2019 OS Project 2 – Report

Group 22

1. Design

I. Master side:

From the master side, the file size is already known. In this case, we use the function *mmap* with the given file descriptor as the beginning location and map the input file in 4096 bytes (page size) once a time until the whole file has been read in. In file I/O, a regular buffer is built. With this method, we directly read in the input through virtual location, this process in file I/O can be omitted, which saves the time.

As we call the function memcpy() in the user program, we copy the input file to the master device kernel socket. In master device, as we receive the input data, we call the function ksend() to send the data to the socket.

II. Slave side:

The kernel will allocate 4096 bytes (page size) by default to the object represented by the file descriptor since the file size is unknown to the slave. If the previous allocation is not enough, more space will be mapped until the whole file is received.

To avoid bus error, we first pass a null character to the chosen mapping address. We call *ioctl* to receive the input data in the slave device and ask the kernel to map the data of map_size to the object represented by the file descriptor with the offset. At first, the offset is set to be the file size. After that, every time we do a mapping, we reset the offset to be the new file size. We use memcpy() to copy the input data to the file address accordingly. The slave side receives the data through a while loop until the end of the file.

2. Result

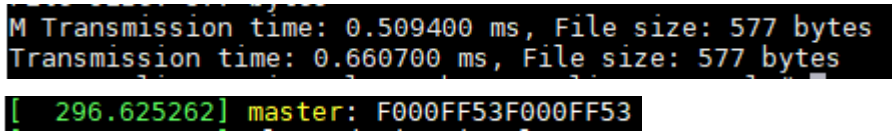
fcntl:

f1:

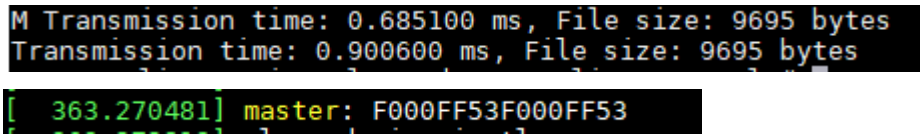
```
Transmission time: 1.241300 ms, File size: 4 bytes
root@nonlinear-VirtualBox:/home/nonlinear/sample# M Transmission time: 1.558700 ms, File size: 4 bytes
```

```
[   49.838931] master: F000FF53F000FF53
```

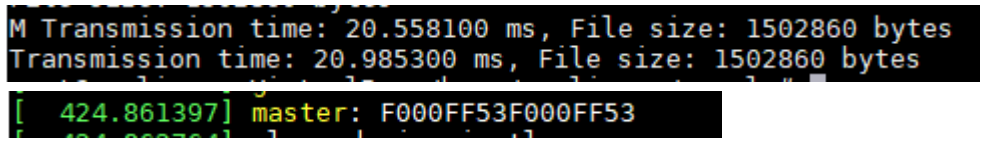
f2



f3

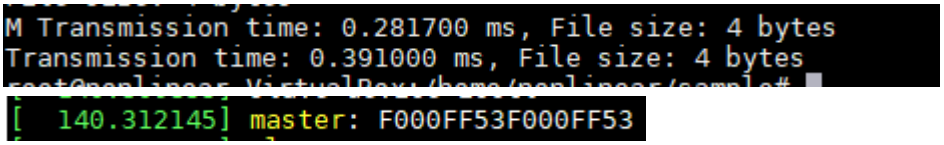


f4

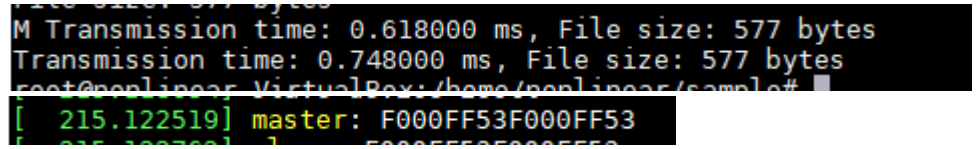


Mmap:

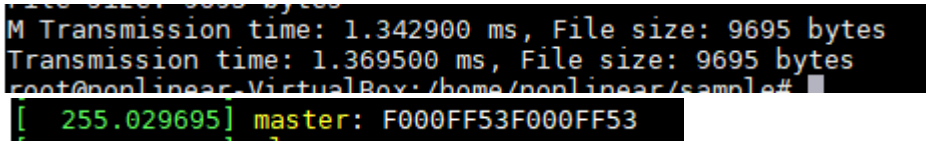
f1:



f2:



f3:



3. Discussion

Generally, memory mapped I/O works better and faster than file I/O. The attribute could be:

I. The division of the file by page size which highly simplifies the operations.

II. Every segment of page size acts like a regular buffer within a process.

III. Copying the whole buffer in file I/O is unnecessary

One of the drawbacks of memory mapped I/O is overheads of *mmap* operation.

In some cases, memory mapped I/O takes longer time compared to *fcntl*. The cause may be that the kernel suffers from too many page faults or the limited page numbers defined by the program. For files of small size, the memory map I/O has the better performance than file I/O, and vice versa.

4. Work division

R07922133 陳則彣：master.c, master_device.c
R07922108 陳鎰龍：slave.c, slave_deveice.c
R07922098 廖經亞：organized data
T07902135 唐宇新：report
B00902039 羅時炘：proper support

5. Reference

Page table management:
https://www.kernel.org/doc/gorman/html/understand/understand006.html
Linux system call (mmap):
http://man7.org/linux/man-pages/man2/mmap.2.html