OS Project2: Synchronous Virtual Device

Team: OS 大師兄

Instructor: Chih-Wen Hsueh

亓天毅 *(R07922097)* 鄭維昭 *(R08922131)* 林子欽 *(R08922128)* 傅羿夫 *(R07922109)*
吳冠霖 *(R08922115)*

1 Program Design

1.1 master

輸入參數為 argv[1]: 傳輸方法, argv[2]: 檔案數, argv[3...]: 檔案名稱
例：sudo ./master mmap 2 ./input/file1 ./input/file2
這部分主要工作為讀 input 檔，並決定使用 fnctl 或 mmap 進行傳輸。當成功開啟 file 以及 master device 後，程式進入主迴圈，根據檔案數依次傳送。
以下是針對一個檔案的傳輸:

- block_size：每次要傳輸的 length
- len_package：每次封包確實傳輸的 length
- len_sent：已傳輸的 length
- offset：此檔案尚未傳輸的部份的起點

這邊是一個迴圈，只要 len_sent<file_size，就要進入迴圈執行下一次傳輸，每次傳輸的 length 為 block_size = (1 << SHIFT_ORDER) * PAGE_SIZE，SHIFT_ORDER 設為 5，所以是 128KB；如果此時剩餘檔案 lenght 小於 block_size，則 block_size 為剩餘檔案 length。file_fd 和 dev_fd 經由 mmap 映射到 file_addr 和 device_addr 兩塊 memory，對這兩塊 memory 執行 memcpy 完成 file 和 device 的資料 mapping，再對 dev_fd 執行 ioctl 並得到 ksend 回傳的此次傳送 lengh，計算 len_sent 和 offset 並執行 munmap 取消 mmap 映射，回到迴圈開頭判斷是否需要下一次傳輸。

1.2 master_device

自定義 mmap 方法，當 open 被呼叫時藉由 "___get_free_pages" 分配一塊 kernel memory 大小為 128KB，位址存於 filp->private_data。每次使用者程式呼叫 ioctl 控制碼 "master_IOCTL_MMAP"，master device 即將 filp->private_data 開始的一個 memory block 資料利用 ksend 傳出去。
最後，close 被呼叫時用 free_pages 將這個 kernel memory block 釋放。

1.3 slave

輸入參數為 argv[1]: 傳輸方法, argv[2]:master device ip, argv[3]: 檔案數, argv[4...]: 檔案名稱
例：sudo ./slave 127.0.0.1 mmap 2 ./output/file1 ./output/file2
slave 同樣必須決定使用 fnctl 或 mmap 進行傳輸。當成功開啟空的 output file 以及 slave device 後，程式進入主迴圈，根據檔案數由 slave device 處依次接收 socket 封包。迴圈裡面 ioctl 接收 block_size 大小的封包，利用 posix_fallocated 開啟一個 disk space 確保空間足夠，將 file_fd 和 dev_fd 利用 mmap 映射到 file_addr 和 dev_addr 兩塊 memory，進行 memcpy 並 munmap。迴圈會執行到接收的封包 length=0 即跳出迴圈。迴圈外執行 ftruncate 將檔案裁切到適合大小。

1.4 slave_device

類似於 master device，slave device 同樣自定義 mmap 方法，當 open 被呼叫時藉由 "___get_free_pages" 分配一塊 kernel memory 大小為 128KB，位址存於 filp->private_data。
每次使用者程式呼叫 ioctl 控制碼 "slave_IOCTL_MMAP"，slave device 即從 krecv 接收一個封包，並存入 filp->private_data 開始的一個 memory block。
最後，close 被呼叫時用 free_pages 將這個 kernel memory block 釋放。

2 Performance Comparison

以下實驗當傳輸不同大小的檔案時，master 端和 slave 端在 mmap 及 fnctl 兩種做法下所需耗費的時間，取 3∼5 次實驗平均值：

- master → master_device

File	Size (bytes)	Time-mmap (ms)	Time-fnctl (ms)
file_1	32	0.0438	0.0461
file_2	859	0.0451	0.0511
file_3	1343	0.0453	0.0501
file_4	1663	0.0409	0.0486
file_5	2042	0.0463	0.0408
file_6	3065	0.0453	0.0435
file_7	2808	0.0439	0.0469
file_8	4056	0.0423	0.0445
file_9	3659	0.0403	0.0481
file_10	4619	0.0413	0.0510
file_large	12022885	5.069	14.58
file_xlarge	24045769	8.929	25.26

- slave_device → slave

File	Size (bytes)	Time-mmap (ms)	Time-fnctl (ms)
file_1	32	0.0368	0.0415
file_2	859	0.0406	0.0432
file_3	1343	0.0416	0.0441
file_4	1663	0.0340	0.0412
file_5	2042	0.0422	0.0344
file_6	3065	0.0410	0.0376
file_7	2808	0.0405	0.0441
file_8	4056	0.0412	0.0415
file_9	3659	0.0423	0.0442
file_10	4619	0.0373	0.0439
file_large	12022885	4.785	13.97
file_xlarge	24045769	8.781	24.96

可以發現，當檔案 size 很小時 (約小於等於一個 page, 4096KB)，fnctl 方法的傳輸速度大於 mmap; 但隨著檔案變大，mmap 的速度明顯優於 fnctl。我們認為這是因為 mmap 初始化階段需要 OS kernel 花費額外 overhead 進行分配 kernel space 以及 vma 映射相關資訊導致，並且第一次 access 記憶體位置一定造成 page fault，kernel 會多出 fault handling 的時間。但當檔案較大時，mmap 直接對 virtual memory 操作的速度優勢抵銷 overhead，就可以發揮較佳效率。

3 Work Division

Name	ID	Work Division	Work Load
亓天毅	R07922097	master device	super heavy
鄭維昭	R08922131	slave program	
林子欽	R08922128	master program	
傅羿夫	R07922109	performance comparison	
吳冠霖	R08922115	slave device	

4 Reference

- https://www.xml.com/ldd/chapter/book/ch13.html
- https://blog.csdn.net/qianlong4526888/article/details/8942187
- https://jlmedina123.wordpress.com/2015/04/14/mmap-driver-implementation/