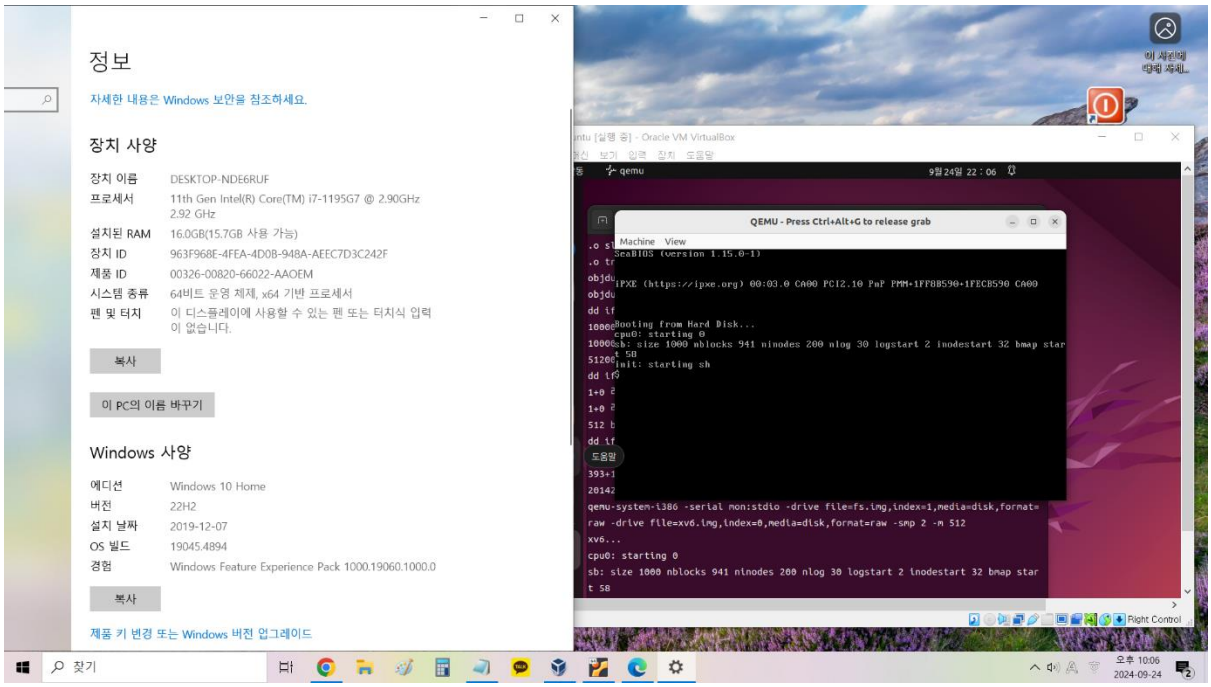
소속 : 소프트웨어학부

학번 : 20221993

이름 : 이재준

# 개발 환경




Lg gram 2022를 사용하고 있습니다. 사양은 위와 같습니다. 운영체제는 virtual machine을 이용한 우분투 리눅스를 사용했습니다.

수정 및 작성한 소스코드에 대한 설명

시스템 콜을 구현하기 위해서는 layering에 대한 이해가 필요합니다. 시스템 콜은 user mode에서 하드웨어를 다루기 위한 인터페이스이고 layer은 인접한 계층과의 통신만이 가능하기 때문입니다. 따라서 하드웨어에서 응용 프로그램까지의 계층을 이어주는 매개가 되어주는 인터페이스를 알아야 시스템 콜을 구현할 수 있습니다.

Russ Cox의 xv6-book-rev11에 따르면 트랩이 발생할 경우 vectors[]에서 트랩의 유형에 맞는 주소를 찾습니다. 그리고 매핑된 alltraps를 통해 trap()에 저장돼 있는 메모리 주소로 접근하여 trap()을 직접 호출합니다. 시스템 콜의 처리 역시 트랩과 유사합니다.

시스템 콜을 실행하면 제어는 적절한 매핑과 인터페이스를 통해 하드웨어 수준까지 계층을 타고 내려간 후에 레지스터에서 syscall()로 다시 올라옵니다. 내려가는 과정을 간략하게 표현하면 다음과 같습니다. 시스템 콜 호출 함수 -> 시스템 콜 호출 인터페이스 -> 시스템 콜 명령어 -> 소프트웨어 인터럽트 -> 커널 모드 -> 알맞은 핸들러 실행 -> 시스템 콜 호출 처리 함수.

과제를 해결하기 위해서는 user application에서 호출하는 forknexec()부터 만들어야 합니다. forknexec()는 프로세스와 관련된 함수이므로 sysproc.c에 작성했습니다. forknexec() 함수는 fork()와 exec()가 합쳐진 함수이므로 xv6에 정의돼 있는 두 함수를 합쳐서 만들었습니다. 과제에서 명시한 return 부분을 비롯한 몇몇 부분만을 수정했습니다.

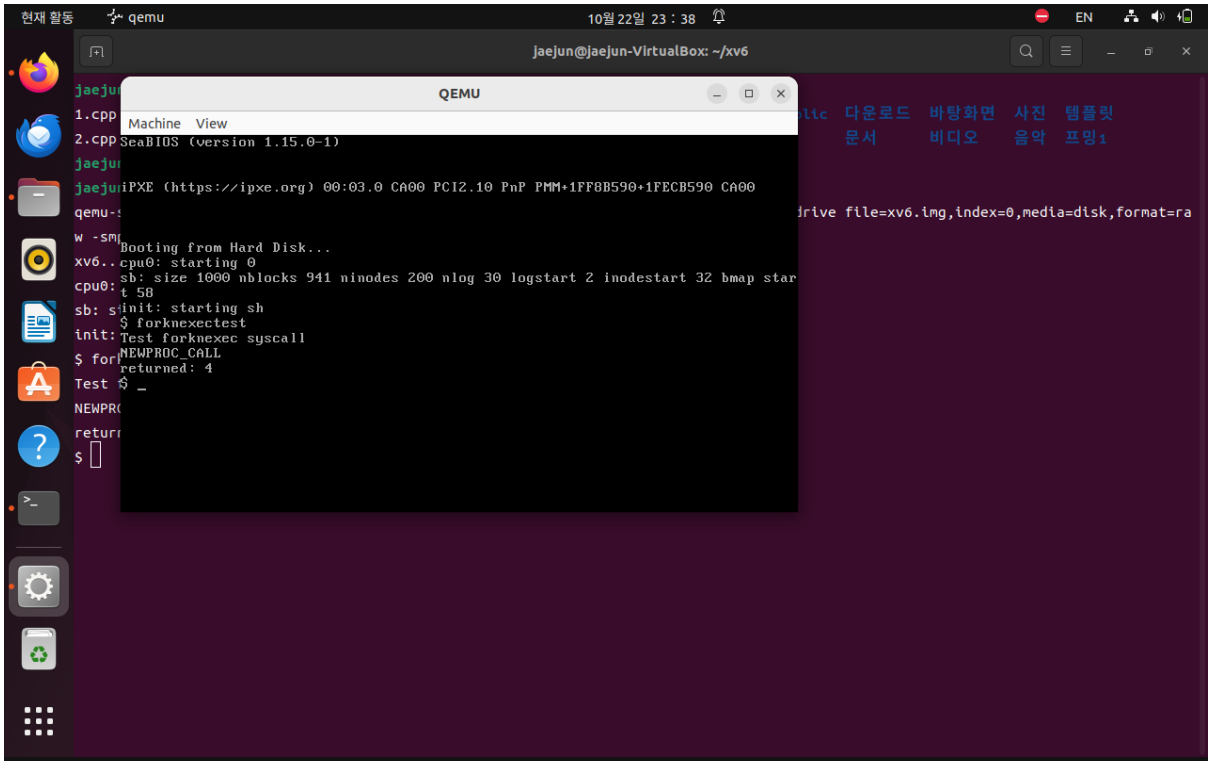
forknexec()에서 시스템 콜 신호 보내면 syscall()이 감지합니다. 정확히는 syscall()이 myproc()를 통해 현재 프로세스의 상태를 확인하고 시스템 콜 신호가 있을 경우 syscall.h와 (*syscalls[])()를 통해 매핑한 함수를 실행합니다. 따라서 위의 항목들을 작성해 주어야 합니다.

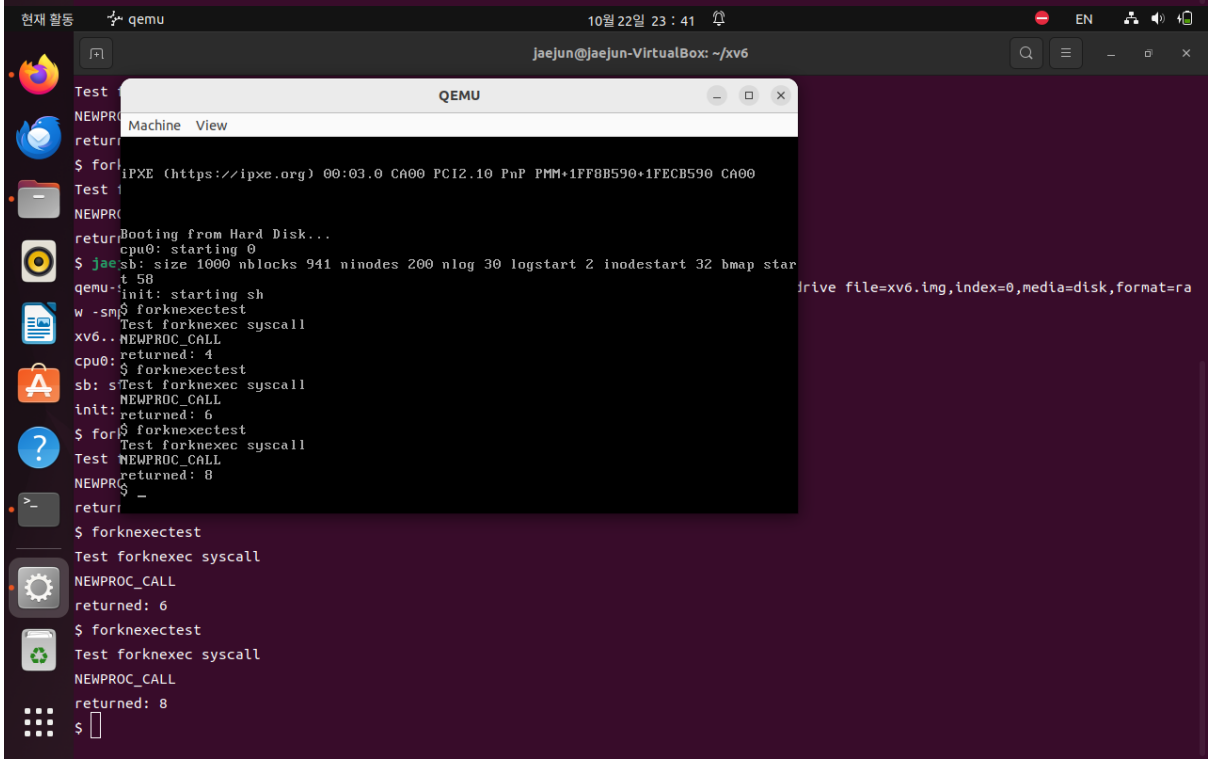
매핑에 이용할 인덱스는 syscall.h에 #define SYS_forknexec 22를 통해 작성했습니다. 인덱스를 통해 호출한 함수는 extern int sys_forknexec(void) 선언을 하고 static int (*syscalls[])(void){}에 [SYS_forknexec] sys_forknexec,를 추가함으로써 구현했습니다. sys_forknexec()은 프로세스 관련 sys call이 정의되어 있는 Sysproc.c에 작성했습니다. 함수 몸체의 경우 정의되어 있는 for와 exec를 참조했습니다. 해당 함수를 작성 후에는 usys.S를 통해 매크로 시스템 콜로 정의했습니다.

| File | Description |
|---|---|
| bio.c | Disk block cache for the file system. |
| console.c | Connect to the user keyboard and screen. |
| entry.S | Very first boot instructions. |
| exec.c | exec() system call. |
| file.c | File descriptor support. |
| fs.c | File system. |
| kalloc.c | Physical page allocator. |
| kernelvec.S | Handle traps from kernel, and timer interrupts. |
| log.c | File system logging and crash recovery. |
| main.c | Control initialization of other modules during boot. |
| pipe.c | Pipes. |
| plic.c | RISC-V interrupt controller. |
| printf.c | Formatted output to the console. |
| proc.c | Processes and scheduling. |
| sleeplock.c | Locks that yield the CPU. |
| spinlock.c | Locks that don't yield the CPU. |
| start.c | Early machine-mode boot code. |
| string.c | C string and byte-array library. |
| swtch.S | Thread switching. |
| syscall.c | Dispatch system calls to handling function. |
| sysfile.c | File-related system calls. |
| sysproc.c | Process-related system calls. |
| trampoline.S | Assembly code to switch between user and kernel. |
| trap.c | C code to handle and return from traps and interrupts. |
| uart.c | Serial-port console device driver. |
| virtio_disk.c | Disk device driver. |
| vm.c | Manage page tables and address spaces. |

Figure 2.2: Xv6 kernel source files.

Xv6 kernel 의 소스 파일




user application 'forknexectest'를 실행한 결과.

리턴된 숫자가 너무 작아서 찾아보니 xv6 의 규모가 작아서 그렇다는 결과를 얻을 수 있었다.

과제 수행 중 발생한 문제점과 해결 방법

모듈 간의 인터페이스가 완벽하지 않아서 처음에는 컴파일되지 않았습니다. 그래서 xv6-book-rev11을 다시 읽어봤습니다. 그리고 defs.h에 함수 원형을 추가하고 user.h에 함수 원형을 선언해줌으로써 해결할 수 있었습니다. defs.h는 xv6에서 함수 원형과 커널 함수를 정의하는 헤더로 인터페이스 역할을 합니다. user.h는 과제에서 정의한 user application에 참조하는 헤더라서 추가해봤습니다. user.h에는 // system calls 항목이 있기도 했습니다.

참조

xv6-book-rev11, Russ Cox(2018)