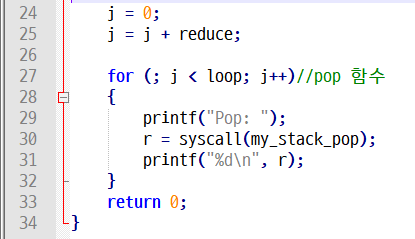
다음은 int 형 숫자 r, n, reduce를 만들었다. r의 작용은 syscall return하는 값을 받는 것이다. n는 입력하는 데이터이다. reduce는 뺄 pop 횟수이다. 처음에 reduce의 값은 0이고 syscall return하는 값은 0이다면(입력 숫자가 stack 안에 있는 숫자와 종복되는 뜻) reduce의 값은 1을 더할 것이다. 그러므로 종복이 없다면 reduce는 0이고 push와 pop의 횟수가 같다. 종복이 있다면 reduce의 값은 많아지고 pop의 횟수가 적어질 것이다.



다음에는 push부분이다. 먼저 n이 입력한 데이터를 얻으며 syscall해서 push할 값을 전달한다. r는 push 함수가 return한 값이며 0이라면 reduce++

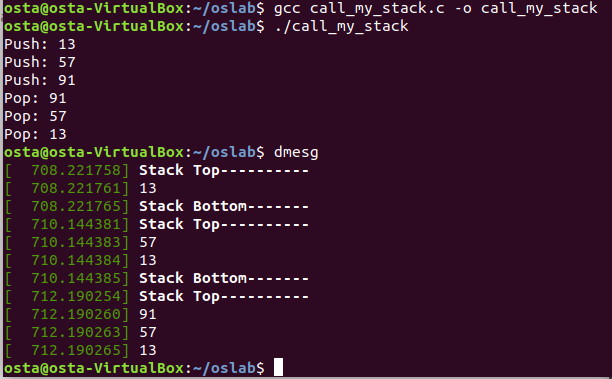


j가 초기화되고 reduce의 값을 더하며 pop함수를 후출한다. print하는 값은 pop한 데이터이다. 마지막 return 0을 해서 끝나다.

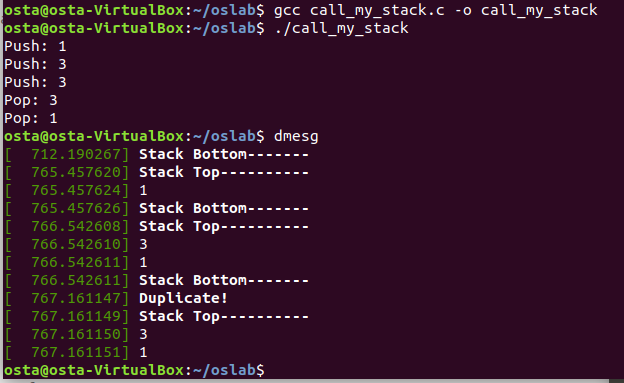


III.실행 결과 스냅샷

push 13, 57, 91



push 1, 3, 3(종복이 있음)



IV.문제점과 해결방법

문제점:

1. kernel에서 프로그래밍해 본 적이 없어서 아무것도 없는 Text Editor를 대하여 어떻게 시작하는지도 몰랐다. 많은 함수를 쓸 수 없고 계속 error가 나왔다.

2. 변수를 어떻게 print하는지를 몰랐다. printf 함수를 쓸 수 없어서 printk(“%d\n”, n)를 쓸 때 오류가 발생하였다.

3. strlib.h 헤더 파일이 안되기 때문에 malloc과 free 함수를 사용할 수 없다.

4. 커널 안에 push 함수에서 선언된 stack는 어떻게 pop함수에서 사용할 수 있다.

해결방법:

1. Hello World를 print하는 것부터 사용 방법을 배운다.

2. printk 함수 안에 KERN\_INFO를 들어서 오류가 발생하지 않았다. 예: printk(KERN\_INFO“%d”,n);

3. <linux/slab.h> 헤더 파일로 kmalloc와 kfree 함수를 사용해서 메모리 분배하고 푸는 작용을 실행하였다.

4. push 함수 안에 stackPtr를 선언하지 않고 전역 변수로 stackPtr를 선언되었다. 그래서 pop 함수에서도 stackPtr를 사용할 수 있다.