OS Project2 report

Programming design

master_device.c

```
void mmap_open(struct vm_area_struct *vma)
{
    /* Do nothing */
}
void mmap_close(struct vm_area_struct *vma)
{
    /* Do nothing */
}
static const struct vm_operations_struct my_vm_ops = {
    .open = mmap_open,
    .close = mmap_close};

static int my_mmap(struct file *file, struct vm_area_struct
*vma)
{
    io_remap_pfn_range(vma,
                       vma->vm_start,
                       virt_to_phys(file->private_data) >>
PAGE_SHIFT,
                       vma->vm_end - vma->vm_start,
                       vma->vm_page_prot);
    vma->vm_ops = &my_vm_ops;
    vma->vm_flags |= VM_RESERVED;
    vma->vm_private_data = file->private_data;
    mmap_open(vma);
    return 0;
}
```

slave_device.c

```
void mmap_open(struct vm_area_struct *vma)
{
    /* Do nothing */
```

```
}
void mmap_close(struct vm_area_struct *vma)
{
    /* Do nothing */
}
static const struct vm_operations_struct my_vm_ops = {
    .open = mmap_open,
    .close = mmap_close};

static int my_mmap(struct file *file, struct vm_area_struct
*vma)
{
    io_remap_pfn_range(vma,
                       vma->vm_start,
                       virt_to_phys(file->private_data) >>
PAGE_SHIFT,
                       vma->vm_end - vma->vm_start,
                       vma->vm_page_prot);
    vma->vm_ops = &my_vm_ops;
    vma->vm_flags |= VM_RESERVED;
    vma->vm_private_data = file->private_data;
    mmap_open(vma);
    return 0;
}
```

user_program master.c

```
    case 'm':
        while (offset < file_size)
        {
            size_t length = PAGE_SIZE;
            if (file_size < length + offset)
            {
                length = file_size - offset;
            }

            file_address = mmap(NULL, length, PROT_READ,
MAP_SHARED, file_fd, offset);
```

```
            kernel_address = mmap(NULL, length, PROT_WRITE,
MAP_SHARED, dev_fd, offset);

            memcpy(kernel_address, file_address, length);
            ioctl(dev_fd, 0x12345678, length);
            ioctl(dev_fd, 0, file_address);

            munmap(file_address, length);
            munmap(kernel_address, length);
            offset += length;
        }
```

user_program slave.c

```
    case 'm':
        while (1)
        {
            ret = ioctl(dev_fd, 0x12345678);
            if (ret == 0)
            {
                file_size = offset;
                break;
            }
            posix_fallocate(file_fd, offset, ret);
            file_address = mmap(NULL, ret, PROT_WRITE,
MAP_SHARED, file_fd, offset);
            kernel_address = mmap(NULL, ret, PROT_READ,
MAP_SHARED, dev_fd, offset);
            memcpy(file_address, kernel_address, ret);
            offset += ret;
            int cnt = 0;
            while (PAGE_SIZE * cnt < ret)
            {
                ioctl(dev_fd, 0, file_address + PAGE_SIZE *
cnt);
                cnt = cnt + 1;
            }
            munmap(file_address, ret);
            munmap(kernel_address, ret);
```

```
        }
        break;
    }
```

The Result

mmap

```
$ sudo ./master ../data/file1_in mmap & sudo ./slave ../data/file1_out mmap 127.0.0.1
[1] 8242
Master Transmission time: 0.028000 ms, File size: 32 bytes
Slave Transmission time: 0.030600 ms, File size: 32 bytes
[1]  + 8242 done       sudo ./master ../data/file1_in mmap
```

```
$ sudo ./master ../data/file2_in mmap & sudo ./slave ../data/file2_out mmap 127.0.0.1
[1] 8279
Master Transmission time: 0.014700 ms, File size: 4619 bytes
Slave Transmission time: 0.057200 ms, File size: 4619 bytes
[1]  + 8279 done       sudo ./master ../data/file2_in mmap
```

```
$ sudo ./master ../data/file3_in mmap & sudo ./slave ../data/file3_out mmap 127.0.0.1
[1] 8313
Master Transmission time: 0.036200 ms, File size: 77566 bytes
Slave Transmission time: 0.082400 ms, File size: 77566 bytes
[1]  + 8313 done       sudo ./master ../data/file3_in mmap
```

```
$ sudo ./master ../data/file4_in mmap & sudo ./slave ../data/file4_out mmap 127.0.0.1
[1] 8327
Master Transmission time: 2.747600 ms, File size: 12022885 bytes
Slave Transmission time: 2.747500 ms, File size: 12022885 bytes
[1]  + 8327 done       sudo ./master ../data/file4_in mmap
```

fcntl

```
$ sudo ./master ../data/file1_in fcntl & sudo ./slave ../data/file1_out fcntl 127.0.0.1
[1] 8365
Master Transmission time: 0.029100 ms, File size: 32 bytes
Slave Transmission time: 0.031400 ms, File size: 32 bytes
[1]  + 8365 done       sudo ./master ../data/file1_in fcntl
```

```
$ sudo ./master ../data/file2_in fcntl & sudo ./slave ../data/file2_out fcntl 127.0.0.1
[1] 8377
Master Transmission time: 0.047800 ms, File size: 4619 bytes
Slave Transmission time: 0.030500 ms, File size: 4619 bytes
[1]  + 8377 done       sudo ./master ../data/file2_in fcntl
```

```
$ sudo ./master ../data/file3_in fcntl & sudo ./slave ../data/file3_out fcntl 127.0.0.1
[1] 8391
Master Transmission time: 0.071200 ms, File size: 77566 bytes
Slave Transmission time: 0.078900 ms, File size: 77566 bytes
[1]  + 8391 done       sudo ./master ../data/file3_in fcntl

$ sudo ./master ../data/file4_in fcntl & sudo ./slave ../data/file4_out fcntl 127.0.0.1
[1] 8403
Master Transmission time: 2.819700 ms, File size: 12022885 bytes
[1]  + 8403 done       sudo ./master ../data/file4_in fcntl
Slave Transmission time: 5.063400 ms, File size: 12022885 bytes
```

Comparison

fcntl 需要先將檔案搬到 buffer 而 mmap 只需要通過對映射的 memory 讀取和修改就能實現對文件的讀取和修改，當檔案較小時優勢比較不明顯，然而當檔案越大時 mmap 就比較有優勢。

Contribution

R07922110 李洋漢 ：all