제목 : p2 과제 1 v2 학번 : 201716472 이름 : 최유진

결과 캡쳐 :

```
root@os201716472:/usr# sudo dmesg -c
root@os201716472:/usr# sudo dmesg -c
root@os201716472:/usr# ./paste
Fail: KBoard is empty
root@os201716472:/usr# sudo dmesg -c
[  153.730553] HCPARK: do_sys_kb_dequeue() CALLED!!
[  153.730555] HCPARK: do_sys_kb_dequeue() ring is empty!!
root@os201716472:/usr# ./copy 111; ./copy 222; ./copy 333;
Copy success: 111
Copy success: 222
Copy success: 333
root@os201716472:/usr# ./paste
Paste success: 111
root@os201716472:/usr# ./copy 444; ./copy 555; ./copy 666; ./copy 777
Copy success: 444
Copy success: 555
Copy success: 666
Fail: KBoard is full or invaild clip
root@os201716472:/usr# sudo dmesg -c
[  181.381578] HCPARK: do_sys_kb_enqueue() CALLED!! clip=111
[  181.383056] HCPARK: do_sys_kb_enqueue() CALLED!! clip=222
[  181.384535] HCPARK: do_sys_kb_enqueue() CALLED!! clip=333
[  185.443790] HCPARK: do_sys_kb_dequeue() CALLED!!
[  320.567167] HCPARK: do_sys_kb_enqueue() CALLED!! clip=444
[  320.569143] HCPARK: do_sys_kb_enqueue() CALLED!! clip=555
[  320.570895] HCPARK: do_sys_kb_enqueue() CALLED!! clip=666
[  320.572809] HCPARK: do_sys_kb_enqueue() CALLED!! clip=777
[  320.572811] HCPARK: do_sys_kb_enqueue() ring is full!
root@os201716472:/usr# ./paste; ./paste; ./paste; ./paste; ./paste
Paste success: 222
Paste success: 333
Paste success: 444
Paste success: 555
Paste success: 666
```

```
root@os201716472:/usr# ./copy -1
Fail: KBoard is full or invaild clip
root@os201716472:/usr# sudo dmesg -c
[  370.494490] HCPARK: do_sys_kb_dequeue() CALLED!!
[  370.496175] HCPARK: do_sys_kb_dequeue() CALLED!!
[  370.497649] HCPARK: do_sys_kb_dequeue() CALLED!!
[  370.499185] HCPARK: do_sys_kb_dequeue() CALLED!!
[  370.500796] HCPARK: do_sys_kb_dequeue() CALLED!!
[  377.804054] HCPARK: do_sys_kb_enqueue() CALLED!! clip=-1
[  377.804080] HCPARK: do_sys_kb_enqueue() invalid integer clip=-1
root@os201716472:/usr#
```

코드 캡쳐 :

os_kboard.c (중요부분):

```
int ring[MAX_CLIP];
int rear = 0;
int front = 0;

long do_sys_kb_enqueue(int item) {

        if(item > 0) {
                if((rear + 1) % MAX_CLIP == front) {
                        printk(KERN_DEBUG "HCPARK: do_sys_kb_enqueue() CALLED!! clip=%d\n", item);
                        printk(KERN_DEBUG "HCPARK: do_sys_kb_enqueue() ring is full!\n");

                        return -1;
                }

                else {
                        rear = (rear+1) % MAX_CLIP;
                        ring[rear] = item;

                        printk(KERN_DEBUG "HCPARK: do_sys_kb_enqueue() CALLED!! clip=%d\n", item);

                        return 0;
                }
        }
        else {
                printk(KERN_DEBUG "HCPARK: do_sys_kb_enqueue() CALLED!! clip=%d\n", item);
                printk(KERN_DEBUG "HCPARK: do_sys_kb_enqueue() invalid integer clip=%d\n", item);
                return -2;
        }
}
```

```
long do_sys_kb_dequeue(int *user_buf) {

        if(rear == front) {
                printk(KERN_DEBUG "HCPARK: do_sys_kb_dequeue() CALLED!!\n");
                printk(KERN_DEBUG "HCPARK: do_sys_kb_dequeue() ring is empty!!\n");
                return -1;
        }

        else {
                front = (front + 1) % MAX_CLIP;

                printk(KERN_DEBUG "HCPARK: do_sys_kb_dequeue() CALLED!!\n");
                copy_to_user(user_buf, &ring[front], sizeof(ring[front]));

                return 0;
        }
}
```

copy.c(중요부분) :

```c
int main(int argc, char *argv[])
{
        int i, n;

        for(int j=1; j<argc; j++) {
                n = atoi(argv[j]);
                i = kboard_copy(n);

                if(i == -1 || i == -2)
                        printf("Fail: KBoard is full or invaild clip\n");
                else
                        printf("Copy success: %d\n", n);

        }

}
```

paste.c(중요부분) :

```c
#define MAX_CLIP 6

int main()
{
        int *ring = malloc(sizeof(int)*MAX_CLIP);
        int n;

        n = kboard_paste(ring);

        if(n == -1) {
                printf("Fail: KBoard is empty\n");
        }
        else if(n == 0) {
                printf("Paste success: %d\n", *ring);
        }
}
```

kboard.h(중요부분) :

```c
#include <stdio.h>

long kboard_copy(int clip);
int kboard_paste(int *clip);

long kboard_copy(int clip) {
        int i =  syscall(335, clip);

        return i;
}

int kboard_paste(int *clip) {
        return syscall(336, clip);
}
```

어려웠던 점 : ./copy를 실행했을 때 FULL인 경우 경고가 뜨지 않고 계속 ring buffer에 값이 들어가는 문제가 있어 코드를 여러 번 고쳐야 했다.

해결 방안 : kboard_copy에 변수를 받는 것을 for문 안으로 넣어 ring buffer에 제대로 들어 갈 수 있도록 하였고, make –j 24 만 실행하고 make –j 24 install을 실행하지 않았었기 때문에 다시 실행해보았다.

어려웠던 점 : copy_to_user 함수를 어떤 값들을 넣어 받아야 하는지 알 수 없어서 여러 번 컴파일을 시도해야 했다.

해결 방안 : 여러 변수를 넣어보고 컴파일을 실행해 본 결과 copy_to_user(user_buf, &ring[front], sizeof(ring[front]);가 적절한 값이었다.

어려웠던 점 : paste.c에서 paste success값을 출력할 때 return 값인 0이 출력되었다.

해결 방안 : ring 배열을 사용하기 위해 malloc을 사용하여 선언하였고, paste success값을 출력하기 위해 ring[front]값을 출력해야 하므로, ring의 포인터 값을 넣어 출력하였다. copy_to_user의 값도 조정하였다. 위에 설명한 바와 값이 조정하였다.

어려웠던 점 : paste 출력시 111만 출력되어 코드를 고쳐봤지만 바뀌지 않았다.

해결 방안 : reboot를 실행하지 않아 생긴 문제였다. reboot를 실행하니 정상적으로 실행되었다.