# 과제 #2

-K-Board v2: over ProcFS

-소스 코드

1) 원형큐에 들어갈 Queue 구조체이다.

```
struct Queue_info {
        int count;
        int ring;
        struct list_head list;
};
```

2) 원형큐에 데이터를 추가하는 enqueue

linux/list.h에 포함되어 있는 list_add를 사용해서 연결리스트처럼 구현하였다.

os_kboard처럼 원형큐의 원소가 5개를 넘어가면 오류가 나도록 구현하였다.

```
static ssize_t enqueue(struct file *inode, const char __user *gdata, size_t length, loff_t *off_what)
{
        info=kmalloc(sizeof(*info), GFP_KERNEL);

        if(!info)
        {
                printk(KERN_ERR "Memory Allocation Failed\n");
                return 0;
        }

        if(length>5) printk(KERN_ERR "Ring is full!!\n");

        else
        {
                copy_from_user(info->ring, gdata, sizeof(int));
                list_add(&info->list, &queue_list);
                info->count=id++;
                printk(KERN_INFO "%s %d %d\n", __func__, info->ring, id);
        }

        return length;
}
```

3) 원형큐의 Head에 있는 원소 제거하는 dequeue

linux/list.h에 포함되어 있는 list_del함수를 이용해서 구현하였다.

```
static ssize_t dequeue(struct file *inode, char  *gdata, size_t length, loff_t *off_what)
{
        if(list_empty(&info->list))
        {
                printk(KERN_ERR "Ring is empty!\n");
                return -1;
        }
        else
        {
                list_del(&info->list);
                info->count=id--;
        }

        return length;
}
```

4) count를 출력하는 함수

기존의 mod_proc.c에 있던 simple_show와 proc_open을 조금 수정해서 구현하였다.

```
static int count_simple_show(struct seq_file *s, void *unused)
{
        seq_printf(s, "count = %d\n", id);

        return 0;
}

static int count_proc_open(struct inode *inode, struct file *file)
{
        return single_open(file, count_simple_show, NULL);
}
```

5) 원형큐 안에 있는 모든 원소들을 출력하는 함수이다.

linux/list.h에 있는 list_for_each_entry함수를 이용해서 구현하였다.

```
static int dump_simple_show(struct seq_file *s, void *unused)
{
        list_for_each_entry(info, &queue_list, list) {
                printk(KERN_INFO "%02d : %d", info->count, info->ring);
        }
        return 0;
}

static int dump_proc_open(struct inode *inode, struct file *file)
{
        return single_open(file, dump_simple_show, NULL);
}
```

6) os_kboard의 dequeue처럼 한 원소만 출력하는 함수이다.

list_head 변수를 이용해서 head로 간 후 출력하도록 구현하였다.

```
static int reader_show(struct seq_file *s, void *unused)
{
        struct list_head *temp;
        struct Queue_info *p;

        temp=queue_list.prev;
        p=list_entry(temp, struct Queue_info, list);

        printk(KERN_INFO "%02d : %d", p->count, p->ring);
        temp=temp->prev;
        return 0;
}

static int reader_proc_open(struct inode *inode, struct file *file)
{
        return single_open(file, reader_show, NULL);
}
```

-어려웠던 점

1) Queue_info 구조체에서 ring변수를 이용해서 enqueue를 하는 방식으로 구현하려고 하였지만 커널에서 유저로부터 복사해서 다루는 것에서 해결이 되지 않았다.

2) rmmod를 수행할려고 했을 때 Device or Resource is busy라는 오류가 나면서 적재 해제가 되지 않아 lsmod를 수행해보니 Used by가 -1이 되어버려 rmmod에서 에러가 발생했었다. 그래서 reboot를 해보니 제대로 실행되었다.