Embedded System Practice Lab 5

2016311821 한승하

```
static ssize_t iodev_write (struct file *file, const char __user *buf, size_t size, loff_t *loff) {
        int i;
        uint32_t lpn;

        if (*loff & (PAGE_SIZE - 1) || size != PAGE_SIZE )
                return -EINVAL;

        lpn = (uint32_t)(*loff >> PAGE_SHIFT);
        copy_from_user((char*)kbuf, buf, size);

        /*(iodev_write)
                1. Read status register (until not DEV_READY)
                2. Write lpn to LBA_REG
                3. Write kbuf repeatedly (size of 4 bytes)f to BUF_REG
                4. Write WRITE_CMD to CMD_REG
                5. Wait for condition variable (condition)
                6. Increase offset
                7. Return size
         */
        while (readl(idata->reg_base + STATUS_REG) != DEV_READY);       //1

        spin_lock_irq(&wait_q_lock);
        condition = 0;

        //2
        writel(lpn,idata->reg_base + LBA_REG);

        for (i = 0; i < PAGE_SIZE / sizeof(uint32_t); i++ ) {
                writel(kbuf[i],idata->reg_base + BUF_REG); //3
        }

        //4
        writel(WRITE_CMD,idata->reg_base + CMD_REG);

        wait_event_lock_irq(wait_q, condition, wait_q_lock);    //5
        spin_unlock_irq(&wait_q_lock);

        (*loff) += size;        //6

        return size;    //7
}
```

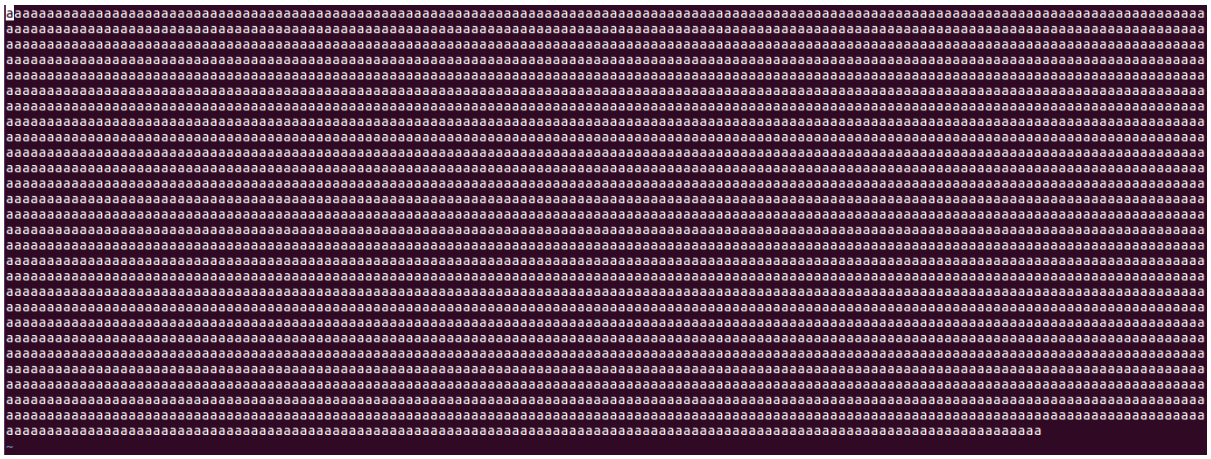
Iodev를 통한 read/write를 위해 protocol 과정에 맞춰 iodev_write함수를 작성한 후, 컴파일 하여 4k_write을 시도하였습니다. 이때 ~/emu의 경로가 실행되지 않아 /home/han/emu의 경로로 iodev_init의 코드를 수정하였습니다.

Iodev write에선 LBA_REG에 lpn을 써주는 과정과, writel을 통해 BUF_REG에 쓰기요청을 하는 과정, CMD_REG에 WRITE_CMD를 전송하는 과정을 작성하였습니다.

```
han@han:~$ ~/my-android-toolchain/bin/x86_64-linux-android-gcc -pie 4k_write.c
4k_write.c:22:8: warning: implicit declaration of function 'write' is invalid in C99 [-Wimplicit-function-declaration]
        ret = write(fd, buf, 4096);
              ^
4k_write.c:26:2: warning: implicit declaration of function 'close' is invalid in C99 [-Wimplicit-function-declaration]
        close(fd);
        ^
2 warnings generated.
han@han:~$ adb push a.out /data/local/tmp
a.out: 1 file pushed, 0 skipped. 0.5 MB/s (7480 bytes in 0.015s)
han@han:~$ adb shell
generic_x86_64:/ $ su
generic_x86_64:/ # ./data/local/tmp/a.out
ret: 4096
generic_x86_64:/ #
```

성공적으로 4096 byte의 character가 쓰여진 모습입니다.

이후에 qemu폴더의 iodev.disk를 확인해 보면



다음과 같이 정상적으로 write가 되어있는 모습을 확인할 수 있었습니다.

이후 Read를 위해 iodev.c를 작성해 주었습니다.

```
static uint32_t goldfish_iodev_read(void* opaque, hwaddr offset)
{
        struct goldfish_iodev_state* s = (struct goldfish_iodev_state*)opaque;
        uint32_t temp;

        if ( offset < 0 ) {
                cpu_abort(cpu_single_env, "iodev_dev_read: Bad offset %" HWADDR_PRIx "\n", offset);
                return 0;
        }

        switch (offset) {
                case IODEV_STATUS_REG:
                        return s->status;
                case IODEV_BUF_REG:
                        return *(s->iodev_buf);
                        break;
        };

        return 0;
}
```

goldfish_iodev_write를 참고하여 Read의 동작으로 올 수 있는 STATUS, BUF에 대한 동작을 정의해 주었습니다.

이후 iodev_read함수를 작성하였습니다.

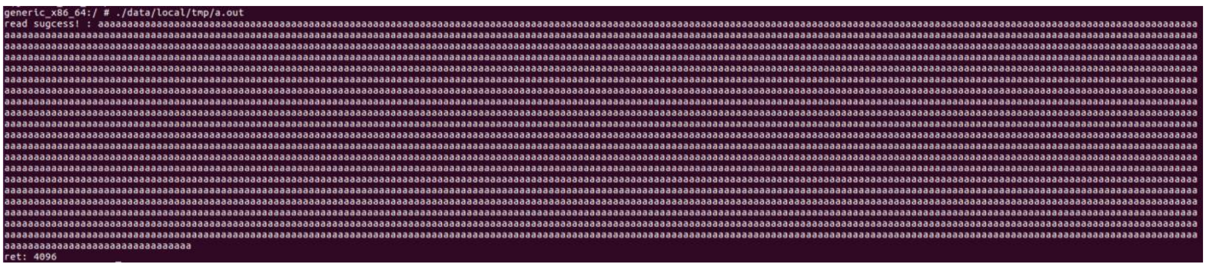
```c
static ssize_t iodev_read (struct file *file, char __user *buf, size_t size, loff_t *loff) {
        int i;
        uint32_t lpn;

        if (*loff & (PAGE_SIZE - 1) || size != PAGE_SIZE )
                return -EINVAL;

        lpn = (uint32_t)(*loff >> PAGE_SHIFT);

        /*(iodev_read)
                1. Read status register (wait if busy)
                2. Write command register
                3. Read buf (check loff whether it aligned to 4K)
                4. Wait for completion (interrupt)
                5. copy_to_user()
         */

        while (readl(idata->reg_base) != DEV_READY); //1

        spin_lock_irq(&wait_q_lock);

        condition = 0;

        writel(lpn,idata->reg_base + LBA_REG);

        writel(READ_CMD,idata->reg_base + CMD_REG);  //2

        wait_event_lock_irq(wait_q, condition, wait_q_lock);
        spin_unlock_irq(&wait_q_lock);

        for (i = 0; i < PAGE_SIZE / sizeof(uint32_t); i++ ) {
                kbuf[i] = readl(idata->reg_base + BUF_REG); //3
        }

        copy_to_user(buf, (char*)kbuf, size);


        return size;
}
```

Iodev read의 protocol순서가 주석과 ppt슬라이드가 달라, ppt를 기준으로 작성하였습니다.

Iodev_write와 유사하게 spin_lock을 잡아주고, LBA_REG에 lpn을 써준 후 READ_CMD를 날리고 신호를 기다려 주었습니다.

이후 kbuf로 data를 읽어 온 후, copy_to_user를 사용하여 user_buffer에 전달해 주었습니다.

이후 emulator를 종료 후 다시 실행시켰을 때



다음과 같이 성공적으로 Read되었습니다.

감사합니다.