

Operating System

HW2

(라) 분반

IT대학 소프트웨어학부

20220221 소희연

1. 개발 환경

- 운영체제
 - MacOS(M2)
- AWS E2C (Elastic Compute Cloud), Ubuntu Server 22.04 LTS
 - AWS에서 제공하는 가상 서버를 이용해 인스턴스를 생성하였고, 인스턴스 상에 Ubuntu 22.04를 올려 과제를 수행함.
- SSH를 통해 로컬 터미널에서 xv6 구동

2. 수정 및 작성한 소스코드에 대한 설명

2.1 과제 명세 개요

이번 과제의 핵심 사항은 이미 존재하는 fork와 exec 시스템 콜을 결합하여 ‘forknexec’이라는 새로운 시스템 콜을 구현하는 데에 그 의의가 있으며, 다음의 세 가지 사항을 목표로 한다.

(1) 새로운 시스템 콜 forknexec 구현

- fork와 exec의 역할을 하나로 결합
- 자식 프로세스를 생성하고, 부모 프로세스로부터 분리되어 별도의 파일(프로그램) 경로 실행

(2) 프로세스 생성 및 실행

- 부모 프로세스로부터 자식 프로세스를 생성(fork)하고, 그 자식 프로세스는 특정 경로의 프로그램을 실행(exec)함
- 프로그램 실행에 필요한 경로(path)와 전달 인자(argv[]) 처리 구현

(3) 에러 처리

- 전달 인자나 프로그램 경로가 잘못 되었을 경우 적절한 값(에러 코드) 리턴
- 시스템 콜 실행이 정상적으로 이루어지지 않은 경우의 에러 처리 또한 필요

과제 수행에 있어 상기한 구현 요구 사항 말고 가장 중요하다고 생각하는 점은 가상 메모리 시스템을 이해하고, 프로세스의 독립적인 주소 공간을 어떻게 설정하고 관리할지에 관해 많은 고민을 요구했다는 점이다. forknexec 시스템 콜은 프로세스의 메모리 구조를 설정하고 관리하는 작업을 하기 때문에 가상 메모리 관리와 프로세스에 대한 지식이 꼭 선행되어야 한다.

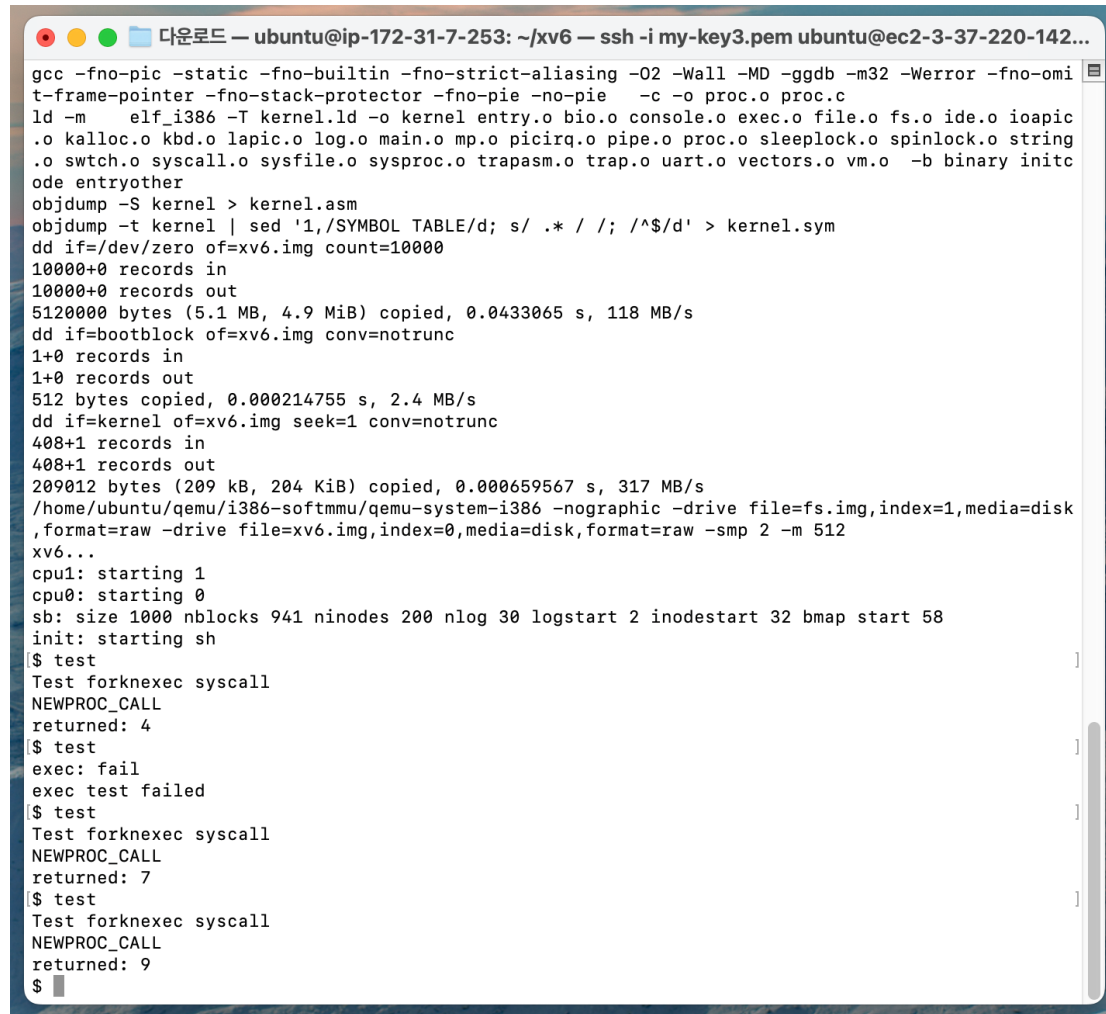
2.2 수정 및 작성한 소스코드

저번 과제와 다르게 수정할 파일들이 많았다. exec()은 별도의 파일(exec.c)에 정의되어 있었고 fork()는 proc.c에 정의가 되어 있는데, forknexec()은 어디에 정의를 해야 할지 고민이 되었다. exec()처럼 별도의 파일로 빼려면 의존성 문제가 생길 것 같아 proc.c 마지막 하단에 정의하였다.

- Makefile: 과제1과 동일하게 test.c와 _test 추가
- test.c: 과제 명세에 제시된 user application 추가
- syscall.h: #define SYS_forknexec 22 추가
- syscall.c: extern int sys_forknexec(void)와 [SYS_forknexec] sys_forknexec 추가
- sysfile.c: sys_forknexec syscall 래퍼 함수 추가
- usys.S: SYSCALL(forknexec) 추가
- user.h: int forknexec(char*, char**) 추가
- defs.h: //proc.c 부분에 int forknexec(char , char*) 추가
- proc.c: forkexec 함수와 헤더 추가

총 9개의 파일을 수정 및 추가했으며, 세부 사항은 소스코드의 각주로 추가하였다.

2.3 실행 화면



```
다운로드 — ubuntu@ip-172-31-7-253: ~/xv6 — ssh -i my-key3.pem ubuntu@ec2-3-37-220-142...
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie -c -o proc.o proc.c
ld -m elf_i386 -T kernel.ld -o kernel entry.o bio.o console.o exec.o file.o fs.o ide.o ioapic.o kalloc.o kbd.o lapic.o log.o main.o mp.o picirq.o pipe.o proc.o sleeplock.o spinlock.o string.o switch.o syscall.o sysfile.o sysproc.o trapasm.o trap.o uart.o vectors.o vm.o -b binary initcode entryother
objdump -S kernel > kernel.asm
objdump -t kernel | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > kernel.sym
dd if=/dev/zero of=xv6.img count=10000
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.0433065 s, 118 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes copied, 0.000214755 s, 2.4 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
408+1 records in
408+1 records out
209012 bytes (209 kB, 204 KiB) copied, 0.000659567 s, 317 MB/s
/home/ubuntu/qemu/i386-softmmu/qemu-system-i386 -nographic -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ test
Test forknexec syscall
NEWPROC_CALL
returned: 4
$ test
exec: fail
exec test failed
$ test
Test forknexec syscall
NEWPROC_CALL
returned: 7
$ test
Test forknexec syscall
NEWPROC_CALL
returned: 9
$
```

정상적으로 실행되는 것을 볼 수 있다.

3. 과제 수행 중 발생한 문제점과 해결 방법

```
proc.c:537:5: error: conflicting types for 'forknexec'; have 'int(const char *, const char **)'
537 | int forknexec(const char *path, const char **argv) {
    |         ^~~~~~
In file included from proc.c:2:
defs.h:123:17: note: previous declaration of 'forknexec' with type 'int(char *, char **)'
123 | int forknexec(char *, char**);
    |         ^~~~~~
```

상기 에러가 계속 생겨 찾아보니 defs.h 파일의 forknexec 함수 매개변수 타입이 proc.c 구현부와 동일하지 않아 발생한 문제인 것을 알게 되었다. defs.h 매개변수를 char *, char**로, proc.c 구현부의 매개변수를 const char *path, const char **argv로 수정하여 에러를 해결하였다.

또한 forknexec 함수 원형의 매개변수나 args[]에 const가 붙어있어 계속해서 에러가 발생했는데, 이는 (void*)로 형 변환을 하여 해결하였다.