OS2020 SPRING Final Project

Member

- B06902016 林紹維
- B06902020 唐浩
- B06902024 黃秉迦
- B06902057 薛佳哲
- B06902074 柯宏穎
- B06902106 宋岩叡

OS and Kernel Version

We run on different OS and environment.

- Arch Linux, 4.14.184
- Ubuntu16.04, 4.14.25

Notice: Our video files is too large to upload to github. If your want to use it, please download it from workstation, or just run the download script in the repo(`src/download.sh`). It will be placed in `./video`. `output/video` would be empty, too. Please excute and generate by yourself.

Design

We design two methods to implement. The difference between them is whether reconnect or not. If there are n files asked to trasmit:

1. They would disconnect after accomplishing a file and reconnect before transmitting the next file.
2. The master would let the slave know the file size before every transmitting. And the slave would read proper size instead of just reading a buffer size.

We referenced lots of groups' implementation, all of them used $\mathrm{Method\ 1}$. But we dont't want to do connection so many times. $\mathrm{Method\ 2}$ is more complicate then $\mathrm{Method\ 1}$. Hence, we implement $\mathrm{Method\ 1}$ first(in $master$ branch) and modify into $\mathrm{Method\ 2}$(in $interaction$ branch).

$fcntl$ is already accomplished in the sample code, so we just did some optimizations and fixed some bugs. However, $mmap$ is much more complicated. We add some $ioctl$ cases and define some mapping functions in our code. We create two spaces and copy data. And we found that while using $mmap$, its offset can only be the multiple of 4096, which means we can only write times as many spaces as

$\mathrm{PAGE\_SIZE}$. Once it violates $\mathrm{PAGE\_SIZE}$, it would cause segmentation fault during next mapping. Therefore this can only happen in the last transmission of each file. To avoid this situation, there is a flag, $\mathrm{MSG\_WAITALL}$, can be set. After setting we can promise that it would always be blocked until it received a $\mathrm{BUF\_SIZE}$ in general. Only when it is the last batch of data, it cannot return until it reads EOF. Since the connection would terminate, the slave can easily know it is the end and stop.

In $\mathrm{Method\ 2}$, master can communicate with slave. We just add `send_msg` and `ksend` into slave_device and add `recv_msg` and `krecv` into master_device. Originally, we want to implement sychronous between them, which means when the master send a file and finish, it can transmit the next file until receiving slave's respond. Thus it is neccessary to implement communication. However, we find it is not neccessary and would reduce the performance. Thus, it is eliminated. The master can still transmit the file size and data continuousily. In order to transmit the file size to slave, we choose $read/write$ to implement. We also try to use $mmap$ to achieve, but there is a problem mentioned above. $mmap$ can only transmit the certain size of data. If we want to send a $8\ Byte$ file size, we need to expand it into a $\mathrm{PAGE\_SIZE}$'s charactor, and the slave needs to convert it back to the number. It is inconvenience, so we just using $read/write$ to implement it. While implementing $\mathrm{Method\ 2}$, we find a few bugs in the sample code:

1. In slave_device's `recv_msg`, it always reads a $\mathrm{BUF\_SIZE}$ of data instead of the size we pass in. We pass the argument by `ioctl`, but the implement didn't reference the argument but just read `sizeof(msg)`. This is horrible because the slave need to read the data precisely, otherwise it will read the size of next file or even more data. It will make the whole system crash.
2. The time computing method is wrong. While convert the microsecond into millisecond, it need to times 0.001 instead of 0.0001.

Experiment

We test on sample input file, and two large video. Video is the common file in our life. There are tens of thousands video files tranmitting on the internet. Hence, we use these kind of file to test. Moreover, video is a good test data since we can check its integrity by checking whether it can play or not, and it is larger enough to test our program.

We conducted each experiment 100 times for stability issue. And we define $\mathrm{PAGE\_SIZE} = 4096, \mathrm{BUF\_SIZE} = 512$ in our code.

Method 1 with MMAP_SIZE = 16×PAGE_SIZE, Synchonous

- sample input 1(10 files, $23.549\ \mathrm{KB}$)

master type	slave type	average time (ms)	std time (ms)	trans. rate (KB/ms)
fcntl	fcntl	5.637	1.415	4.183
fcntl	mmap	3.899	1.13	6.048
mmap	fcntl	4.895	1.239	4.817
mmap	mmap	4.005	0.558	5.888

- sample input 2(1 files, $11.466\ \mathrm{MB}$)

master type	slave type	average time (ms)	std time (ms)	trans. rate (KB/ms)
fcntl	fcntl	176.136	108.727	66.659
fcntl	mmap	40.123	2.464	292.627
mmap	fcntl	163.104	58.762	71.985
mmap	mmap	27.206	2.507	431.558

- video(2 files, $347.466\ \mathrm{MB}$)

master type	slave type	average time (ms)	std time (ms)	trans. rate (KB/ms)
fcntl	fcntl	4033.317	456.828	88.217
fcntl	mmap	1261.831	259.081	281.976
mmap	fcntl	4170.859	566.597	85.307
mmap	mmap	793.33	319.4	448.496

Method 1 with MMAP_SIZE = 256$\times$PAGE_SIZE, Synchonous

- sample input 1(10 files, $23.549\ \mathrm{KB}$)

master type	slave type	average time (ms)	std time (ms)	trans. rate (KB/ms)
fcntl	fcntl	5.181	1.177	4.545
fcntl	mmap	3.725	0.964	6.322
mmap	fcntl	5.101	1.145	4.617
mmap	mmap	3.872	1.372	6.082

- sample input 2(1 files, $11.466\ \mathrm{MB}$)

master type	slave type	average time (ms)	std time (ms)	trans. rate (KB/ms)
fcntl	fcntl	169.378	66.153	69.319
fcntl	mmap	40.993	3.381	286.417
mmap	fcntl	172.246	68.422	68.165
mmap	mmap	20.635	1.886	569.001

- video(2 files, $347.466\ \mathrm{MB}$)

master type	slave type	average time (ms)	std time (ms)	trans. rate (KB/ms)
fcntl	fcntl	4299.917	557.462	82.747
fcntl	mmap	1202.439	153.996	295.903
mmap	fcntl	4237.969	458.522	83.957
mmap	mmap	626.746	134.751	567.703

Method 1 with MMAP_SIZE = 256$\times$PAGE_SIZE, Asynchonous

- sample input 1(10 files, $23.549\ \mathrm{KB}$)

master type	slave type	average time (ms)	std time (ms)	trans. rate (KB/ms)
fcntl	fcntl	5.041	1.352	4.671
fcntl	mmap	3.953	1.179	5.957
mmap	fcntl	5.399	1.670	4.361
mmap	mmap	4.043	0.707	5.824

- sample input 2(1 files, $11.466\ \mathrm{MB}$)

master type	slave type	average time (ms)	std time (ms)	trans. rate (KB/ms)
fcntl	fcntl	175.629	59.490	66.852
fcntl	mmap	41.580	3.973	282.375
mmap	fcntl	168.728	58.901	69.586
mmap	mmap	21.303	1.756	551.148

- video(2 files, $347.466\ \mathrm{MB}$)

master type	slave type	average time (ms)	std time (ms)	trans. rate (KB/ms)
fcntl	fcntl	4420.579	469.268	80.488
fcntl	mmap	1198.245	245.860	296.939
mmap	fcntl	4141.126	399.668	85.920
mmap	mmap	648.876	263.477	548.341

Method 2 with MMAP_SIZE = 256$\times$PAGE_SIZE, Synchonous

- sample input 1(10 files, $23.549\ \mathrm{KB}$)

master type	slave type	average time (ms)	std time (ms)	trans. rate (KB/ms)
fcntl	fcntl	1.666	0.777	14.155
fcntl	mmap	0.828	0.589	28.492
mmap	fcntl	1.604	0.687	14.702
mmap	mmap	0.731	0.278	32.244

- sample input 2(1 files, $11.466\ \mathrm{MB}$)

master type	slave type	average time (ms)	std time (ms)	trans. rate (KB/ms)
fcntl	fcntl	147.499	65.04	79.601
fcntl	mmap	41.716	4.272	281.455
mmap	fcntl	168.529	64.934	69.668
mmap	mmap	22.906	6.274	512.576

- video(2 files, $347.466\ \mathrm{MB}$)

master type	slave type	average time (ms)	std time (ms)	trans. rate (KB/ms)
fcntl	fcntl	4091.654	404.769	86.959
fcntl	mmap	1232.795	203.403	288.617
mmap	fcntl	4105.299	377.01	86.67
mmap	mmap	621.746	112.91	572.268

Method 1 v.s. Method 2

We can notice that the performance of $\mathrm{Method}\ 2$ is better than $\mathrm{Method}\ 1$. By comparing the transmission rate, we notice it is much faster on sample input 1, and a little improvement on larger files. The reason is that there are ten files in sample input 1. In $\mathrm{Method}\ 1$, it needs to connect 10 times. This is an overhead that makes the performance of $\mathrm{Method}\ 1$ worse. Since files in sample input 1 is smaller, these overheads impact the transmission rate more significantly. On the contrary, there are only few amount files and video in sample input 2. It would not do so many connections. And the transmission time is much larger than overhead. So there is no significant change in $\mathrm{Method}\ 2$.

Compare different page size

In sample input 1, we observe that changing the size of the *page size* does not make a large difference on transmission time and rate, that's because files in sample input 1 are all smaller than the size of a page$(16 \times 4K = 64K)$. As a result, it would not effect the efficiency. In sample input 2 and video files, nevertheless, we observe that when both master and slave use *mmap* as their method, transimission rate would drop tremendously. We think that this is because when files are much larger than the size of a page$(16 \times 4K)$, decreasing the size of a page would lead to more iteration of memory operations, which causes more time to operate.

Compare file I/O & memory-mapped I/O

From the data in our experiment above, we saw that when files we transmitted are quite small, the effectiveness of $mmap$ is less then the effectiveness of $fcntl$ **on the master side**. The reason is that the overhead of memory-mapped is large (eg. constructing page table...), and situations like page fault will increase the latency very much. Constrastly, when files are large, $mmap$ is more efficient than $fcntl$, that's because $mmap$ can effectively lower the rate of disk I/O, and the latency of this is in microseconds, which is why $mmap$ has its effeciency much better than $fcntl$. Our experiment results are based on slave's finish time since we thought that a task is accomplished only when the data are transfered completely. Hence, the bottleneck is which mode the slave should use. As discussions above, we know that the performance of $mmap$ is way better than $fcntl$. We all know that writing is slower than reading in general, so we can assume that the master can always send out the data faster. When the master using $mmap$ and the slave using $fcntl$, the master can send out the file very fast, but the receiving and writing as slave is much slower. So the transmission time is longer. In constrast, when the master using $fcntl$ and the slave using $mmap$, the slave can get the file faster than before. So the

performance improves significantly. However, $fcntl$ can still restrict the master's transimission rate, this mode is still much slower than both of them using $mmap$.

Bonus

To implement $\mathrm{Asynchronous\ I/O}$, we can just modify the socket's flag, which is defined in `ksocket.c`. After initializing the socket struct, add FASYNC into it(`sk -> flags |= FASYNC`). And we use `ifdef` to control which mode we want to use.

We thinks that the performance of Async would be better than Sync. Since $\mathrm{Async.\ socket}$ don't need to wait the buffer filled up. But our $\mathrm{BUF\_SIZE}$ is really small. There is no significant difference between them.

Page Descriptor(Base on sample input 1)

- master: $fcntl$, slave: $mmap$

```
[kehongying@Laxingyang OS2020_project2]$ sudo dmesg | grep descriptor
[ 2034.976869] slave : 800000020a4ab067 (descriptor)
[ 2034.976955] slave : 80000002086c8067 (descriptor)
[ 2034.977073] slave : 8000000209cdc067 (descriptor)
[ 2034.977182] slave : 8000000209097067 (descriptor)
[ 2034.977230] slave : 8000000205f11067 (descriptor)
[ 2034.977279] slave : 8000000203db4067 (descriptor)
[ 2034.977335] slave : 800000020a735067 (descriptor)
[ 2034.977386] slave : 800000020125c067 (descriptor)
[ 2034.977448] slave : 800000020b8aa067 (descriptor)
[ 2034.977507] slave : 8000000209935067 (descriptor)
```

- master: $mmap$, slave: $fcntl$

```
[kehongying@Laxingyang OS2020_project2]$ sudo dmesg | grep descriptor
[ 1987.874066] master: 800000020c3d5025 (descriptor)
[ 1987.874127] master: 800000020ab9f025 (descriptor)
[ 1987.874168] master: 800000020bfb2025 (descriptor)
[ 1987.874205] master: 800000020978b025 (descriptor)
[ 1987.874241] master: 800000020cae8025 (descriptor)
[ 1987.874278] master: 8000000208646025 (descriptor)
[ 1987.874311] master: 800000020fef4025 (descriptor)
[ 1987.874351] master: 800000020e319025 (descriptor)
[ 1987.874380] master: 800000020b19c025 (descriptor)
[ 1987.874405] master: 800000020b881025 (descriptor)
```

- master: $mmap$, slave: $mmap$

```
[kehongying@Laxingyang OS2020_project2]$ sudo dmesg | grep descriptor
[ 1911.428561] master: 800000020c3d5025 (descriptor)
[ 1911.428629] master: 800000020ab9f025 (descriptor)
[ 1911.428688] master: 800000020bfb2025 (descriptor)
[ 1911.428744] master: 800000020978b025 (descriptor)
[ 1911.428800] master: 800000020cae8025 (descriptor)
[ 1911.428856] master: 8000000208646025 (descriptor)
[ 1911.428913] master: 800000020fef4025 (descriptor)
[ 1911.429019] master: 800000020e319025 (descriptor)
[ 1911.429070] master: 800000020b19c025 (descriptor)
[ 1911.429108] master: 800000020b881025 (descriptor)
[ 1911.429439] slave : 80000002050d6067 (descriptor)
[ 1911.429542] slave : 8000000206577067 (descriptor)
[ 1911.429641] slave : 800000020286c067 (descriptor)
[ 1911.429734] slave : 8000000204117067 (descriptor)
[ 1911.429816] slave : 8000000203543067 (descriptor)
[ 1911.429905] slave : 8000000209f28067 (descriptor)
[ 1911.430057] slave : 8000000203c64067 (descriptor)
[ 1911.430149] slave : 80000002064f1067 (descriptor)
[ 1911.430236] slave : 80000002094a4067 (descriptor)
[ 1911.430346] slave : 8000000203dcf067 (descriptor)
```

Conclusion

- Our experiment runs in a virtual machine with four cores and 4G RAM.

In our project, we have four variables in the experiment: *type of I/O*, *page size*, *transmission method*, and *synchronization or not*. To compare the differences, we designed three groups of test data: multiple small files, one larger file, and one video file.

By observing the comparisons above, we can learn that $\mathrm{Method}\ 2$ would be better than $\mathrm{Method}\ 1$ in most cases, which matches our assumption. Also, properly enlarging the page size could effectively decrease the transmission time.

However, the advantage variables might change depends on different conditions. For example, the type of I/O. On input with smaller files, we found that when we fix the I/O method used by slave, especialy when slave is using $fcntl$, although in most cases it would be slower.

Besides, due to our small buffer size, there is no significant difference between synchronization or not. (In fact, many results of synchronization is better than asynchronization). Accordingly, we have to choose the suitable variables depends on different situations.

Contribution

- B06902016 林紹維
  - Experiment on Method 1(Different $\mathrm{MMAP\_SIZE}$)
  - Report(Experiment, Comparison)
- B06902020 唐浩
  - Report(Conclusion, fix typo and grammar)
- B06902024 黃秉迦
  - Implement Method 1's master
  - Fix sample code's bugs
  - Experiment on Method 1
  - Design statical program
  - Report(Design, Experiment)
  - Bonus
- B06902057 薛佳哲
  - Report(Comparison)
- B06902074 柯宏穎
  - Implement Method 2 & Comparison
  - Implement Method 1's slave
  - Fix sample code's bugs
  - Experiment on Method 2
  - Report(Design, Experiment, Bonus, Comparison)
  - Bonus
- B06902106 宋岩叡
  - Report(Conclusion, fix typo and grammar)

Reference

- https://github.com/domen111/OS-Porject-2
- https://github.com/slSeanWU/OS19S_Proj_2
- https://github.com/wangyenjen/OS-Project-2
- https://stackoverflow.com/questions/41090469/linux-kernel-how-to-get-physical-address-memory-management
- Discuss with Regular Group 39
- Discuss with Regular Group 43