Operating System Assignment1

2023017856 한진호

1. Design

kernel 파일 내에서 system call (getppid)를 만들고자 getppid를 정의하는 c파일 (my_syscall.c)을 만들었다.

내가 정의한 getppid를 system call로 만들고자 아래와 같이 진행하였다.

1) Makefile 내에 $K/my_syscall.o를 추가 후 make clean, make | grep my_syscall 명령어 입력

2) defs.h (kernel/defs.h)내에 int getppid(void); 추가

3) 위 과제에서는 parameter를 받아 system call에서 사용하지 않기에 꼭 필요한 작업은 아니나, 연습을 위해 getppid에 wrapper function (sys_getppid)을 취했다.

4) syscall.h (kernel/syscall.h)내에 #define SYS_getppid 22 추가,
syscall.c (kernel/syscall.c)내에 [SYS_getppid] sys_getppid, }; 추가

5) user.h (user/user.h)내에 int getppid(void); 추가

6) usys.pl (user/usys.pl)내에 entry("getppid"); 추가

그 후 내가 정의한 getppid가 의도대로 동작하는지 확인하기 위해 user 파일 내에 user program (ppid.c)을 작성하였다.

make qemu 후 ppid 명령어로 해당 프로그램을 실행시키고자 Makefile 내에 $U/_ppid\ 를 추가하였다.

2. Implementation

1) kernel 파일 내 my_syscall.c

```
#include "types.h"
#include "riscv.h"
#include "defs.h"
#include "param.h"
#include "spinlock.h"
#include "proc.h"

int getppid(void)
{
        return myproc()->parent->pid;
}

int sys_getppid(void)
{
        return getppid();
}
~
```

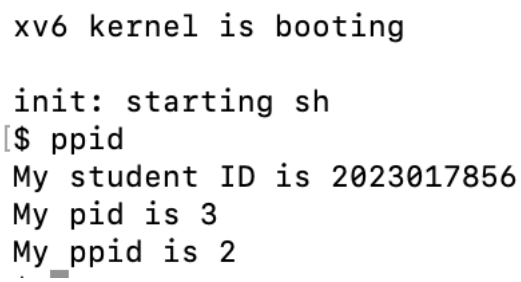
현재 실행되는 프로세스의 parent process에 대한 정보를 얻고자 struct proc를 이용하였다. myproc()로 현재 실행되는 프로세스에 대한 object를 만들고 해당 object 내 parent process를 가리키는 member variable인 parent에 접근한 후 parent 내 pid에 접근하여 parent process의 pid를 가져왔다.

2) user 파일 내 ppid.c

```
#include "kernel/types.h"
#include "kernel/stat.h"
#include "user/user.h"

int main()
{
    printf("My student ID is 2023017856\n");

    printf("My pid is ");
    int pid = getpid();
    printf("%d\n",pid);

    printf("My ppid is ");
    int ppid = getppid();
    printf("%d\n",ppid);
}
```

Assignment 내 example 형식을 따라 프로그램을 작성하였으며, getpid()를 통해 현재 process의 pid, 앞서 정의한 getppid()를 통해 현재 process의 parent process pid를 가져와 print 하였다.

3) Results



4) Troubleshooting

초기에는 parent process에 대해 어떻게 접근하는지 방법을 몰라 fork()를 사용하여 현재 프로세스가 parent process가 되어 이를 getpid()로 pid를 가져오고, fork의 return value로 child process의 pid를 가져오는 방법으로 design하였다.

하지만 이렇게 구현한다면 원래 의도인 현재 프로세스의 parent process의 pid를 가져오는 것이 아니기에 어떻게 design하여야 할지 search하며 process에 대한 struct가 있다는 것을 발견하였다.

이 struct를 사용하기 위해 #include "proc.h"를 하였으나 아래와 같은 error message가 출력되며 프로그램이 실행되지 않았다.

kernel/proc.h:29:24: error: 'NCPU' undeclared here (not in a function) 29 | extern struct cpu cpus[NCPU]; | ^~~~ kernel/proc.h:86:19: error: field 'lock' has incomplete type 86 | struct spinlock lock; | ^~~~ kernel/proc.h:104:22: error: 'NOFILE' undeclared here (not in a function) 104 | struct file *ofile[NOFILE]; // Open files | ^~~~~~ make: *** [kernel/my_syscall.o] Error

이 문제를 해결하기 위해 vim proc.h로 proc.h 내용을 관찰하였다.

위 error message의 내용과 같이 proc.h 내에 NCPU, NOFILE의 값이 정의되어 있지 않았으며 spinlock lock에 대해서도 별다른 내용이 작성되어 있지 않았다.

처음에는 이 문제를 해결하기 위해 proc.h 내에 NCPU, NOFILE에 대한 정보가 들어있는 param.h와 lock이 정의되어 있는 spinlock.h를 include 하였으나 문제가 해결되지 않았으나, 내가 작성한 프로그램 (ppid.c) 내에 #include "param.h" , #include "spinlock.h" 를 하여 해당 문제를 해결하였고 내 의도대로 프로그램이 동작하도록 할 수 있었다.