commands for practice

```
sudo apt install qemu

qemu: 하드웨어 가상화. 이 위에서 xv6를 돌릴 것이다.
```

시스템콜 호출 과정

1. 프로그램 → 시스템콜 호출 → 커널 →하드웨어
2. 프로그램 →라이브러리(프린트등) → 시스템콜 → 커널 - > 하드웨어

프로그램 종류

1. 유저 프로그램(ls.c, echo.c등)
2. 유저 라이브러리(ulib.c, usys.S, user.h)
3. 커널 코드(swtch.S, mp.c 등)

cat 명령어 사용시

read() 함수 호출 : user.h에 선언되어있음.

정의는 usys.S에 되어있고, 어셈블리로 뭐라뭐라 되어있음

그 어셈블리가 시스템콜 인터럽트를 발생시킨다.

인터럽트 발생하면, 인터럽트 핸들러가 syscall.c 에서 eax 레지스터에 시스템콜 번호 처박는다.

시스템콜 만드는 법

int myfunction(char*) 라는 시스템콜을 만드는 과정

1. prac_syscall.c 에 함수와 wrapper함수를 각각 정의한다.
    - (types.h, defs.h 필요)
    - int myfunction(char*), sys_myfunction(void)
        - wrapper 에서는 내부에서 트랩 프레임을 통해 아규먼트를 받아 마이펑션에 직접 넘겨 호출
    - 오브젝트 파일을 prac_sycall.o에 추가한다.
2. defs.h 에 int myfunction(char*)을 선언
    - 다른 c 파일에서 myfunction을 visible하게 만든다.
3. syscall.h에 시스템콜 번호 추가(#defineSYS_myfunction 22),
4. syscall.c : wrapper function을 시스템 콜로 등록하는 과정

    > 💡 extern int sys_myfunction(void); 아래 배열에 [SYS_myfunction] sys_myfunction 등록

5. user.h에 함수 int myfunction(char*); 선언 : 시스템콜이 유저 프로그램에서 보임
6. usys.S에 SYSCALL(myfunction) 매크로 추가

마지막 : make clean, make, make fs.img

usys.S : 그냥 매크로임. 시스템콜이 호출되면 그 이름을 전처리해 syscall.h에 등록된 시스템콜 번호를 가져옴. 그리고 그걸 eax 레지스터에 넣고 시스템콜 인터럽트를 발생시킨다.

syscall.h : 시스템콜 번호 등록; #define SYS_myfunction 22

아!! syscall.c 에서 defs.h를 참조하니까 굳이 아니네!

유저프로그램 만드는 법

user.h, types.h, stat.h 인클루드하고 메인에다가 만들어라.

그리고 myfunction() 호출함(시스템콜 이름을 직접 호출)

Makefile에서

UPROGS에 *my_userapp\ 등록*

Extra에 my_userapp.c 등록

:cs find t getoud