NachOS

HW3: Multiprogramming & Virtual Memory

I. Multiprogramming

If we use NachOS downloaded from HW1(wget -d https://hsn167.cs.nthu.edu.tw/git/root/nachos/-/archive/master/nachos-master.tar.gz?path=nachos-4.0-hw1 -O nachos4.0-hw1.tar.gz) to run test1 and test2 individually, we get the results shown below.

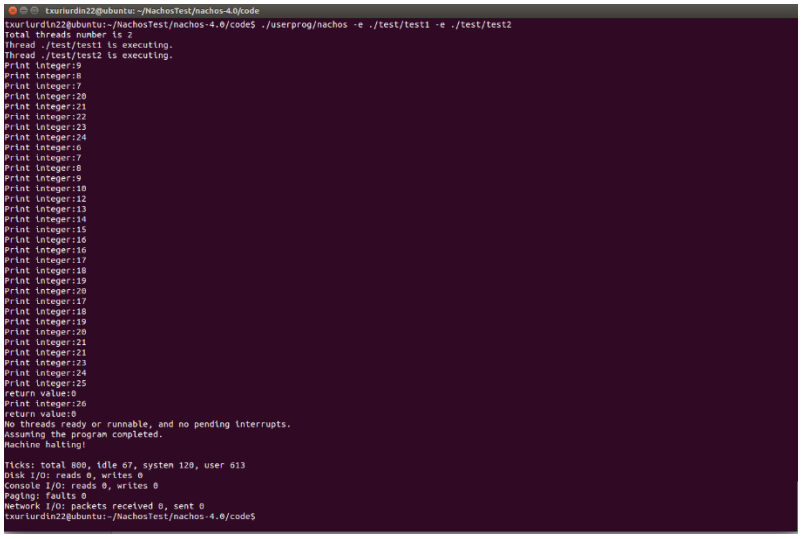
```
txuriurdin22@ubuntu:~/NachosTest/nachos-4.0/code$ ./userprog/nachos -e ./test/test1
Total threads number is 1
Thread ./test/test1 is executing.
Print integer:9
Print integer:8
Print integer:7
Print integer:6
return value:0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 200, idle 66, system 40, user 94
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
```

```
txuriurdin22@ubuntu:~/NachosTest/nachos-4.0/code$ ./userprog/nachos -e ./test/test2
Total threads number is 1
Thread ./test/test2 is executing.
Print integer:20
Print integer:21
Print integer:22
Print integer:23
Print integer:24
Print integer:25
return value:0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 200, idle 32, system 40, user 128
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
```

However, if we run test1 and test2 simultaneously, we get the faulty results.



In HW2, we are able run multiple programs together and output correct results, why? Implement the multiprogramming task.

II. Virtual Memory

We get core dumped if we executed bubble/quick/merge directly.

```
txuriurdin22@ubuntu:~/NachosTest/nachos-4.0/code$ ./userprog/nachos -e ./test/bubble
Total threads number is 1
Thread ./test/bubble is executing.
Assertion failed: line 118 file ../userprog/addrspace.cc
Aborted (core dumped)
```

There is a tip at line 118 in *userprog/addrspace.cc*, hinting us to apply virtual memory for bigger program/programs.

```
118     ASSERT(numPages <= NumPhysPages);       // check we're not trying
119                                             // to run anything too big --
120                                             // at least until we have
121                                             // virtual memory
122
```

You are requested to implement virtual memory, in order to execute the programs individually or simultaneously. You will need to create an empty file and use **ReadAt** and **WriteAt** in *filesys/openfile.h*.

III. Trace code

Trace the following code. Explain (1) *how a program is loaded and executed* (2) *how page faults are handled*. You shall explain as detail as possible in your report to claim better marks.

1. Load program
    a. userprog/userkernel.cc
        - UserProgKernel::Run()
    b. userprog/addrspace.cc
        - AddrSpace::AddrSpace()
    c. threads/kernel.cc
        - ThreadedKernel::Run()
    d. threads/thread.cc
        - Thread::Finish()
    e. threads/thread.cc
        - Thread::Sleep (bool finishing)
    f. threads/scheduler.cc
        - Scheduler::Run (Thread *nextThread, bool finishing)
    g. threads/switch.s
        - SWITCH (thread *t1, thread *t2)
    h. thread/thread.cc
        - ThreadBegin()
    i. userprog/userkernel.cc
        - ForkExecute (Thread *t)
    j. userprog/addrspace.cc

- AddrSpace::Execute (char *fileName)

k. userprog/addrspace.cc
   - AddrSpace::Load (char *fileName)

2. Page Faults

a. machine/ mipssim.cc
   - Machine::Run()

b. machine/ mipssim.cc
   - Machine::OneInstruction(Instruction *instr)

c. machine/translate.cc
   - Machine::ReadMem(int addr, int size, int *value)

d. machine/translate.cc
   - Machine::Translate(int virtAddr, int* physAddr, int size, bool writing)

e. machine/machine.cc
   - Machine::RaiseException(ExceptionType which, int badVAddr)

f. userprog/exception.cc
   - ExceptionHandler(ExceptionType which)

IV. Goal

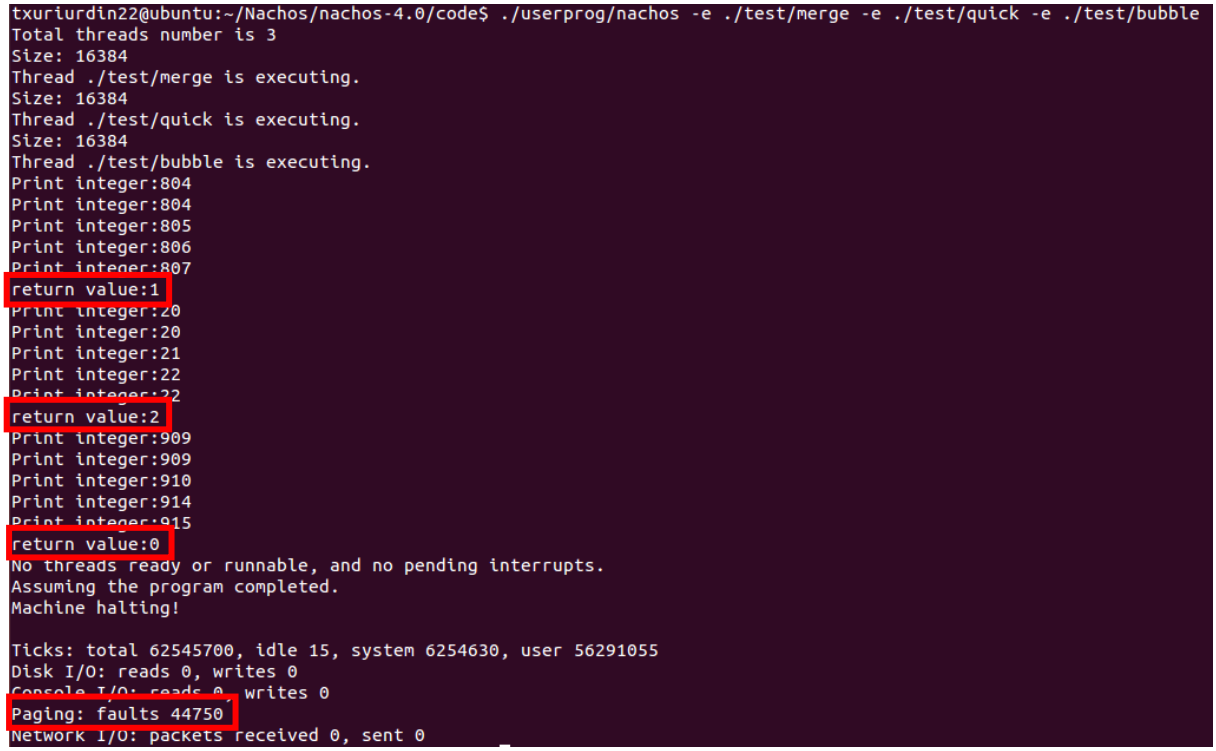
- Make bubble, quick and merge executable together on NachOS. These three programs will be found in ***test*** folder.
- You should put *bubble.c*, *quick.c*, *merge.c* and *array.h* in *test* folder and modify *Makefile* in *test*.

```
36    all: halt shell matmult sort test1 test2 bubble quick merge
```

```
75    bubble: bubble.o start.o
76        $(LD) $(LDFLAGS) start.o bubble.o -o bubble.coff
77        ../bin/coff2noff bubble.coff bubble
78
79    quick: quick.o start.o
80        $(LD) $(LDFLAGS) start.o quick.o -o quick.coff
81        ../bin/coff2noff quick.coff quick
82
83    merge: merge.o start.o
84        $(LD) $(LDFLAGS) start.o merge.o -o merge.coff
85        ../bin/coff2noff merge.coff merge
```

- When bubble finished, we get return value 0. When quick finished, we get return value 1. When merge finished, we get return value 2.
- You can use *sort_.py* to test the sorting results.
- You are request to implement demand paging which means in AddrSpace::Load(char* fileName), the program's code and data should be loaded into a empty file not memory.
- You may implement any page replacement algorithm, modify page size, and page number, but memory size needs to be smaller than 4096(incl.).

- Use **numPageFaults** in *machine/stats.cc* to count the number of page faults and try to make the number as small as possible.
- As merge sort's space complexity is O(n), you will need to increase **UserStackSize** (defined in *userprog/addrspace.h*) to ensure your program will work.
- To execute three programs simultaneously, you could use the following command:
  - ./userprog/nachos -e ./test/merge -e ./test/quick -e ./test/bubble
  - The orders can be reversed.
  - The example of results is shown below.



V. Grading

- Demo: 70pts
  - 60pts: Your page replacement algorithm will be compared to the others. We will consider the number of page faults as evaluation; points distribution is shown below:
    - Top 30%: 60pts
    - 30%-60%: 40pts
    - Others: 20pts
  - 10pts: Questions

        - Everyone must answer a question. You might get different points among your teammates.
    - During demonstration, you will be asked to print out 5 random digits, to ensure your sorting results match our expectations.
- Report: 30pts
- Deadline: **6/22, No late submission allowed**. Please upload your report to iLMS.
- Demo: **6/24, 6/25**. If you are unable to demo on these two days, please email us for further arrangements.
- Feel free to discuss with TAs, and it's encouraged to use the discussion form on iLMS, so others could help or learn.
- **Plagiarism is forbidden and will be punished strictly**.