운영체제 1차과제

학과	컴퓨터학과
학번	2019320054
이름	고영인
제출날짜	2023/04/11
Freeday 사용	0일

1. 개발환경: window 10 home, Oracle VM, ubuntu-18.04
2. 시스템 콜
   A. 리눅스라는 운영체제는 두 가지 모드에서 함수, 프로그램 등을 수행합니다. 사용자들에게 보이는 user mode와 kernel mode입니다. user mode에서는 GUI, batch, command line 등을 통해서 user application 실행이 됩니다. user mode에서 현재 실행되고 있는 user process에서 syscall.h, unistd.h 라이브러리에 저장되어 있는 system call()을 호출하면 mode switch가 일어나서 kernel mode에 진입하게 됩니다. 이때 유저는 syscall() 함수를 이용해서 호출하려는 syscall 번호와 parameter를 넘겨주게 됩니다. parameter의 개수는 syscall 함수가 선언될 때, SYSCALL_DEFINE{parameter의 숫자}(호출할 syscall name, parameter1의 자료형, prar1의 이름 ..) 이런 방식으로 정해지기 때문에 임의로 parameter의 개수를 추가하거나 뺄 수 없습니다. kernel mode로 전환한 후, kernel은 system call 번호를 확인하고, 해당하는 함수를 실행합니다. 이 system call이 끝나면 다시 user mode로 mode switch가 이루어지고 system call 결과를 반환합니다.
3. 변경, 수정한 파일 설명
   A. syscall_64.tbl 에서 추가한 내용은 다음과 같습니다.

```
335     common  os2023_push             __x64_sys_os2023_push
336     common  os2023_pop              __x64_sys_os2023_pop
```

   335,336은 새로 만드는 시스템콜의 시스템콜 번호입니다. 그리고 common은 시스템콜의 유형 중 하나로 user mode에서 시스템콜이 호출되어 kernel mode로 mode switch를 해서 시스템콜을 처리하는 것입니다. os2023_push, os2023_pop은 새로 추가한 시스템콜의 이름이며, 마지막 __x64_sys_os2023_push, __x64_sys_os2023_pop은 실제로 수행되는 시스템콜의 이름입니다.

   B. syscalls.h에서 추가한 내용은 다음과 같습니다.

```
asmlinkage void sys_os2023_push(int);
asmlinkage int sys_os2023_pop(void);
```

   asmlinkage는 어셈블리에서 직접 호출(link)할 수 있다는 의미이며 이 함수는 커널 소스의 <include/linux/linkage.h>에 선언이 되어있습니다. arm기반의 아키텍쳐에서는 register를 사용해서 함수의 인자를 넘긴다는 규약이 있지만, x86기반의 아키텍쳐는 규약이 정해져 있지 않기 때문에 register 혹은 stack을 통해 parameter를 전달합니다. push는 int를 parameter로 받고 return을 하지 않고, pop은 parameter없이 int를 리턴합니다.

   C. Makefile에서 추가한 내용은 다음과 같습니다.

~~obj-y = async.o range.o smpboot.o ucount.o~~ **oslab_my_stack.o**

   obj-y에 oslab_my_stack.o을 추가해줌으로 작성한 oslab_my_stack.c를 컴파일하고 나서 만들어진 object file의 이름을 oslab_my_stack.o로 지정해줍니다.

   D. 첫째로 만든 파일은 oslab_my_stack.c입니다.

   여기서는 배열을 이용해 stack을 선언해주고 간단하게 push, pop을 구현해주었습니다. push는 int값 하나를 받아와서 stack의 top에 넣어주기 때문에 SYSCALL_DEFINE1(os2023_push, int, a) 이런 식으로 선언해주었습니다. 안에서 size만큼 stack을 돌면서 중복된 값이 있는지 검사하고 있다면 1로 초기화가 되어 있는 d를 0으로 만들어줍니다. 중복된 원소가 없다면 stack의 top(size - 1)에 parameter로 온 a를 넣어주고 size++(top도 +1임) 해줍니다. 그 다음 stack을 top부터 Bottom까지 가며 출력해주는 print_stack() 함수를 호출해줍니다.

pop함수는 parameter가 없기 때문에 SYSCALL_DEFINE0(os2023_pop) 이런 식으로 선언해주었습니다. return 해줄 값을 저장할 ret를 선언해주고 stack에 원소가 있는 경우인 size가 0보다 클 때만 ret에 stack의 top에 있는 원소를 넣어 주고 size를 줄여줍니다. 이때 stack이 비었다면 -1을 리턴하게 해주었습니다. 그 이후 stack을 출력하는 print_stack()을 호출하고 return ret를 해주었습니다.

E. 두번째로 작성한 파일은 oslab_call_stack.c입니다. 이 파일은 user mode에서 작성되었습니다. 이 파일에서는 syscall()을 이용해서 위에서 작성한 함수들을 호출하고 실행합니다. <unistd.h> 라이브러리에 선언된 syscall()함수를 이용해서 kernel mode에 진입을 합니다. os2023_push를 call하는 것을 void push(int a)로 선언된 함수에 넣어서 간단하게 사용하게 만들었습니다. os2023_pop을 call하는 것도 void pop(void)로 선언된 함수에 넣어서 만들었습니다. push()에서는 "Push a" 이런 식으로 출력해주고, pop은 syscall에서 return된 값을 이용해서 "Pop ret" 이런 식으로 출력하게 하였습니다

```
[   37.975037] [System Call] os2023_push :
[   37.975038] Stack Top ----------------------
[   37.975039] 1
[   37.975040] Stack Bottom --------------------
[   37.975042] [System Call] os2023_push :
[   37.975043] Stack Top ----------------------
[   37.975043] 1
[   37.975044] Stack Bottom --------------------
[   37.975046] [System Call] os2023_push :
[   37.975046] Stack Top ----------------------
[   37.975047] 2
[   37.975047] 1
[   37.975047] Stack Bottom --------------------
[   37.975049] [System Call] os2023_push :
[   37.975050] Stack Top ----------------------
[   37.975050] 3
[   37.975051] 2
[   37.975051] 1
[   37.975052] Stack Bottom --------------------
[   37.975052] [System Call] os2023_pop :
[   37.975053] Stack Top ----------------------
[   37.975053] 2
[   37.975054] 1
[   37.975054] Stack Bottom --------------------
[   37.975060] [System Call] os2023_pop :
[   37.975060] Stack Top ----------------------
[   37.975061] 1
[   37.975061] Stack Bottom --------------------
[   37.975063] [System Call] os2023_pop :
[   37.975063] Stack Top ----------------------
[   37.975064] Stack Bottom --------------------
```

```
osta@osta-VirtualBox:~/Desktop$ ./oslab_call_stack
Push 1
Push 1
Push 2
Push 3
Pop 3
Pop 2
Pop 1
```

4. 문제점, 해결방법

우분투에서 자동업데이트를 체크해제를 안해놔서 가상머신을 껐다 키는 과정에서 업데이트가 되어서, syscall()이 제대로 되지 않는 문제점이 있었고, ubuntu를 재설치하고나서 같은 코드로 시도해보니 정상적으로 작동하였습니다.