ProcFS Module Compiling REPORT

201514768

임유택

- Makefile

```
obj-m += mod_proc.o

all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

ppt에 있는대로 Makefile 작성

- modread

```
static ssize_t foo_proc_modread(struct file *file, char __user *user, size_t size_t1, loff_t *loff_t1)
{

	/*down(&read);
	up(&read);
	down(&rmutex);
	readcount++;
	if(readcount == 1)
		down(&wrt);
	up(&rmutex);*/ //2솔루션

	down(&mutex);
	if(wwc>0 || writecount>0) {
		rwc++;
		up(&mutex);
		down(&read);
		down(&mutex);
		rwc--;
	}
	readcount++;
	up(&mutex); //3솔루션


	/*down(&read);
	readcount++;
	if(readcount == 1)
	down(&wrt);
	up(&read);*/ //1솔루션

	re = ring[count % 5];
	read_iterator++;

	/*down(&read);
	readcount--;
	if(readcount == 0)
	up(&wrt);
	up(&read);*/ //1솔루션

	down(&mutex);
	readcount--;

	if(readcount==0 && wwc>0)
		up(&wrt);
	up(&mutex); //3솔루션

	/*down(&rmutex);
	readcount--;
	if(readcount == 0)
		up(&wrt);

	up(&rmutex);*/ //2솔루션

	return 0;
}
```

- modwrite

```
static ssize_t foo_proc_modwrite(struct file *file, const char __user *user, size_t size_t1, loff_t *loff_t1)
{
        if(count > 4 )
                return size_t1;

        /*down(&wmutex);
        writecount++;
        if (writecount == 1)
                 down(&read);
        up(&wmutex);
        down(&wrt);*/ //2솔루션


        down(&mutex);
        if(readcount>0 || writecount>0 || rwc>0 || wwc>0) {
                wwc++;
                up(&mutex);
                down(&wrt);
                down(&mutex);
                wwc--;
        }
        writecount++;
        up(&mutex); //3솔루션


        //down(&wrt); // 1솔루션
        ring[ring_lp++] = 1;
        if(ring_lp == MAX_CLIP) ring_lp = 0;
        count++;
        write_iterator++;
        //up(&wrt); //1솔루션


        down(&mutex);
        writecount--;

        if (rwc>0) {
                for(k=0;k<rwc;k++)
                        up(&read);
        }
        else if(wwc>0)
                up(&wrt);
        up(&mutex); //3솔루션

        /*up(&wrt);
        down(&wmutex);
         writecount--;
        if (writecount == 0)
                up(&read);
        up(&wmutex);*/ //2솔루션



        return size_t1;
}
```

- modwriteread

```
static ssize_t foo_proc_writer_read(struct file *file, char __user *user, size_t size ,loff_t *g){



        if(count < 1){
                return 0;
        }

        /*down(&wmutex);
        writecount++;
        if (writecount == 1)
                down(&read);
        up(&wmutex);
        down(&wrt);*/ // 2솔루션


        down(&mutex);
        if(readcount>0 || writecount>0 || rwc>0 || wwc>0) {
                wwc++;
                up(&mutex);
                down(&wrt);
                down(&mutex);
                wwc--;
        }
        writecount++;
        up(&mutex); // 3솔루션

        //down(&wrt); //1솔루션
        ring[ring_cp++] = -1;
        if(ring_cp == MAX_CLIP) ring_cp = 0;
        count--;
        write_iterator++;
        //up(&wrt); //1솔루션


        down(&mutex);
        writecount--;
        if (rwc>0) {
                for(k=0;k<rwc;k++)
                        up(&read);
        }
        else if(wwc>0)
                up(&wrt);
        up(&mutex); //3솔루션

        /*up(&wrt);
        down(&wmutex);
        writecount--;
        if (writecount == 0)
                up(&read);
        up(&wmutex);*/ //2솔루션


        return 0;
}
```

- init

```
static int __init foo_init(void)
{
        //sema_init(&wrt, 1); //1, 2 솔루션
        //sema_init(&read, 1); //1, 2 솔루션
        sema_init(&wrt, 0); //3 솔루션
        sema_init(&read, 0); //3 솔루션
        sema_init(&wmutex, 1);
        sema_init(&rmutex, 1);
        sema_init(&mutex, 1); // 3솔루션
        return foo_proc_init();
}
```

- procinitprocwrite

```c
static struct proc_dir_entry *foo_proc_dir = NULL;
static struct proc_dir_entry *foo_proc_read = NULL;
static struct proc_dir_entry *foo_proc_write = NULL;
static struct proc_dir_entry *foo_proc_count = NULL;
static struct proc_dir_entry *foo_proc_dump = NULL;

int foo_proc_init(void)
{
        foo_proc_dir = proc_mkdir("kboard", NULL);
        if (foo_proc_dir == NULL)
        {
                printk("Unable to create /proc/kboard\n");
                return -1;
        }

        foo_proc_read = proc_create("read", 0, foo_proc_dir, &foo_proc_ops_read); /* S_IRUGO */
        foo_proc_write = proc_create("write", 0, foo_proc_dir, &foo_proc_ops_write); /* S_IRUGO */
        foo_proc_count = proc_create("count", 0, foo_proc_dir, &foo_proc_ops_count);
        foo_proc_dump = proc_create("dump", 0, foo_proc_dir, &foo_proc_ops_dump);

        if (foo_proc_read == NULL)
        {
                printk("Unable to create /proc/kboard/read\n");
                remove_proc_entry("kboard", NULL);
                return -1;
        }

        if (foo_proc_write == NULL)
        {
                printk("Unable to create /proc/kboard/write\n");
                remove_proc_entry("kboard", NULL);
                return -1;
        }

        printk(KERN_INFO "Created /proc/kboard/read\n");
        printk(KERN_INFO "Created /proc/kboard/write\n");
        return 0;
}

void foo_proc_exit(void)
{
        /* remove directory and file from procfs */
        remove_proc_subtree("kboard", NULL);

        /* remove proc_dir_entry instance */
        proc_remove(foo_proc_read);
        proc_remove(foo_proc_write);
        proc_remove(foo_proc_count);
        proc_remove(foo_proc_dump);
        proc_remove(foo_proc_dir);

        printk(KERN_INFO "Removed /proc/kboard/read\n");
        printk(KERN_INFO "Removed /proc/kboard/write\n");

}
```

- Solution 1

```
root@os201514768:~/proj2/K-board# gcc -o rwh rwh.c -lpthread
root@os201514768:~/proj2/K-board# insmod mod_proc.ko
root@os201514768:~/proj2/K-board# ./rwh
^C
root@os201514768:~/proj2/K-board# rmmod mod_proc
root@os201514768:~/proj2/K-board# dmesg -c
[ 1367.622740] Created /proc/kboard/read
[ 1367.622741] Created /proc/kboard/write
[ 1381.036327] Removed /proc/kboard/read
[ 1381.036329] Removed /proc/kboard/write
[ 1381.036330] write: 44 read: 817766
root@os201514768:~/proj2/K-board#
```

ppt에 있는 솔루션1 세마포어 이용 시 read가 독점

- Solution 2

```
root@os201514768:~/proj2/K-board# gcc -o rwh rwh.c -lpthread
root@os201514768:~/proj2/K-board# insmod mod_proc.ko
root@os201514768:~/proj2/K-board# ./rwh
^C
root@os201514768:~/proj2/K-board# rmmod mod_proc
root@os201514768:~/proj2/K-board# dmesg -c
[ 1612.109154] Created /proc/kboard/read
[ 1612.109155] Created /proc/kboard/write
[ 1626.723580] Removed /proc/kboard/read
[ 1626.723580] Removed /proc/kboard/write
[ 1626.723581] write: 171615 read: 60
root@os201514768:~/proj2/K-board#
```

Ppt에 있는 솔루션2 세마포어 이용 시 write가 독점

- Solution 3

진행중

어려웠던 점 및 해결 방안

이번 과제는 모든 부분에서 정말 어려웠다. mod_proc에 구조에 맞춰 프로그래밍을 하는 부분부터 너무 어려웠고 교수님께서 예제로 주신 c코드를 계속 봤다. 어떤 부분에서 넘어와 어디로 넘어가는 흐름을 파악하고 지난 k-board를 업그레이드 하기 위해 기존에 구현해놓았던 원형 큐가 구현되어있는 틀을 가지고 프로그래밍했다. 이 때 까다로웠지만 계속 예제 c코드를 보며 구조를 파악하려 노력했다. 그 이후 세마포어를 적용하는 부분도 난관이었다. Init, down과 up을 자체적으로 추가하는 것인지 헤더에 포함되어 있는지 또 그 헤더는 무엇인지 헷갈렸다. 인터넷을 찾아보아도 거의 유저영역에서 세마포어를 적용한 사례밖에 없어서 어려웠다. 그래서 이것저것 시도해보던 중 init, down과 up을 지웠더니 오류가 없어 계속 진행하던 도중 솔루션 1을 찾게 되었다. 솔루션 2와, 솔루션3은 ppt에 있기 때문에 이제 거의 끝났다고 생각했지만 그대로 따라 썼는데 되지 않아 ppt를 보며 무엇이 문제인지 찾았다. 하지만 3개 솔루션 모두 계속 read의 독점이 많았고 교수님께 도움을 청했다 교수님께선 쓰레드 사용 시 file을 한번만 오픈하고 해당 파일 디스크립터를 여러 쓰레드가 동시에 쓸 때 모듈 내 구현에 따라 순서대로 처리되는 경우가 있다고 하셨다. 교수님께서 보내주신 소스를 보니 설명이 약간 이해가 되었다. 그렇게 2번 솔루션까지 해결을 하였고 3번 솔루션은 진행 중에 있다.