OS Project 2

Test Environment

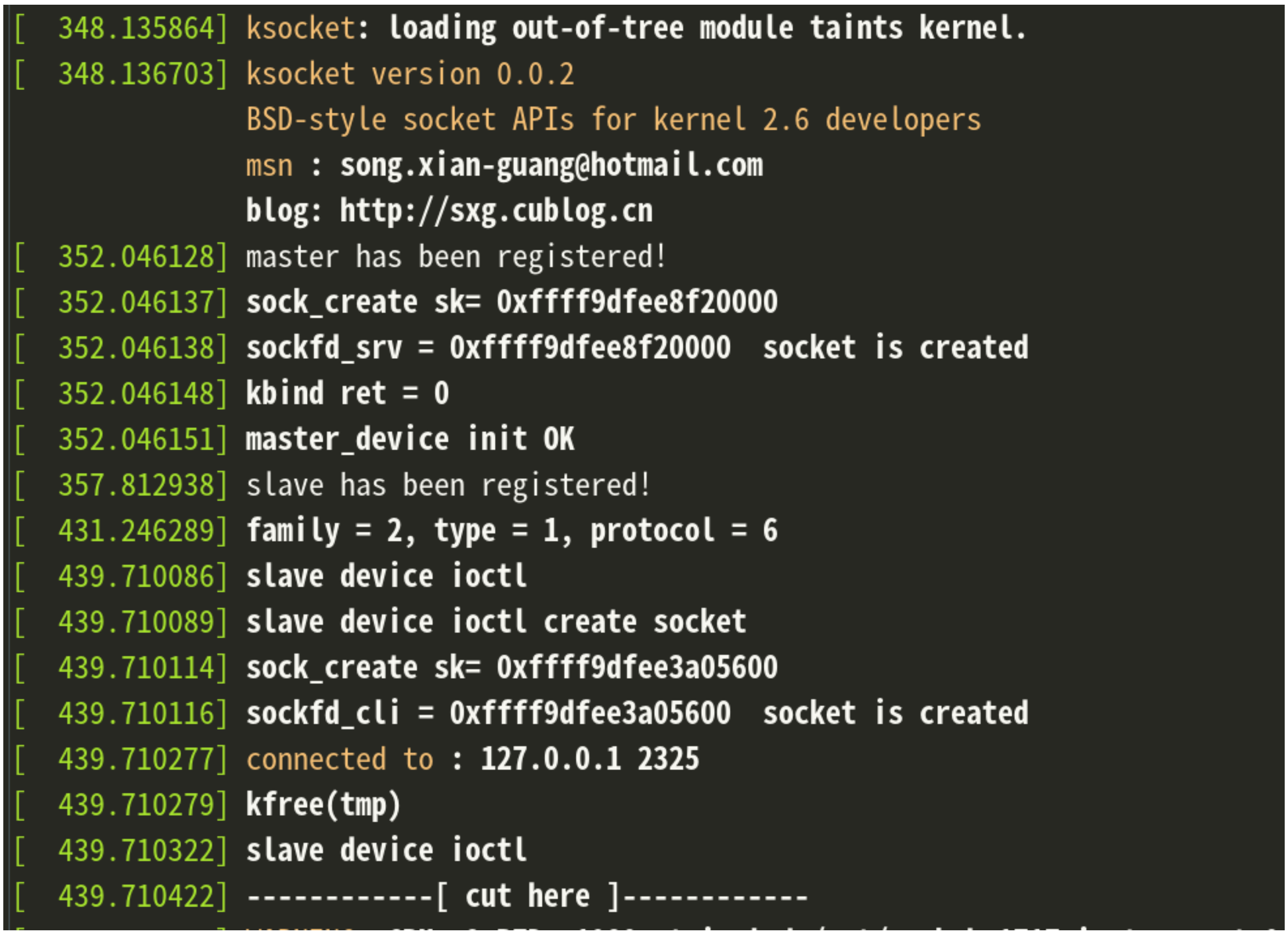
```
$uname -r
4.15.0-106-generic
```

- CPU:i5-3210M @ 2.50GHz
- OS:Linux Mint 19.3
- RAM:8GB
- Storage:180GiB
- Test on a physical machine

DMESG Output

```
#./user_program/master 1 mmap input/sample_input_1/target_file_1
```

```
#./user_program/slave 1 mmap 127.0.0.1 output/target_file_1
```



```
[  348.135864] ksocket: loading out-of-tree module taints kernel.
[  348.136703] ksocket version 0.0.2
               BSD-style socket APIs for kernel 2.6 developers
               msn : song.xian-guang@hotmail.com
               blog: http://sxg.cublog.cn
[  352.046128] master has been registered!
[  352.046137] sock_create sk= 0xffff9dfee8f20000
[  352.046138] sockfd_srv = 0xffff9dfee8f20000  socket is created
[  352.046148] kbind ret = 0
[  352.046151] master_device init OK
[  357.812938] slave has been registered!
[  431.246289] family = 2, type = 1, protocol = 6
[  439.710086] slave device ioctl
[  439.710089] slave device ioctl create socket
[  439.710114] sock_create sk= 0xffff9dfee3a05600
[  439.710116] sockfd_cli = 0xffff9dfee3a05600  socket is created
[  439.710277] connected to : 127.0.0.1 2325
[  439.710279] kfree(tmp)
[  439.710322] slave device ioctl
[  439.710422] ------------[ cut here ]------------
```

Design

Master Device(master_device.c)

Design master_device file_operations and mmap_operations:

```
void project2_open(struct vm_area_struct *vma)
void project2_close(struct vm_area_struct *vma)
static int project2_fault(struct vm_fault *vmf)
static const struct vm_operations_struct project2_vm_ops = {
    .open = project2_open,
    .close = project2_close,
    .fault = project2_fault
};
static int project2_mmap(struct file *file,struct vm_area_struct *vma){
```

- In `project2_open` and `project2_close`, just print debug message or do nothing.
- In `project2_fault`, deal with page fault.
- In `project2_mmap`, use `virt_to_phys` and `remap_pfn_range` to handle memory mapping.
- Design master_ioctl_mmap operations: Use `ksend` to send data to socket.

Slave Device(slave_device.c)

- Design master_device file_operations and mmap_operations:

```
void project2_open(struct vm_area_struct *vma){
}
void project2_close(struct vm_area_struct *vma){
}
static int project2_fault(struct vm_fault *vmf){
}
static const struct vm_operations_struct project2_vm_ops = {
    .open = project2_open,
    .close = project2_close,
    .fault = project2_fault
};
static int project2_mmap(struct file *file,struct vm_area_struct *vma){
}
```

- In `project2_open` and `project2_close`, just print debug message or do nothing.
- In `project2_fault`, deal with page fault.
- In `project2_mmap`, use `virt_to_phys` and `remap_pfn_range` to handle memory mapping.
- Design slave_ioctl_mmap operations:
- Use `krecv` with `MSG_WAITALL` flag to get data,`MSG_WAITALL` should block until all data from socket has been received. Copy data to file. If data_size >= P2_MAP_SIZE: return

Master User Program(master.c)

- Input Parameter: ./master num_of_files method file(s) `Example: ./master 1 mmap test.txt`

Step by Step:

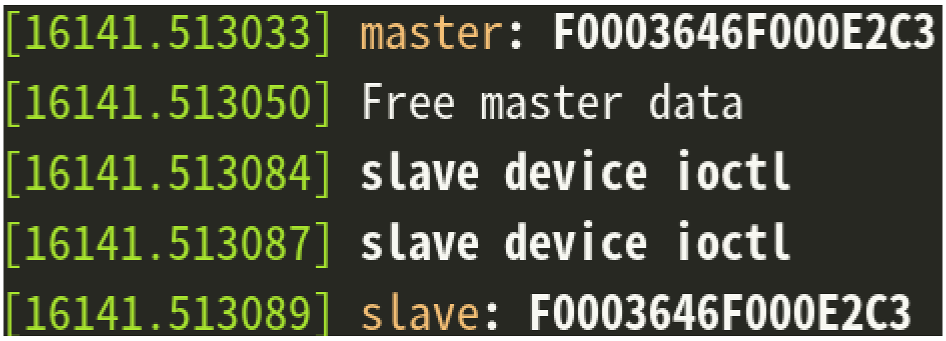
1. 使用一個迴圈來依序傳送輸入的檔案 1.1. 使用Sample Code 的fcntl傳送:直接使用原本的sample code 1.2. 使用mmap接收:一次最多傳送P2_MAP_SIZE大小的檔案內容至master_device，並重複此步驟至該檔案傳送完畢
2. 迴圈結束後印出其執行時間及檔案大小

Slave User Program(slave.c)

- Input Parameter: ./slave num_of_files method IP file(s) `Example: ./slave 3 fcntl 127.0.0.1 test.txt test2.txt test3.txt`

1. 使用一個迴圈來依序接收檔案 1.1. 使用Sample Code 的fcntl接收:使用原本的sample code 1.2. 使用mmap接收:持續從slave_device讀取資料，並重複此步驟至所有資料讀取完畢(ret=0)
2. 迴圈結束後印出其執行時間及檔案大小

Page Descriptor



```
[16141.513033] master: F0003646F000E2C3
[16141.513050] Free master data
[16141.513084] slave device ioctl
[16141.513087] slave device ioctl
[16141.513089] slave: F0003646F000E2C3
```

Difference between file-I/O and memory-mapped I/O

為了方便比較，統一使用slave的output(因為master的有手動操作延遲的問題)

Sample_Output

File I/O(Slave)

- Sending 1 file per exeution(target_file 1~10 and target_file)

```
Transmission time: 0.165000 ms, File size: 4 bytes
Transmission time: 0.085700 ms, File size: 107 bytes
Transmission time: 0.121700 ms, File size: 167 bytes
Transmission time: 0.066900 ms, File size: 207 bytes
Transmission time: 0.091500 ms, File size: 255 bytes
Transmission time: 0.084000 ms, File size: 383 bytes
Transmission time: 0.055100 ms, File size: 351 bytes
Transmission time: 0.056600 ms, File size: 507 bytes
Transmission time: 0.044300 ms, File size: 457 bytes
Transmission time: 0.058700 ms, File size: 577 bytes
Transmission time: 8.604500 ms, File size: 1502860 bytes
```

- Sending 5 file per execution
- Output of `./5_slave.sh fcntl`

```
Transmission time: 0.306000 ms, File size: 2275 bytes
Transmission time: 0.369900 ms, File size: 2275 bytes
Transmission time: 0.240400 ms, File size: 2275 bytes
Transmission time: 0.239800 ms, File size: 2275 bytes
Transmission time: 0.327300 ms, File size: 2275 bytes
Transmission time: 0.425900 ms, File size: 2275 bytes
Transmission time: 0.320200 ms, File size: 2275 bytes
Transmission time: 0.375800 ms, File size: 2275 bytes
Transmission time: 0.270000 ms, File size: 2275 bytes
Transmission time: 0.352500 ms, File size: 2275 bytes
```

Average time: 0.320008 ms

- Sending 10 file per execution

- Output of `./10_slave.sh fcntl`

```
Transmission time: 0.505900 ms, File size: 3018 bytes
Transmission time: 0.528700 ms, File size: 3018 bytes
Transmission time: 0.562100 ms, File size: 3018 bytes
Transmission time: 0.556900 ms, File size: 3018 bytes
Transmission time: 0.572800 ms, File size: 3018 bytes
Transmission time: 0.527900 ms, File size: 3018 bytes
Transmission time: 0.486800 ms, File size: 3018 bytes
Transmission time: 0.493300 ms, File size: 3018 bytes
Transmission time: 0.488000 ms, File size: 3018 bytes
Transmission time: 0.388300 ms, File size: 3018 bytes
```

Average Time: 0.51107 ms

Memory-mapped I/O(Slave)

- Sending 1 file per exeution(target_file 1~10 and target_file)

```
Transmission time: 0.501900 ms, File size: 4 bytes
Transmission time: 0.078600 ms, File size: 107 bytes
Transmission time: 0.096500 ms, File size: 167 bytes
Transmission time: 0.196100 ms, File size: 207 bytes
Transmission time: 0.069900 ms, File size: 255 bytes
Transmission time: 0.080000 ms, File size: 383 bytes
Transmission time: 0.057000 ms, File size: 351 bytes
Transmission time: 0.062800 ms, File size: 507 bytes
Transmission time: 0.048000 ms, File size: 457 bytes
Transmission time: 0.169500 ms, File size: 577 bytes
Transmission time: 2.693700 ms, File size: 1502860 bytes
```

- Sending 5 file per execution
- Output of `./5_slave.sh mmap`

```
Transmission time: 0.289900 ms, File size: 2275 bytes
Transmission time: 0.244300 ms, File size: 2275 bytes
Transmission time: 0.306400 ms, File size: 2275 bytes
Transmission time: 0.219100 ms, File size: 2275 bytes
Transmission time: 0.232500 ms, File size: 2275 bytes
Transmission time: 0.273700 ms, File size: 2275 bytes
Transmission time: 0.308200 ms, File size: 2275 bytes
Transmission time: 0.522400 ms, File size: 2275 bytes
Transmission time: 0.346200 ms, File size: 2275 bytes
Transmission time: 0.335200 ms, File size: 2275 bytes
Transmission time: 0.183800 ms, File size: 2275 bytes
Transmission time: 0.366400 ms, File size: 2275 bytes
Transmission time: 0.248100 ms, File size: 2275 bytes
```

Average time:0.298169231 ms

- Sending 10 file per execution
- Output of `./10_slave.sh mmap`

```
Transmission time: 0.358500 ms, File size: 3018 bytes
Transmission time: 0.307200 ms, File size: 3018 bytes
Transmission time: 0.395400 ms, File size: 3018 bytes
Transmission time: 0.484600 ms, File size: 3018 bytes
Transmission time: 0.332700 ms, File size: 3018 bytes
Transmission time: 0.431600 ms, File size: 3018 bytes
Transmission time: 0.379500 ms, File size: 3018 bytes
Transmission time: 0.500100 ms, File size: 3018 bytes
Transmission time: 0.521000 ms, File size: 3018 bytes
Transmission time: 0.436200 ms, File size: 3018 bytes
```

Average Time: 0.41468 ms

Self-designed-Input (Large files)

Use dd to generate two sets of image files:

1. `dd bs=1M count=100 if=/dev/zero of=./input/100M.img`
2. `dd bs=1 count=100M if=/dev/random of=./input/R100M.img` We have 5 different size:1K,5K,1M,10M,100M. Use `md5sum` or `sha512sum` to check the integrity of file. Example:`sudo sha512sum output/R10M.img input/R10M.img` Note:for *.img files,use `git lfs` instead.(More info :https://git-lfs.github.com/)

```
89a3d69233efd946c57b7a62cc300d50254cf1d7e5cb380960a328a197f462fec9e4c4ef728
4ed74ea61333606ae09ec26c5322a4d1fc05f06cabcae41541d54  output/R10M.img
89a3d69233efd946c57b7a62cc300d50254cf1d7e5cb380960a328a197f462fec9e4c4ef728
4ed74ea61333606ae09ec26c5322a4d1fc05f06cabcae41541d54  input/R10M.img
```

Use `./zero_(slave/master).sh $method and ./random_(slave/master).sh $method` to test.

File I/O

Master

```
$cat output/zero_fcntl_master_result.txt
```

```
Transmission time: 6993.987400 ms, File size: 14549760 bytes
Transmission time: 1073.988100 ms, File size: 14549760 bytes
Transmission time: 2912.243900 ms, File size: 14549760 bytes
Transmission time: 1976.975900 ms, File size: 14549760 bytes
Transmission time: 1074.965800 ms, File size: 14549760 bytes
Transmission time: 2012.050300 ms, File size: 14549760 bytes
Transmission time: 1079.192600 ms, File size: 14549760 bytes
Transmission time: 1987.399100 ms, File size: 14549760 bytes
Transmission time: 1994.656800 ms, File size: 14549760 bytes
Transmission time: 1981.934100 ms, File size: 14549760 bytes
```

Note:忽略第一個output(手動控制所造成的延遲) Average Time: 1788.156288889 ms

```
$cat output/random_fcntl_master_result.txt
```

```
Transmission time: 1957.694600 ms, File size: 14549760 bytes
Transmission time: 2010.371100 ms, File size: 14549760 bytes
Transmission time: 2031.600700 ms, File size: 14549760 bytes
Transmission time: 1984.331500 ms, File size: 14549760 bytes
Transmission time: 1086.119700 ms, File size: 14549760 bytes
Transmission time: 2008.644700 ms, File size: 14549760 bytes
Transmission time: 1978.362400 ms, File size: 14549760 bytes
Transmission time: 2924.766200 ms, File size: 14549760 bytes
Transmission time: 2012.708300 ms, File size: 14549760 bytes
Transmission time: 2960.611000 ms, File size: 14549760 bytes
```

Average Time: 2095.52102 ms

Slave

```
$cat output/zero_fcntl_slave_result.txt
```

```
Transmission time: 66.640300 ms, File size: 14549760 bytes
Transmission time: 72.154200 ms, File size: 14549760 bytes
Transmission time: 970.938200 ms, File size: 14549760 bytes
Transmission time: 65.921200 ms, File size: 14549760 bytes
Transmission time: 65.828800 ms, File size: 14549760 bytes
Transmission time: 974.680900 ms, File size: 14549760 bytes
Transmission time: 66.441900 ms, File size: 14549760 bytes
Transmission time: 970.419200 ms, File size: 14549760 bytes
Transmission time: 974.920800 ms, File size: 14549760 bytes
Transmission time: 967.156000 ms, File size: 14549760 bytes
```

Average Time: 519.51015 ms

```
$cat output/random_fcntl_slave_result.txt
```

```
Transmission time: 970.686900 ms, File size: 14549760 bytes
Transmission time: 972.074100 ms, File size: 14549760 bytes
Transmission time: 969.775500 ms, File size: 14549760 bytes
Transmission time: 966.934100 ms, File size: 14549760 bytes
Transmission time: 970.128600 ms, File size: 14549760 bytes
Transmission time: 967.237200 ms, File size: 14549760 bytes
Transmission time: 970.282600 ms, File size: 14549760 bytes
Transmission time: 971.151200 ms, File size: 14549760 bytes
Transmission time: 970.238600 ms, File size: 14549760 bytes
Transmission time: 965.929300 ms, File size: 14549760 bytes
```

Average Time: 969.44381 ms

Memory-mapped I/O

Master

```
$cat output/zero_mmap_master_result.txt
```

```
Transmission time: 1939.596700 ms, File size: 14549760 bytes
Transmission time: 1059.252700 ms, File size: 14549760 bytes
Transmission time: 1043.389000 ms, File size: 14549760 bytes
Transmission time: 1048.843200 ms, File size: 14549760 bytes
Transmission time: 1041.104900 ms, File size: 14549760 bytes
Transmission time: 1076.614900 ms, File size: 14549760 bytes
Transmission time: 1014.962200 ms, File size: 14549760 bytes
Transmission time: 1044.944900 ms, File size: 14549760 bytes
Transmission time: 1044.045600 ms, File size: 14549760 bytes
Transmission time: 1028.866900 ms, File size: 14549760 bytes
```

Average Time: 1134.1621 ms

```
$cat output/random_mmap_master_result.txt
```

```
Transmission time: 8944.142500 ms, File size: 14549760 bytes
Transmission time: 1991.522000 ms, File size: 14549760 bytes
Transmission time: 1064.617100 ms, File size: 14549760 bytes
Transmission time: 1998.334800 ms, File size: 14549760 bytes
Transmission time: 1043.246200 ms, File size: 14549760 bytes
Transmission time: 1955.157000 ms, File size: 14549760 bytes
Transmission time: 1045.301100 ms, File size: 14549760 bytes
Transmission time: 1950.522900 ms, File size: 14549760 bytes
Transmission time: 1052.616800 ms, File size: 14549760 bytes
Transmission time: 1054.362700 ms, File size: 14549760 bytes
```

Note:忽略第一個output(手動控制所造成的延遲) Average Time: 1461.742288889 ms

Slave

```
$cat output/zero_mmap_slave_result.txt
```

```
Transmission time: 920.174600 ms, File size: 14549760 bytes
Transmission time: 919.943400 ms, File size: 14549760 bytes
Transmission time: 20.325300 ms, File size: 14549760 bytes
Transmission time: 20.366700 ms, File size: 14549760 bytes
Transmission time: 19.711800 ms, File size: 14549760 bytes
Transmission time: 59.114600 ms, File size: 14549760 bytes
Transmission time: 19.609100 ms, File size: 14549760 bytes
Transmission time: 19.834100 ms, File size: 14549760 bytes
Transmission time: 28.700200 ms, File size: 14549760 bytes
Transmission time: 20.873400 ms, File size: 14549760 bytes
```

Average Time: 204.86532 ms

```
$cat output/random_mmap_slave_result.txt
```

```
Transmission time: 935.257900 ms, File size: 14549760 bytes
Transmission time: 938.134300 ms, File size: 14549760 bytes
Transmission time: 38.290600 ms, File size: 14549760 bytes
Transmission time: 938.044400 ms, File size: 14549760 bytes
Transmission time: 20.668500 ms, File size: 14549760 bytes
Transmission time: 937.098300 ms, File size: 14549760 bytes
Transmission time: 20.424500 ms, File size: 14549760 bytes
Transmission time: 938.356700 ms, File size: 14549760 bytes
Transmission time: 41.646600 ms, File size: 14549760 bytes
Transmission time: 36.699400 ms, File size: 14549760 bytes
```

Average Time: 484.46212 ms

比較

我們可以發現，當檔案大小較小時，memory-mapped I/O雖比file快速，但是差別並不明顯。即使一次傳送5/10檔案的差距仍然不顯著。但從自行設計的input中可以發現，當檔案較大時(100MB左右)，memory-mapped的效率就比file好，很可能是因為file需要多次呼叫system call並回傳user space，造成時間上的浪費。加上mmap為對Virtual Memory進行操作，效率更佳。另外，從我們產生的兩組數據可以發現，傳送全為0的image比起random產生的image更快速，可能和CPU處理檔案時造成的差距。

組內分工表

- Code Design: 鄭昊昕 (b07902125@ntu.edu.tw)
- Test Data Generation: 何政勳 (b07902129@ntu.edu.tw)
- Report: 鄭昊昕 (b07902125@ntu.edu.tw),何政勳 (b07902129@ntu.edu.tw)

Reference:

1. http://www.jollen.org/blog/2007/01/linux_virtual_memory_areas_vma.html
2. https://blog.csdn.net/eZiMu/article/details/54910019
3. https://stackoverflow.com/questions/5748492/is-there-any-api-for-determining-the-physical-address-from-virtual-address-in-li
4. https://www.kernel.org/doc/gorman/html/understand/understand006.html
5. http://www.jollen.org/blog/2007/04/mmap_remap_page_range_nutshell.html
6. https://github.com/paraka/mmap-kernel-transfer-data/
7. https://stackoverflow.com/questions/8470403/socket-recv-hang-on-large-message-with-msg-waitall
8. https://www.kernel.org/doc/gorman/html/understand/understand006.html
9. https://sysplay.github.io/books/LinuxDrivers/book/Content/Part05.html
10. http://www.jollen.org/blog/2007/01/linux_virtual_memory_areas_vma.html
11. https://stackoverflow.com/questions/29695247/what-is-the-relation-between-virt-to-phys-and-the-cpus-mmu-in-the-linux-kernel
12. https://www.kernel.org/doc/html/v4.15/kernel-hacking/hacking.html?highlight=file_operation
13. https://blog.csdn.net/eZiMu/article/details/54910019
14. https://stackoverflow.com/questions/5748492/is-there-any-api-for-determining-the-physical-address-from-virtual-address-in-li
15. http://www.jollen.org/blog/2007/04/mmap_remap_page_range_nutshell.html
16. https://github.com/paraka/mmap-kernel-transfer-data/blob/master/mmap-example.c
17. https://labs.f-secure.com/assets/BlogFiles/mwri-mmap-exploitation-whitepaper-2017-09-18.pdf
18. https://elixir.bootlin.com/linux/latest/source/include/linux/mm.h
19. https://hant-kb.kutu66.com/c/post_2768082
20. https://man7.org/linux/man-pages/man2/mmap.2.html
21. https://stackoverflow.com/questions/14063046/fallocate-vs-posix-fallocate
22. https://www.man7.org/linux/man-pages/man2/fallocate.2.html
23. https://stackoverflow.com/questions/12210451/how-to-prevent-memcpy-buffer-overflow
24. https://unix.stackexchange.com/questions/128213/how-is-page-size-determined-in-virtual-address-space
25. https://cboard.cprogramming.com/c-programming/143410-segment-fault-shared-memory-set-using-mmap.html
26. https://stackoverflow.com/questions/4779188/how-to-use-mmap-to-allocate-a-memory-in-heap
27. https://lwn.net/Articles/129480/

28. https://github.com/MarkTseng/mySampleCode/blob/master/linux_device_driver_template/linux_module_example/mmapdriver.c
29. https://hackmd.io/@DIuvbu1vRU2C5FwWIMzZ_w/H1cuTQrCH