Operating System Project03 Report

生機碩一 R09631007 吳乙澤

1 Motivation

1.1 The Problem

The goal of project number three is to run the /test/matmult.c and /test/sort.c concurrently, get the correct result, and don't use the method that simply modify the memory size in machine.h. Before dealing with the memory management problem, I run the malmult.c and sort.c firstly. Look at the following figures, I get assertion failed and Aborted (core dumped) error message. I need to use memory management methods to make the two programs output correct answers, 7220 and 1.

```
yitse@ubuntu:~/nachos-4.0/code/userprog$ ./nachos -e ../test/matmult
Total threads number is 1
Thread ../test/matmult is executing.
Assertion failed: line 136 file ../userprog/addrspace.cc
Aborted (core dumped)
```

```
yitse@ubuntu:~/nachos-4.0/code/userprog$ ./nachos -e ../test/sort
Total threads number is 1
Thread ../test/sort is executing.
Assertion failed: line 136 file ../userprog/addrspace.cc
Aborted (core dumped)
```

Assertion failed and Aborted (core dumped) error

1.2 My plan to deal with the problem

I plan to use the documents from the teaching assistant, google browser, and OS textbook to make virtual memory management and implement FIFO algorithm to do page replacement. The below URL I found, including GitHub and personal blogs, can help me complete the project.

1. https://hackmd.io/@2xu_sb9JT2KDaAH-UKS7PA/rkhnLKuuP#/
2. https://github.com/lyctw/OS2018_NachOS/blob/master/Lab3_Memory_management/README.md
3. https://github.com/srijanshetty/nachos-memory-management
4. http://140.118.125.216/homework/99/OS/homework/homework3/B9715053-hw3-1/random.html
5. http://neuron.csie.ntust.edu.tw/homework/99/OS/homework/homework3/A9715020_hw3-1/the4.htm
6. http://neuron.csie.ntust.edu.tw/homework/99/OS/homework/homework3/B9732005-hw3-1/focus.html

2 Implementation

2.1 The way I implement to solve the problem

According to the documents, in order to make NachOS support virtual memory function, I modify the codes in machine and usrprog folder. I changed some code files, including userkernel.h, userkernel.cc,

machine.h, addrspace.h, addrspace.cc, translate.h and translate.cc to implement virtual memory management.

2.2 Important code segments and comments

Look at the following figures, the figures show some code segments. I add a new variable named vm_Disk in Userkernel.h to store extra pages. Then, modify machine.h, addrspace.h and addrspace.cc to solve the main memory lack problem.

```
class UserProgKernel : public ThreadedKernel {
  public:
    UserProgKernel(int argc, char **argv);
                            // Interpret command line arguments
    ~UserProgKernel();      // deallocate the kernel

    void Initialize();      // initialize the kernel

    void Run();             // do kernel stuff

    void SelfTest();        // test whether kernel is working
    SynchDisk *vm_Disk;     // Save the pages that main memory have not enough memory to save it
```

userkernel.h

After adding virtual memory, I revise translate.cc to implement FIFO algorithm. According to FIFO algorithm contents, I write the below code segments. It's the key point to swap out the first arriving page and add the next page in frame.

```
// Fifo
victim = fifo%32;
printf("The NO.%d page is swap out\n",victim);

// Get the page victm and save in the disk
bcopy(&mainMemory[victim*PageSize],buf_1,PageSize);
kernel->vm_Disk->ReadSector(pageTable[vpn].virtualPage, buf_2);
bcopy(buf_2,&mainMemory[victim*PageSize],PageSize);
kernel->vm_Disk->WriteSector(pageTable[vpn].virtualPage,buf_1);

main_tab[victim]->virtualPage=pageTable[vpn].virtualPage;
main_tab[victim]->valid=FALSE;

// Save the page into the main memory
pageTable[vpn].valid = TRUE;
pageTable[vpn].physicalPage=victim;
kernel->machine->PhyPageName[victim]=pageTable[vpn].ID;
main_tab[victim]=&pageTable[vpn];
fifo = fifo + 1; // For fifo
printf("Page Replacement Done\n");
```

translate.h

3 Result

3.1 Experiment result and some discussion

After my efforts, I still can't run the /test/matmult.c and /test/sort.c concurrently and get the correct result.

Look at the following figures, I got a error message, Segmentation fault, as same as I encounter in the project number two. After one month passed, I learn the theory of segmentation. However, I can't solve it so that I download the NachOS again and modify the program files I mention in 2.1.

```
yitse@ubuntu:~/nachos-4.0/code/userprog$ ./nachos -e ../test/matmult
Total threads number is 1
Thread ../test/matmult is executing.
Segmentation fault (core dumped)
```

```
yitse@ubuntu:~/nachos-4.0/code/userprog$ ./nachos -e ../test/sort
Total threads number is 1
Thread ../test/sort is executing.
Segmentation fault (core dumped)
```

Segmentation fault error

In addition, after I modify the NachOS programs, another error occurred, unexcepted user mode exception2. Though I have not the ability to solve it, the programs print "Page Fault!!!" and "Number = 11 page swap out" message. In my opinion, my completion rate is at least 40%.

```
yitse@ubuntu:~/OSHW/nachos-4.0/code/userprog$ ./nachos -e ../test/matmult -e ../test/sort
Total threads number is 2
Thread ../test/matmult is executing.
Thread ../test/sort is executing.
Page Fault!!!
The NO.11 page is swap out
Page Replacement Done
Unexpected user mode exception2
Assertion failed: line 91 file ../userprog/exception.cc
Aborted (core dumped)
```

Unexcepted user mode exception2 and assertion failed error

3.2 Extra effort or observation

The final test is around the corner, so I start to do the last two years exams downloaded from the professor website. After completing the project, not only do I learn the basic memory management theory, but I also practice the virtual memory management, such as page table and page replacement algorithm.