108-2 OS PROJECT 2 REPORT

B06902087 陳品學
B05902082 方銘浩
B06902126 柳相宇

GROUP 54

1. *Environment*

Linux(kernel version 4.15.106)

2. *Input Parameter*

./master [file num] [file directory] [method]

./slave [output num] [input directory] [method] 127.0.0.1

Instead of sequentially enter all the paths of files,we decide to make it read/write to a set of files under the same directory automatically.

3. *Design of our project*

First of all, the basic concept of this project 2 is using mmap() system call to implement I/O and compare with normal file I/O. Our project consist of Master and Slave sending the file descriptor into mmap and output the file size as mmap length. In the concept of Master / Slave the direction of control will always go to master to slave. In a Master / Slave system, a supervisory processor handles the OS. This Os treats all other available processors such as memory and I/O devices. The master allocates to each process and various resources that is necessary for its execution.

Master Device

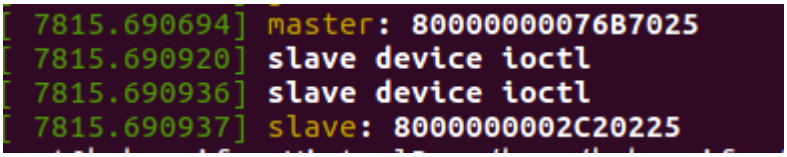
Since the direction of control will always go from Master to Slave, in the beginning we do know some data of the file in the Master. For example, we can know the file size of the Master. Using this data, the whole file used by mmap() will be saved in to different random addresses. After this the master device with the information that files saved in different addresses can read the files from the addresses they are being mapped. According to this if the master processor fails the entire system will fail. All interprocessor communication has to be done through master processor only. The master processor can become overload if it has to manage a number of slave processors and it could lead to low efficiency. So in our project we tried the minimize the buffer size required for the file I/O. We tried to bring file size data as an input file and kept deliver to slave device.

Slave Device

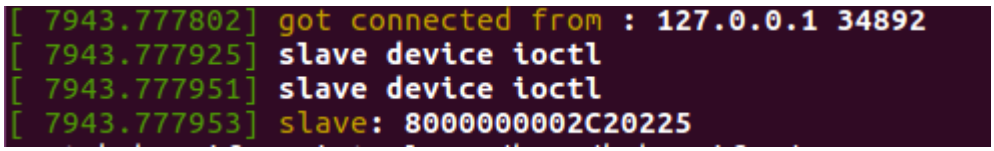
Different from the master device, the thing that is different is that slave device always

gets unknown current file size that has been already used by mmap() prior. Our program will quickly open a new data if the slave device is sure of the data that have been processed ahead is mapped. This routine is repeated until all the inputs have been processed. Since the files are arranged into different mapping address there will be files that has been mapped new and mapped old. To make the order it will need some kind of note to verify the order of the files that have been mapped in the address. If the send of file is complete extra NULL files will be no needed. This is crucial since it will prevent any problems that could occur like opening the files which is empty but being mapped.

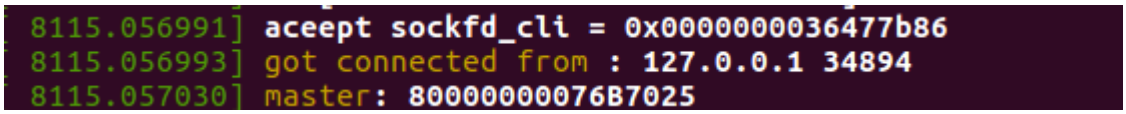
4. *Result of file I/O and memory-mapped I/O*



(master -> master)



(fcntl -> master)



(mmap -> fcntl)

The conclusion of the result is it is that the transmission time of file I/O is smaller than memory-mapped I/O when the data are small. However, if the data are very big, in some point in the transmission, memory-mapped I/O will show better results than the file I/O. It is actually pretty obvious in OS system that as input data goes larger memory-mapped I/O will be much more efficient to use. Some could say when memory-mapped I/O are being used, it requires full decoding which can result into more complex designing of program. However the advantage of memory-mapped I/O is that all instructions and addressing modes can be used for I/O access. So, if we are handling a large data memory-mapped I/O will be much more efficient because it requires less demand than normal I/O. Memory-mapped I/O can simplify the operations as an advantage but there are disadvantages which can become slower.

5. *Members in our group*

i. 方銘浩 B05902082(33%)

- slave.c ,parameter implementation

ii. 陳品學 B06902087(33%)

- master.c ,master_device.c

iii. 柳相宇 B04902126(33%)

- slave_device.c ,test data ,report

iv. 傅子芸(0%)

v. 許家綸(0%)

6. *References*

mmap manual:https://man7.org/linux/man-pages/man2/mmap.2.html

Usage of mmap: https://stackoverflow.com/questions/26259421/use-mmap-in-c-to-write-into-memory

Usage of remap_pfn_range:https://blog.csdn.net/hongkangwl/article/details/12163141

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