2020 OS Project 2

資工二 b07502159 黃昱翔
資工二 b07902017 蔡昭信
資工二 b07902037 蔡沛勳
資工二 b07902039 曾暐翔
資工二 b07902049 謝獻沅
資工二 b07902115 陳致元

1 設計

概念：

Master User Program：將file的長度與資料映射進master device，由master device處理傳送
Slave User Program： 將slave device從master device接收到的資料分段映射到files
Master Device：實作了mmap的函式，並且將master user program 的資料從 socket傳出去
Slave Device：從kernel socket接收資料，將資料寫進/dev/slave_device的virtual memory中

Device 實作：

Master device 與 slave device 實作時，mmap定義的方法相同(mmap_exec()，如下)，只在ioctl 中與socket的互動操作上有差異。

Device程式碼實作：

mmap_exec 的實作如下：

```
static int mmap_exec(struct file *filp, struct vm_area_struct *vma)
{
	unsigned long start = vma->vm_start;
	unsigned long pfn = virt_to_phys(filp->private_data) >> PAGE_SHIFT;
	unsigned long size = vma->vm_end - vma->vm_start;
	pgprot_t prot = vma->vm_page_prot;

	if (remap_pfn_range(vma, start, pfn, size, prot))
		return -EIO;

	vma->vm_flags |= VM_RESERVED;
	vma->vm_ops = &mmap_operation;
	vma->vm_private_data = filp->private_data;
	mmap_open(vma);

	return 0;
}
```

Master device ioctl 實作如下：

```
case master_IOCTL_MMAP:
      //send data with kernel socket
      ksend(sockfd_cli, file->private_data, ioctl_param, 0);
      break;
```

Slave device ioctl 實作如下：

```
case slave_IOCTL_MMAP:
      // receive data from kernel socket
      memset(file->private_data, '\0', MAP_LENS);
      while((len = krecv(sockfd_cli, file->private_data+offset,
                                                PAGE_SIZE, 0)) > 0){
            offset += len;
      }
      ret = offset;
      offset = 0;
      break;
```

User program mmap實作方法：
Master.c：由input files中讀取資料並利用mmap將檔案寫入master device中，在此處我定義了一個MAP_SIZE = PAGE_SIZE * 100在記憶體映射時能夠更好的貼合每一個PAGE的大小。

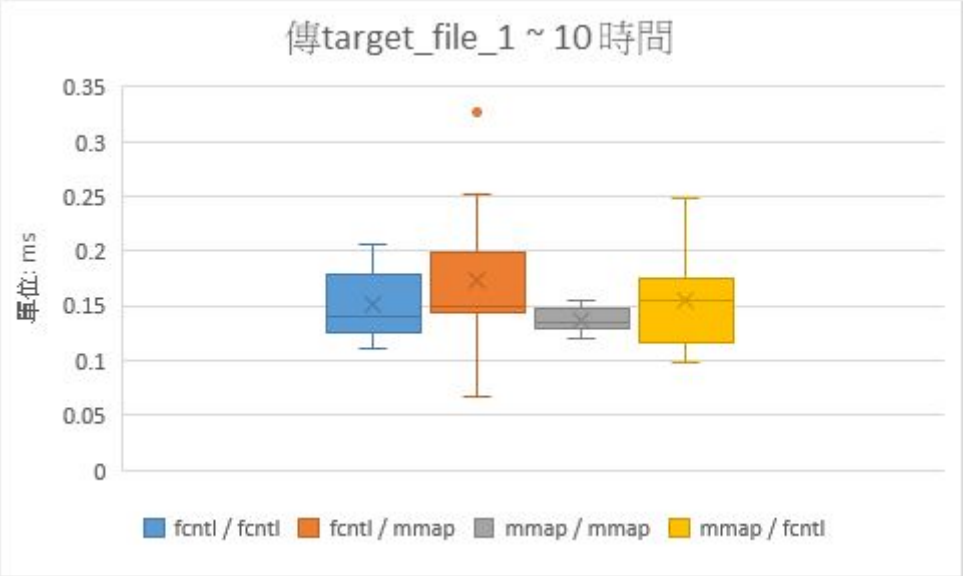
Slave.c：由slave device去接收資料並利用mmap將kernel中的資料寫入檔案，在此處也定義MAP_LENS = PAGE_SIZE * 100，在記憶體映射時一次直接映射最多100個PAGE。

Synchornized 實作方法：
為了確保slave接收的資料與master同步及讓slave知道檔案已結束，我們的做法是在每個檔案傳輸的時候都開一次socket；也就是說每次傳輸檔案時，socket必須重新連接。這麼做的效果是當slave device在該socket已收不到資料時，就代表著一個檔案的結束。

2 比較

測試版本為LINUX 4.14.25，虛擬機作業系統為Ubuntu，空間為2048MB並配有4顆CPU。

FILE	/sample_input_1/target_file_1 (4bytes)				/sample_input_2/target_file (1502860 bytes)			
master / slave	fcntl / fcntl	fcntl / mmap	mmap / mmap	mmap / fcntl	fcntl / fcntl	fcntl / mmap	mmap / mmap	mmap / fcntl
mean	0.05159	0.05924	0.06465	0.06714	7.12763	7.48959	7.17684	6.63603
std	0.01260	0.01573	0.01772	0.03452	0.63929	0.42717	0.53631	0.43940



FILE	./sample_input_1 中所有檔案 (target_file_1 ~ target_file_10) (2932 bytes in total)			
master / slave	fcntl / fcntl	fcntl / mmap	mmap / mmap	mmap / fcntl
mean	0.15199	0.17338	0.13582	0.15565
std	0.032404	0.07064	0.010864	0.043757

觀察在檔案較小時，master與slave直接用fcntl傳送時傳送花費時間最小。
推測是因為mmap一次需要操作的大小是一個page size，因此在檔案較小的情形下，用fcntl的作法，傳輸速度會略快於用mmap。

傳送檔案較大時，master用mmap傳送slave用fcntl接收花費時間為最小，與預計的用mmap傳送時間應最小的結果不符。

傳送多個檔案時，master和slave皆用mmap傳送時所花費時間為最小，推測是mmap直接映射操作I/O會略快於用fcntl操作。

3 分工

b07502159 黃昱翔	master.c 實作、報告
b07902017 蔡昭信	master device 實作、報告
b07902037 蔡沛勳	slave device 實作、報告、比較
b07902039 曾暐翔	slave device 實作、報告
b07902049 謝獻沅	master device 實作、報告
b07902115 陳致元	slave.c 實作、報告、Demo

4 Reference

1. https://github.com/wangyenjen/OS-Project-2
2. https://github.com/andy920262/OS2016/tree/master/project2
3. https://reurl.cc/yZjmlM
4. https://reurl.cc/20k8XE
5. https://reurl.cc/V6vRqy
6. https://reurl.cc/b5jGaM
7. https://reurl.cc/20k8VO
8. https://reurl.cc/QdkWg9
9. https://reurl.cc/z8jrx0