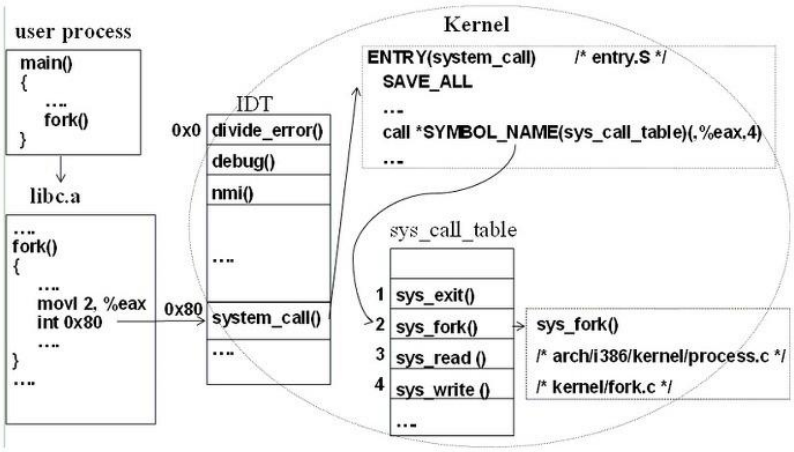
Operating System
Assignment 1

학과: 컴퓨터학과
학번: 2019320137
이름: 황상민
제출일: 2023/04/09
Freeday 사용: 0

1. 개발 환경

   - Host OS: Windows 10 Home 64비트
   - Guest OS: Ubuntu 18.04.2 LTS (커널 버전: 4.20.11)

2. 리눅스 System call 설명



C를 이용할 경우 libc에서 제공하는 Wrapper function의 형태로 system call을 사용하게 되는데, 이러한 경우 system call이 호출되는 routine은 다음과 같습니다.

1) Libc.a에서 호출할 system call의 고유번호를 레지스터에 저장하고(movl 2, %eax), trap(Software interrupt)을 발생시킵니다(int 0x80).

2) 리눅스는 인터럽트 처리 시에 IDT(Interrupt Descriptor Table)를 사용하는데, 이는 각 인터럽트를 처리하는 서비스 루틴의 시작점 주소가 저장된 테이블을 말합니다. 이 테이블을 참조하여 어떤 루틴을 수행해야 할지를 알 수 있습니다. X86 cpu의 경우 0x80번째 interrupt가 system_call로 지정되어 있습니다.

3) System_call()에서는 이전에 레지스터에 저장되었던 고유번호를 이용, system call table을 탐색하여 고유 번호에 맞는 system call 함수를 호출합니다.

   이때 system call 함수에 파라미터를 전달할 수 있는데, 리눅스의 경우에는 메모리에 파라미터를 저장한 뒤 그 시작 주소를 레지스터에 넣어 전달하는 방식을 사용합니다. 이렇게 저장된 파라미터는 커널이 사용할 때 커널 메모리 영역으로 복사된 뒤에 사용됩니다.

4) 함수가 호출되고 서비스가 끝나면 결과값이 반환되고, 그 값이 유저 영역으로 복사됩니다. mode switch가 일어나면서 user process로 실행 흐름이 돌아옵니다.

3. 작성한 코드 설명

1) syscall_64.tbl

시스템 콜 테이블에 구현할 system call을 등록하였습니다. 각각 335번을 push로, 336번을 pop으로 하여 고유 번호를 지정하였습니다.

2) syscalls.h

구현할 system call의 프로토타입을 정의하였습니다. 함수명은 각각 sys_os2023_push와 sys_os2023_pop이며, tutorial에 따라 return type 및 parameter type을 지정하였습니다.

3) oslab_my_stack.c

시스템 콜을 구현하였습니다. 구현에는 <linux syscalls.h> 헤더에 정의된 SYSCALL_DEFINEx라는 매크로 함수를 사용하였습니다. Integer array 형식의 stack을 선언하고, 가장 나중에 들어온원소의 인덱스를 가리키는 변수 top을 두었습니다. (스택이 비어있으면 -1)

Push에서는 먼저 중복 원소가 있는지 검사한 뒤 없는 경우에만 원소를 삽입합니다. 삽입할 때는 top을 1 증가시키고, 그 배열 위치에 원소를 대입하게 됩니다. 마지막에는 dmesg 확인을 위한 출력 구문을 작성하였습니다.

Pop에서는 현재 top 위치에 있는 원소를 반환하고, top을 1 감소시킵니다. 마지막 부분에는 dmesg 확인을 위한 출력 구문이 작성되어 있습니다.

스택의 원소를 출력하는 구문의 경우에는 print_stack 함수로 모듈화하여 작성하였습니다.
* 예제와 동일한 결과를 위해 터미널에 출력되는 글자 색을 바꾸는 포맷을 사용하였습니다.

4) Makefile

Kernel make시에 새로 추가한 부분을 반영하여 컴파일하도록 obj-y 부분에 추가하였습니다.

5) oslab_call_stack.c (user application)

syscall() 매크로 함수를 이용하여 추가한 시스템 콜을 사용합니다. 이때 함수 인자로는 시스템 콜 테이블에 지정했던 번호들을 전달합니다. Push 함수를 호출할 때에는 인자가 필요하므로, 시스템 콜 번호 뒤에 전달할 인자를 추가로 넣습니다.

4. 실행 결과 캡처

```
bbang3@bbang3-VirtualBox:~$ ./oslab_call_stack
Push 1
Push 1
Push 2
Push 3
Pop 3
Pop 2
Pop 1
bbang3@bbang3-VirtualBox:~$
```

```
[14630.081299] [System Call] os2023_push :
[14630.081300] Stack Top --------------------
[14630.081301] 1
[14630.081301] Stack Bottom ------------------
[14630.081310] [System Call] os2023_push :
[14630.081310] Stack Top --------------------
[14630.081311] 1
[14630.081311] Stack Bottom ------------------
[14630.081313] [System Call] os2023_push :
[14630.081314] Stack Top --------------------
[14630.081314] 2
[14630.081314] 1
[14630.081315] Stack Bottom ------------------
[14630.081353] [System Call] os2023_push :
[14630.081354] Stack Top --------------------
[14630.081354] 3
[14630.081355] 2
[14630.081355] 1
[14630.081356] Stack Bottom ------------------
[14630.081361] [System Call] os2023_pop :
[14630.081362] Stack Top --------------------
[14630.081362] 2
[14630.081363] 1
[14630.081363] Stack Bottom ------------------
[14630.081365] [System Call] os2023_pop :
[14630.081365] Stack Top --------------------
[14630.081366] 1
[14630.081366] Stack Bottom ------------------
[14630.081368] [System Call] os2023_pop :
[14630.081368] Stack Top --------------------
[14630.081369] Stack Bottom ------------------
bbang3@bbang3-VirtualBox:~$
```

5. 과제 중 발생한 문제점과 해결 방법

Make, make install로 새로운 커널을 빌드한 후 설치까지 완료하였는데, user application에서 push 및 pop이 전부 제대로 이뤄지지 않는 문제가 있었습니다. syscall의 return value를 확인해보니 -1로 오류를 나타내고 있었고, 추가로 errno를 이용해 오류의 원인을 파악해보니 'invalid system call number'이었습니다. System call table에 추가했음에도 이러한 원인이 일어나는 것으로 보아 커널에 변경 사항이 반영되지 않은 것은 아닌지 의심했고, 재부팅을 해보니 해결되었습니다. 이러한 과정을 통해 커널 설치 후에는 재부팅을 해야 변경 사항이 반영된다는 것을 알게 되었습니다.