作業二：使用 mmap 進行檔案複製
題目：
請從 github 下載 mmap_cp.c
可以從網站上下載單獨的檔案
https://github.com/shiwulo/system-programming/blob/master/ch04/mmap_cp.c
或者使用 git 將所有的範例都下載
git clone https://github.com/shiwulo/system-programming.git

接下來使用 gdb 對 rdtsc.c 除錯
mmap_cp 使用 mmap 函數實現「cp」，mmap 函數可以將指定檔案的內容映射到記憶體中，如果 mmap 的參數是「MAP_SHARED」，那麼對記憶體修改的內容會被作業系統寫回到檔案系統中。底下這三行程式碼是主要的部分：

```
1.  inputPtr = mmap(NULL, fileSize, PROT_READ, MAP_SHARED , inputFd , 0);
2.  outputPtr = mmap(NULL, fileSize, PROT_WRITE, MAP_SHARED , outputFd , 0);
3.  memcpy(outputPtr, inputPtr, fileSize);
```

在 1. 2. 中將 inputFd 和 outputFd 所代表的檔案「複製到記憶體」，mmap 的回傳值是「該記憶體」的 address
在 3. 中只要進行記憶體複製（即 memcpy），就可以將檔案內容從 inputFd 複製到 outputFd。請注意：如果不是使用 MAP_SHARED，作業系統不會將變更過的記憶體內容更新到檔案中。請參見 man mmap

```
MAP_SHARED
       Share  this  mapping.   Updates  to the mapping are visible to
       other processes mapping the same region, and (in the  case  of
       file-backed  mappings)  are  carried through to the underlying
       file.  (To precisely control when updates are carried  through
       to the underlying file requires the use of msync(2).)
```

要修改的部分：
mmap_cp 並不支援 file hole，請你撰寫一個程式稱之為 mmap_cp2，mmap_cp2 可以跳過 file hole，只複製檔案中有「真正內容」的部分。

附註一：
作業系統會用自己的記憶體管理機制，決定何時將檔案內容複製到記憶體中

附註二：
mmap()在處理大型檔案時特別好用，例如一個很大的文件檔案，使用者可能在這個檔案中任意的瀏覽。

附註三：
mmap()的另一個常用的用法是：二個應用程式將同一個檔案印射到各自的記憶體中，任何一個應用程式對該段記憶體做修改，另一個應用程式可以立即看到這個修改。

附註四：
執行檔在電腦中的執行方式與 mmap 非常的類似。實際上作業系統並未將執行檔立即的複製到記憶體中，而是「假裝已經複製完畢」然後就開始執行，這當然會發生很多「『類似』segmentation fault」的情況，當這種情況發生時，作業系統才將所需要的執行檔的「該片段程式碼」複製到記憶體。因此在執行的過程會發生很多「『類似』segmentation fault」。這種技巧稱之為「demand paging」，可以大幅度的減少記憶體的需求量。

報告：
請自行設計實驗，量測 mmap_cp2 的效能與 mycp2 的速度。提示：使用 Linux 內建的 time 指令。

繳交：
1. 繳交報告，報告的格式並須為 pdf。報告前請附上姓名（可隱匿一個字）及學號
2. 程式碼：請繳交你的程式碼及 makefile，助教執行 make 以後必須產生 mmap_cp2。將程式碼及 makefile 使用 tar 壓縮成.tar.bz2
3. 繳交到 ecourse2 上
4. 再次提醒，助教會將所有人的作業於 dropbox 上公開
5. 繳交期限：2020/03/23 早上 8:00。不能遲交
6. 如果真的不會寫，記得去請教朋友。在你的報告上寫你請教了誰即可。