OS'19S Project 2 -- Synchronous Virtual Device

Group 18

B06902007 王棠葳　B06902010 陳家穎　B06902060 鄒宗霖　B06902062 陳法熏

B06902072 李　謙　B06902080 吳士綸　B06902112 邱睿成

GitHub Repository: https://github.com/slSeanWU/OS19S_Proj_2

壹、設計

以下以 src/ 為主要目錄。

(I) User Programs

在 user_program/目錄底下：

- master.c：主要利用 master device 相關 IO 介面的 program。編譯完成並執行之後將開啟、讀取 disk file 並將其寫入 master device 以傳送出去。
- slave.c：主要利用 master device 相關 IO 介面的 program，需要輸入 master 端的 IP。編譯完成並執行之後將從 slave device 接收檔案並寫到自己的 disk。

(II) Device Drivers

在 master_device/目錄底下：

- master_device.c：master device 的 kernel module，提供給 master 使用，可說是此 character device 的本體。其包括了接收 master 的資料和與 ksocket 建立、溝通(傳送資料)的相關功能。

在 slave_device/目錄底下：

- slave_device.c：slave device 的 kernel module，與上述相似，差在它是從 ksocket 接收資料，並把 data 轉交給 slave。

(III) Ksocket

在 ksocket/目錄底下：

- ksocket.c：負責 master 與 slave 之間的資料傳輸，使兩個 device 能透過 socket 溝通。此資料夾基本上與 sample code 相同，未做任何改動。

在主要目錄執行 compile.sh 之後，將會產生各個模組(master_device, slave_device, ksocket 等)，之後便能在 user_program/底下執行./master 與./slave，並且選定使用 fcntl 或是 mmap 完成 user_programs 與 device 之間資料的傳輸。

此份程式碼各個主要部份大致都已完成，我們的工作主要是將 mmap 的各個相關 function 以及 system call 寫好。

(IV) Implementation of mmap Related Functions

- In user programs (master, slave)

將所有資源都視為 files, 這邊調用 mmap()函式，分別將 disk file 跟 char device 的 data 區段直接映射到 user-space virtual memory，再使用 memcpy()函式，就可以完成資料讀寫。

- In device drivers (master_device, slave_device)

首先需要一塊空間，讓 mmap() 時可以使用，因此我們在 open/close device 時調用了 kmalloc()/kfree()函式，確保 file->private_data 有可用空間。
其次，為了支援 mmap 的操作，在 struct file_operations 裡頭需要新增 my_mmap() 函式，透過 remap_pfn_range()，讓 user 可以用 virtual memory 直接操作 file->private_data 的內容。

貳、執行範例測資結果

(I) Testing Platform

OS: Ubuntu 16.04 LTS w/ Linux Kernel v4.14.25 on VirtualBox 6.0.8

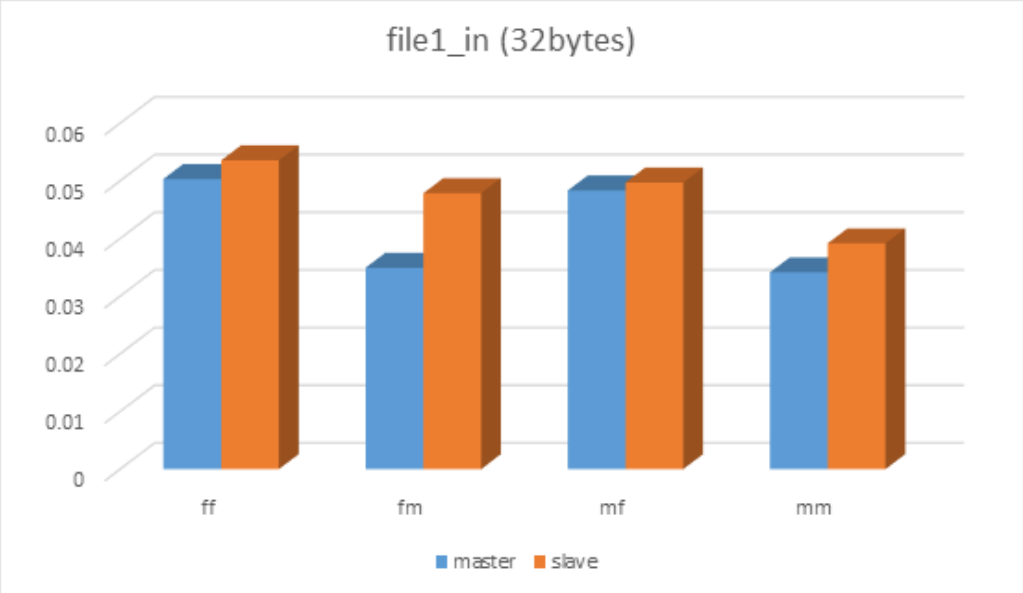
(II) Results

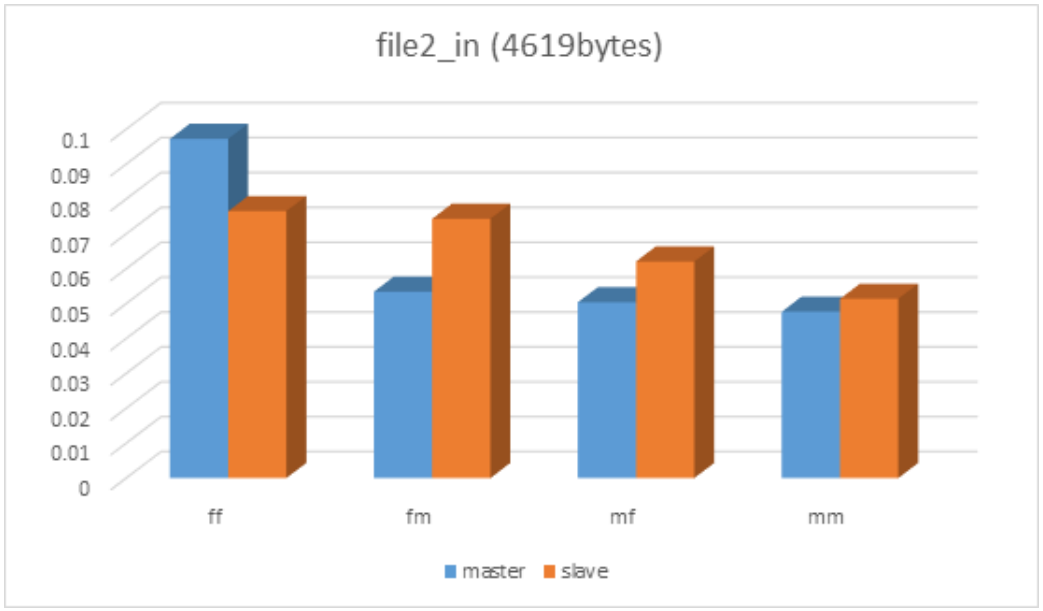
All files can be successfully transmitted under ***Mmap*** and ***Fcntl*** methods.

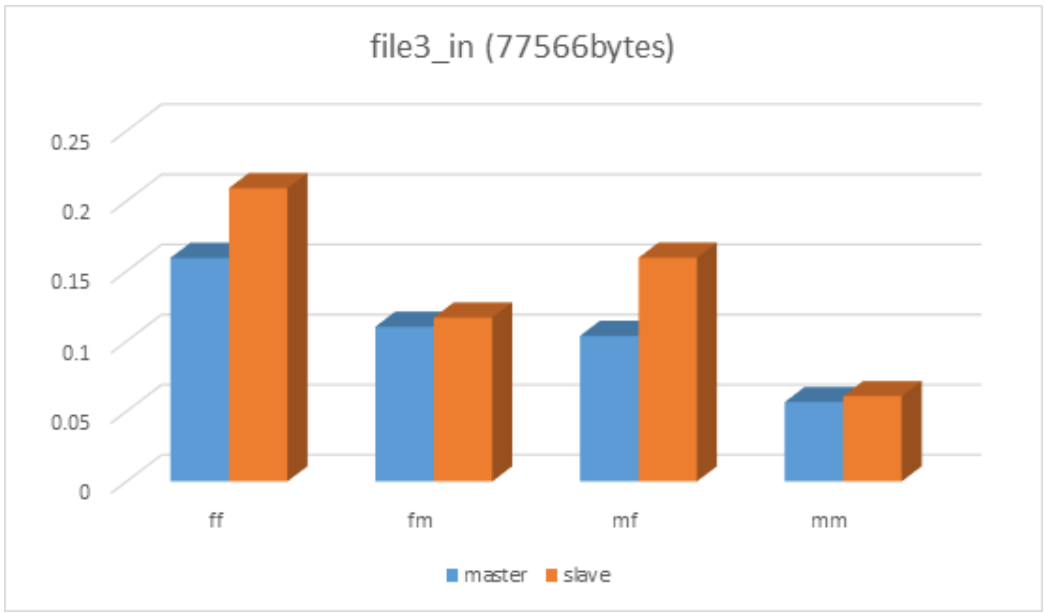
--Use Transmission Time as a Benchmark. (method: master v.s. slave)

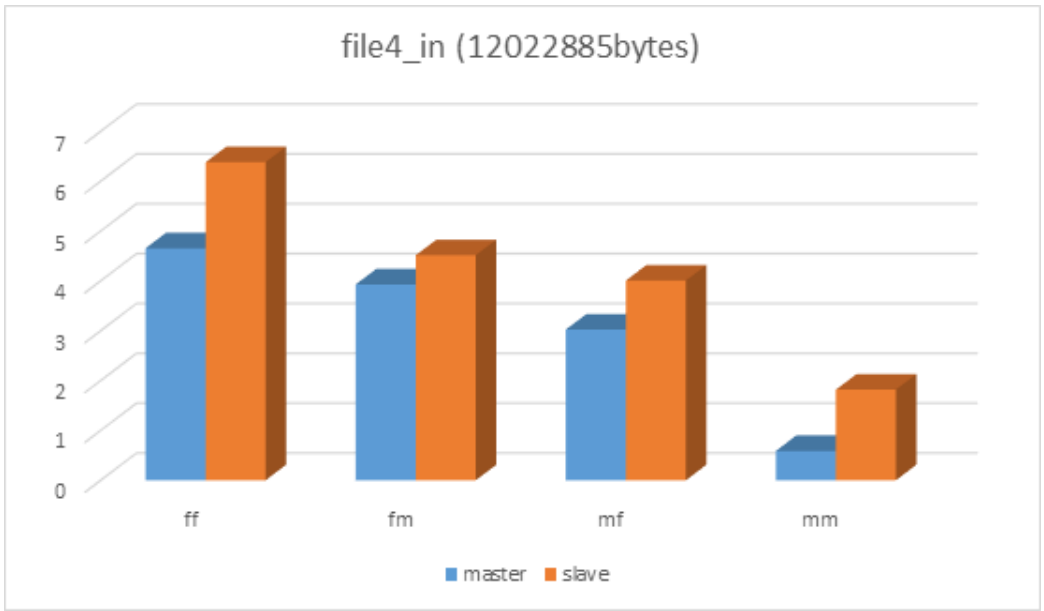
ff : fcntl v.s. fcntl **fm** : fcntl v.s. mmap **mf** : mmap v.s. fcntl **mm** : mmap v.s. mmap

Y-axis unit : millisecond




file2_in (4619bytes)
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
ff
fm
mf
mm
master
slave


file3_in (77566bytes)
0.25
0.2
0.15
0.1
0.05
0
ff
fm
mf
mm
master
slave


file4_in (12022885bytes)
7
6
5
4
3
2
1
0
ff
fm
mf
mm
master
slave

參、結果分析、Mmap I/O 與 File I/O 差異的原因

(I) 結果分析

根據每筆資料測試三十次後平均的結果，若是小檔案如 file1_in、file2_in，執行的時間小於 0.1ms 時，m/m 並沒有明顯的優勢，但隨著檔案越來越大，m/m 與 f/f 逐漸拉開了差距，說明了 Mmap I/O 在大檔案傳遞時，確實能夠提高傳遞的速率。

(II) Mmap I/O 與 File I/O 差異的原因

主要的原因有兩個，第一個原因是 File I/O 作 read、write 時，需要呼叫 system call，但 Mmap I/O 只需要在 call mmap()完成映射之後，操作 local memory 就可以讀取/寫入檔案，而 system call 花費的時間比起更動 local memory 的時間高上幾個數量級；第二個原因是 Mmap I/O 不用把檔案一層層拷貝到 cache, buffer，減少了資料的拷貝次數。

肆、各組員的貢獻

- **Coding**
  - **slave_device.c, slave.c** -- 李謙
  - **master_device.c, master.c** -- 吳士綸
- **Testing**
  - **Automated testing scripts** -- 吳士綸
  - **Result graphing** -- 鄒宗霖
- **Report**
  - 陳法熏、王棠葳、陳家穎、吳士綸、李謙、鄒宗霖、邱睿成

伍、參考資料

- Project website: http://rswiki.csie.org/dokuwiki/courses:107_2:project_2
- Memory Mapping Implementation:
  - https://linux-kernel-labs.github.io/master/labs/memory_mapping.html
  - https://static.lwn.net/images/pdf/LDD3/ch15.pdf