OS Project 2 - Synchronous Virtual Device

組員

B07902004 陳品臻 B07902008 劉倢希 B07902010 陳柏鴻
B07902070 陳昱妤 B07902106 夏 寧 B07902110 張漢芝

設計

Master Program

- 為了可以一次傳送多個檔案，在兩次取時間的函式間加入 for 迴圈，
  每送一個 file 就重新建立一個 socket，等待 slave 建立連線
- 當 offset < file_size（表示尚未完整傳完 data ），重複執行以下步驟：
  1. 決定本次要傳的 data 長度 len
     - 若 file_size - offset < PAGE_SIZE ，表示剩餘檔案小於一個page ，
       因此 len = file_size - offset
     - 若 file_size - offset ≥ PAGE_SIZE ，表示剩餘檔案至少為一個page ，
       因此 len = PAGE_SIZE
  2. 用 mmap 取得兩塊 len 大小的空間位置
     - 第一塊是 input file 的位置（ file address ）
     - 第二塊是 device file 的位置（ kernel address ）
  3. 若為第一次進入 device ，印出 page descriptor
  4. 將大小為 len 的 data 從 file address 複製到 kernel address
  5. offset 加上本次傳的 data 長度 len
  6. 真正將 data 送出給 slave 端
  7. 用 munmap 將 memory 還給系統，結束 mmap

Slave Program

- 為了接收多個檔案，在兩次取時間的函式間加入迴圈，
  每收一個 file 就重新建立一個 socket 連線
- 當 ioctl 仍能接到 data ，表示 master 尚未傳完所有資料，重複執行以下步驟：
  1. 保證接下來 output file 可以在 file_size 後，以 ret 大小寫入資料
  2. 用 mmap 取得兩塊 len 大小的空間
     - 第一塊是 output file 的位置（file address）
     - 第二塊是 device file 的位置（kernel address）
  3. 將大小為 ret 的 data 從 device file 複製到 output file

4. file_size 加上本次接收到的 data 長度 ret
5. 若為第一次進入 device ，印出 page descriptor
6. 用 munmap 將 memory 還給系統，結束 mmap
7. 若收到大小為完整的一個 page_size 則 page_num++ ，若小於 page_size 則只移動指標位置

Device

- 定義了 master 和 slave fops 裡面的 mmap

```
static int master_mmap(struct file *file, struct vm_area_struct *vma);
void mmap_open(struct vm_area_struct *vma) {}
void mmap_close(struct vm_area_struct *vma) {}
static struct file_operations master_fops = {
    .owner = THIS_MODULE,
    .unlocked_ioctl = master_ioctl,
    .open = master_open,
    .write = send_msg,
    .release = master_close,
    .mmap = master_mmap
};
static const struct vm_operations_struct mmap_vm_ops = {
    .open = mmap_open,
    .close = mmap_close
};
static int master_mmap(struct file *file, struct vm_area_struct *vma) {
    if( 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) < 0 ){
            printk("io_remap error.");
            return -1;
    }
    vma->vm_ops = &mmap_vm_ops;
    vma->vm_flags |= VM_RESERVED;
    vma->vm_private_data = file->private_data;
    mmap_open(vma);
    return 0;
}
int master_close(struct inode *inode, struct file *filp)
{
    kfree(filp->private_data);
    return 0;
}
int master_open(struct inode *inode, struct file *filp)
{
    filp->private_data = kmalloc(PAGE_SIZE, GFP_KERNEL);
    return 0;
}
```

```
static int slave_mmap(struct file *file, struct vm_area_struct *vma);
void mmap_open(struct vm_area_struct *vma) {}
void mmap_close(struct vm_area_struct *vma) {}
static struct file_operations slave_fops = {
    .owner = THIS_MODULE,
    .unlocked_ioctl = slave_ioctl,
    .open = slave_open,
    .read = receive_msg,
    .release = slave_close,
    .mmap = slave_mmap
};
static const struct vm_operations_struct mmap_vm_ops = {
    .open = mmap_open,
    .close = mmap_close
};
static int slave_mmap(struct file *file, struct vm_area_struct *vma) {
    if( 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) < 0 ) {
            printk("io_remap error.");
            return -1;
    }
    vma->vm_ops = &mmap_vm_ops;
    vma->vm_flags |= VM_RESERVED;
    vma->vm_private_data = file->private_data;
    mmap_open(vma);
    return 0;
}
int slave_close(struct inode *inode, struct file *filp)
{
    kfree(filp->private_data);
    return 0;
}
int slave_open(struct inode *inode, struct file *filp)
{
    filp->private_data = kmalloc(PAGE_SIZE, GFP_KERNEL);
    return 0;
}
```

- ioctl 內資料傳送接收部分
    - 在 master device 的 `case master_IOCTL_MMAP` 中：

      ```
      ret = ksend(sockfd_cli, file->private_data, ioctl_param, 0);
      ```

    - 在 slave device 的 `case slave_IOCTL_MMAP` 中：

      ```
      ret = krecv(sockfd_cli, file->private_data, PAGE_SIZE, 0);
      ```

比較 file I/O 和 memory-mapped I/O 的結果與效能差異

測試結果

註 ：Demo時，因為中間有更換網路的關係，shell script echo 出的 ip 和實際測試時使用的 ip 不一樣。影片中連線時， `slave` 皆使用 `192.168.43.55`

target_file

- fcntl -> fcntl

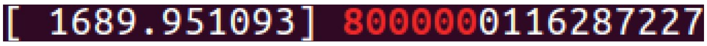
```
./master 1 target_file fcntl
./slave 1 test fcntl 192.168.43.55
```

```
Transmission time: 16997.158500 ms, File size: 12022885 bytes
```

- mmap -> fcntl

```
./master 1 target_file mmap
./slave 1 test fcntl 192.168.43.55
```

```
Transmission time: 18057.518200 ms, File size: 12022885 bytes
```

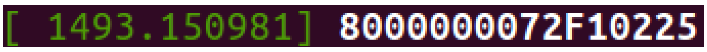
master page descriptors 

```
[ 1689.951093] 8000000116287227
```

- fcntl -> mmap

```
./master 1 target_file fcntl
./slave 1 test mmap 192.168.43.55
```

```
Transmission time: 15016.440200 ms, File size: 12022885 bytes
```

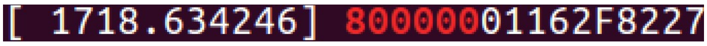
slave page descriptors 

```
[ 1493.150981] 8000000072F10225
```

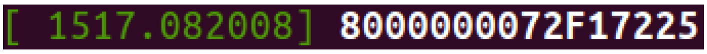
- mmap -> mmap

```
./master 1 target_file mmap
./slave 1 test mmap 192.168.43.55
```

```
Transmission time: 17970.239500 ms, File size: 12022885 bytes
```

master page descriptors 

```
[ 1718.634246] 80000001162F8227
```

slave page descriptors 

```
[ 1517.082008] 8000000072F17225
```

target_file_1, target_file_2, ... target_file_10

- fcntl -> fcntl

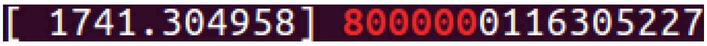
```
./master 10 target_file_1 target_file_2 target_file_3 target_file_4 \
    target_file_5 target_file_6 target_file_7 target_file_8 \
    target_file_9 target_file_10 fcntl
./slave 10 test_1 test_2 test_3 test_4 test_5 test_6 test_7 \
    test_8 test_9 test_10 fcntl 192.168.43.55
```

```
Transmission time: 1066.912300 ms, File size: 24146 bytes
```

- mmap -> fcntl

```
./master 10 target_file_1 target_file_2 target_file_3 target_file_4 \
    target_file_5 target_file_6 target_file_7 target_file_8 \
    target_file_9 target_file_10 mmap
./slave 10 test_1 test_2 test_3 test_4 test_5 test_6 test_7 \
    test_8 test_9 test_10 fcntl 192.168.43.55
```

```
Transmission time: 982.365600 ms, File size: 24146 bytes
```

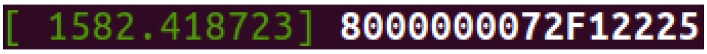
master page descriptors 

```
[ 1741.304958] 8000000116305227
```

- fcntl -> mmap

```
./master 10 target_file_1 target_file_2 target_file_3 target_file_4 \
    target_file_5 target_file_6 target_file_7 target_file_8 \
    target_file_9 target_file_10 fcntl
./slave 10 test_1 test_2 test_3 test_4 test_5 test_6 test_7 \
    test_8 test_9 test_10 mmap 192.168.43.55
```

```
Transmission time: 29.463700 ms, File size: 24146 bytes
```

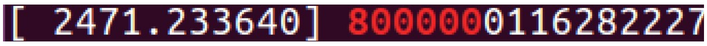
slave page descriptors 

```
[ 1582.418723] 8000000072F12225
```

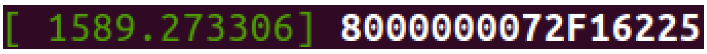
- mmap -> mmap

```
./master 10 target_file_1 target_file_2 target_file_3 target_file_4 \
    target_file_5 target_file_6 target_file_7 target_file_8 \
    target_file_9 target_file_10 fcntl
./slave 10 test_1 test_2 test_3 test_4 test_5 test_6 test_7 \
    test_8 test_9 test_10 mmap 192.168.43.55
```

```
Transmission time: 4005.635700 ms, File size: 24146 bytes
```

master page descriptors 

```
[ 2471.233640] 8000000116282227
```

slave page descriptors 

```
[ 1589.273306] 8000000072F16225
```

in_40960

- fcntl -> fcntl

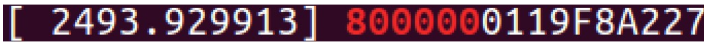
```
./master 1 in_40960 fcntl
./slave 1 out_40960 fcntl 192.168.43.55
```

```
Transmission time: 25.964300 ms, File size: 40960 bytes
```

- mmap -> fcntl

```
./master 1 in_40960 mmap
./slave 1 out_40960 fcntl 192.168.43.55
```

```
Transmission time: 21.367500 ms, File size: 40960 bytes
```

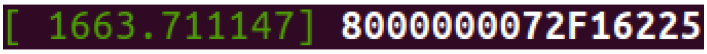
master page descriptors 

```
[ 2493.929913] 8000000119F8A227
```

- fcntl -> mmap

```
./master 1 in_40960 fcntl
./slave 1 out_40960 mmap 192.168.43.55
```

```
Transmission time: 6.055700 ms, File size: 40960 bytes
```

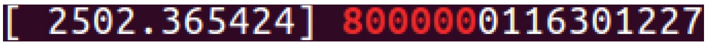
slave page descriptors 

```
[ 1663.711147] 8000000072F16225
```

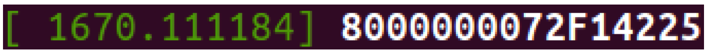
- mmap -> mmap

```
./master 1 in_40960 mmap
./slave 1 out_40960 mmap 192.168.43.55
```

```
Transmission time: 6.163800 ms, File size: 40960 bytes
```

master page descriptors 

```
[ 2502.365424] 8000000116301227
```

slave page descriptors 

```
[ 1670.111184] 8000000072F14225
```

in_1, in_2, ... in_10

- fcntl -> fcntl

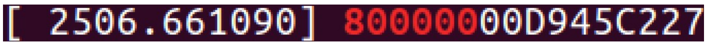
```
./master 10 in_1 in_2 in_3 in_4 in_5 in_6 in_7 in_8 in_9 in_10 fcntl
./slave 10 out_1 out_2 out_3 out_4 out_5 out_6 out_7 out_8 \
    out_9 out_10 fcntl 192.168.43.55
```

```
Transmission time: 30.899900 ms, File size: 40960 bytes
```

- mmap -> fcntl

```
./master 10 in_1 in_2 in_3 in_4 in_5 in_6 in_7 in_8 in_9 in_10 mmap
./slave 10 out_1 out_2 out_3 out_4 out_5 out_6 out_7 out_8 \
    out_9 out_10 fcntl 192.168.43.55
```

```
Transmission time: 1997.062400 ms, File size: 40960 bytes
```

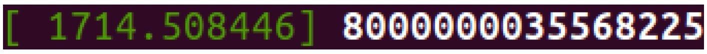
master page descriptors 

```
[ 2506.661090] 80000000D945C227
```

- fcntl -> mmap

```
./master 10 in_1 in_2 in_3 in_4 in_5 in_6 in_7 in_8 in_9 in_10 fcntl
./slave 10 out_1 out_2 out_3 out_4 out_5 out_6 out_7 out_8 \
    out_9 out_10 mmap 192.168.43.55
```

```
Transmission time: 1029.870400 ms, File size: 40960 bytes
```

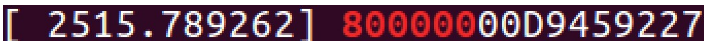
slave page descriptors 

```
[ 1714.508446] 8000000035568225
```

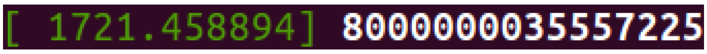
- mmap -> mmap

```
./master 10 in_1 in_2 in_3 in_4 in_5 in_6 in_7 in_8 in_9 in_10 mmap
./slave 10 out_1 out_2 out_3 out_4 out_5 out_6 out_7 out_8 \
    out_9 out_10 mmap 192.168.43.55
```

```
Transmission time: 999.040800 ms, File size: 40960 bytes
```

master page descriptors 

```
[  2515.789262] 80000000D9459227
```

slave page descriptors 

```
[  1721.458894] 8000000035557225
```

結果分析

多檔下比較 **fcntl** 和 **mmap** 兩種 **I/O** 方式

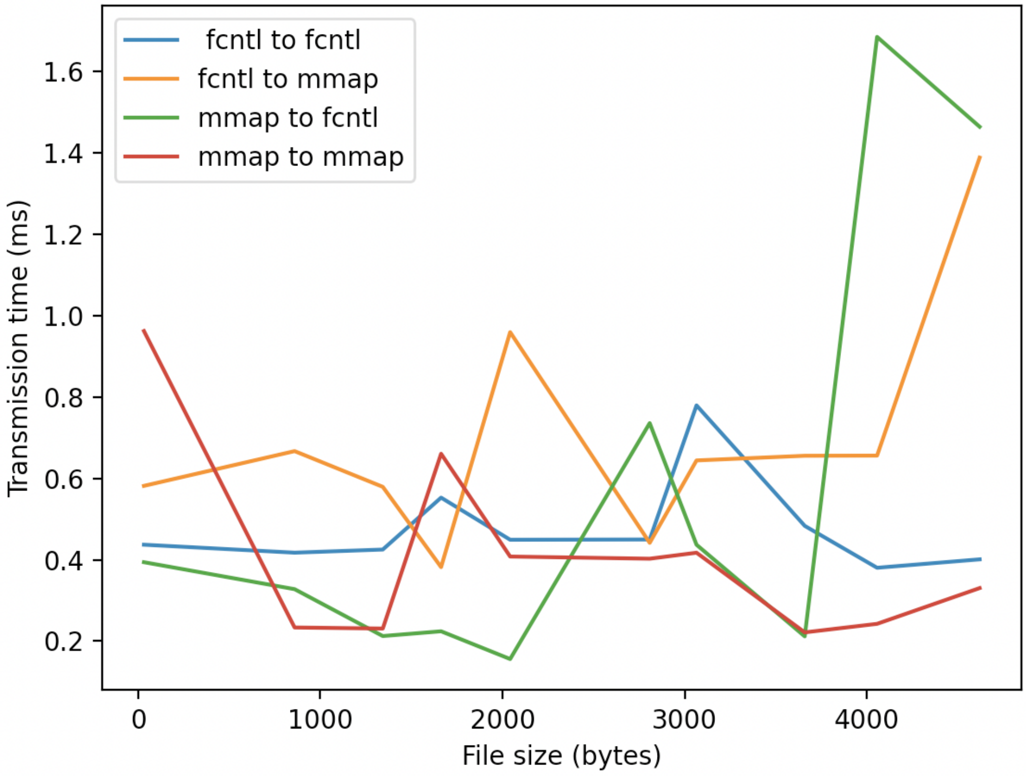
在進行多檔案的傳輸時，fcntl比mmap要快上許多，可能是因為mmap在搬運過程中需進行多次記憶體投放和搬移，故累計耗時較多

比較多檔和單檔傳輸的差異

由於每傳輸一個檔案就要連一次socket，在傳送同樣大小的資料時，以一個檔案傳遞會比以多個檔案傳遞快速

傳送不同的檔案大小

比較不同方法，傳輸不同大小的範例測資（ `target_file_1, ..., target_file_10` ），並繪製成折線圖



由圖可以得知：

- mmap to mmap 或是 fcntl to fcntl 傳輸較為穩定
- 使用 fcntl ，檔案愈大會有傳輸時間愈長的趨勢
- 總體來說， mmap to mmap 的速度最快

比較本機傳輸與連線傳輸的速度差異

target_file

- 在本機傳輸 target_file：

```
- fcntl to fcntl
Transmission time: 1043.199200 ms, File size: 12022885 bytes

- mmap to fcntl
Transmission time: 1019.906800 ms, File size: 12022885 bytes

- fcntl to mmap
Transmission time: 1931.916900 ms, File size: 12022885 bytes

- mmap to mmap
Transmission time: 1010.749400 ms, File size: 12022885 bytes
```

- 連線傳輸 target_file：

```
- fcntl to fcntl
Transmission time: 16997.158500 ms, File size: 12022885 bytes

- mmap to fcntl
Transmission time: 18057.518200 ms, File size: 12022885 bytes

- fcntl to mmap
Transmission time: 15016.440200 ms, File size: 12022885 bytes

- mmap to mmap
Transmission time: 17970.239500 ms, File size: 12022885 bytes
```

in_40960 ：

- 在本機傳輸 in_40690：

```
- fcntl to fcntl
Transmission time: 1.502600 ms, File size: 40960 bytes

- mmap to fcntl
Transmission time: 1.693900 ms, File size: 40960 bytes

- fcntl to mmap
Transmission time: 2.079700 ms, File size: 40960 bytes

- mmap to mmap
Transmission time: 1.162800 ms, File size: 40960 bytes
```

- 連線傳輸 `in_40960`：

```
- fcntl to fcntl
Transmission time: 25.964300 ms, File size: 40960 bytes

- mmap to fcntl
Transmission time: 21.367500 ms, File size: 40960 bytes

- fcntl to mmap
Transmission time: 6.055700 ms, File size: 40960 bytes

- mmap to mmap
Transmission time: 6.163800 ms, File size: 40960 bytes
```

總體來說，連線傳輸速度較本地傳輸慢；推測是 socket 連線時，連線會比本地花上更多的時間，造成傳輸時間變長

組內分工表及分工比重

姓名	分工	比重
陳品臻	Demo、mmap改寫	14 %
劉倢希	分析數據、Report撰寫	14 %
陳柏鴻	Demo、繪製數據分析圖	14 %
陳昱妤	多檔案傳送、 mmap 改寫	30 %
夏寧	錄製Demo、分析數據	14 %
張漢芝	Report撰寫、測試程式	14 %

Reference

https://github.com/wangyenjen/OS-Project-2 (https://github.com/wangyenjen/OS-Project-2)

https://github.com/andy920262/OS2016/tree/master/project2 (https://github.com/andy920262/OS2016/tree/master/project2)