운영체제 암기

과제

**1. 시스템 콜**

- 운영체제의 커널이 제공하는 서비스에 대해, 응용프로그램의 요청에 따라 커널에 접근하기 위한 인터페이스.

- 직접 커널에 접근할 수 없기 때문에, API를 통해 시스템 호출에 접근 하게 하는 방법이다.

**2, 인터럽트**

- CPU가 프로세스를 실행 중에, I/O등의 예외상황이 발생하여 처리가 필요한 경우를 알려 처리하도록 하는 것.

- I/O, system call, timeout, division by zero

- 인터럽트가 발생하면 커널 내의 인터럽트 핸들러가 이를 처리.

**3. 시스템 콜 처리 과정**

0) 시스템콜 함수에 접근하고 싶어하는 함수가 호출됨 -> interrupt발생.

1) interrupt 발생 : cpu가 user mode에서 kernel 모드로 넘어감.

2) kernel 메모리에 있는 IDT(interrupt descriptor table)에서 interrupt 번호 참조,

------- interrupt 가 발생하고 이를 처리, idt.c / entry\_32.S : interrupt마다 처리해야 할 일-------------

3) 0x80 : system\_call(), 시스템콜임을 확인하고, system call table를 참조함. 이때 %eax 레지스터에서 function number 가져옴. (57 -> fork)

4) system call table에서 호출해야 할 시스템 콜 함수(sys\_fork)를 확인, 헤더파일로부터 이를 호출하여 실행한다.

결국 C에서 fork를 하고 싶은데, sys\_fork()라는 linux kernel이 제공하는 함수를 이용해야만 함. 따라서 커널에 하고 싶음. 이처럼 interrupt를 발생시키고, system call임을 확인하고, system call table에서 그에 해당하는 함수를 찾아 실행한다. 이 과정, 또는 이런 interface를 시스템 콜이라고 한다. Linux같은 os에서는 응용프로그램에서 이를 잘 활용할 수 있도록 이것들이 다 정의 되어 있음.

**4. 시스템 콜 함수 추가하기**

1) 시스템콜 번호할당 (syscall\_64.tbl : 시스템 콜 테이블)

2) 시스템콜 함수구현 (new\_syscall.c : 새로운 시스템 콜 함수 정의)

3) 시스템콜 함수선언 (syscalls.h : 시스템 콜 함수를 선언하여, 불러올 수 있게끔함.)

4) makefile 변경 : 새로 만든 system call함수가 제대로 컴파일 될 수 있게 해주는 파일.

새로운 커널 만들기 -> 시스템콜이 하나가 추가된 새로운 linux kernel

Dmesg -> 시스템 부팅 메세지 확인

**5, 프로세스**

Process의 생성

- 부모프로세스 : 병렬로 실행 또는 wait()

- 자식프로세스 : 부모와 같은 상태 또는 새로운 프로세스를 로드

fork()

- 부모 프로세스 복제본 생성 (memory - stack, heap, data, text 영역 복사)

- return : 자식 프로세스라면 0을, 부모 프로세스라면 자식프로세스의 id를.

exec()

- 실행하고자 하는 프로그램을 load

- int execl(“실행할 프로그램“, “실행할 명령어(커맨드)“, NULL)

exit()

- void exit(int status)

- 정상적으로 종료되었다면 0이, 아니라면 0 이외의 값이 전달됨.

wait()

- pid\_t wait(int \* status);

- return : (pid\_t) 자식프로세스의 pid, or -1

- parameter : 자식 프로세스가 전달하고자 하는 값을 status에 저장할 수 있음.

**6. 쓰레드,** a light weight process

목적 : multicore, parallel

이점 1) 쓰레드 간의 정보 공유가 편리함, 하나의 core에서 메모리를 공유하기 떄문에.

이점 2) parallel이 가능함.

pthread\_create()

- pthread\_create((pthread\_t \*) &tid, NULL, (void\*) 실행할 함수, (void \*) 함수 인자)

pthread\_exit()

- pthread\_exit(void\* retval)

- 프로세스와 달리, 쓰레드는 종료하며 return 할 값을 인자로 넣어서 전달해줄 수 있음.

pthread\_join

-pthread\_join((pthread\_t) &tid, (void\*\*) retval )

-retval 은 받아온 값을 저장할 인자.

- 기다린다기보단, 특정 thread가 종료되고 retval 을 받아오는 역할.