CS342301: Operating System
MP2: Multi-Programming

Deadline: 2018/11/18 23:59

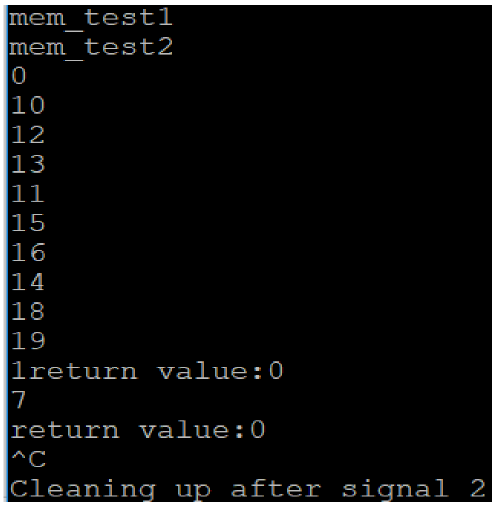
I. Goal

1. Understand how memory management works in NachOS
2. Understand how to implement page table mechanism

II. Assignment

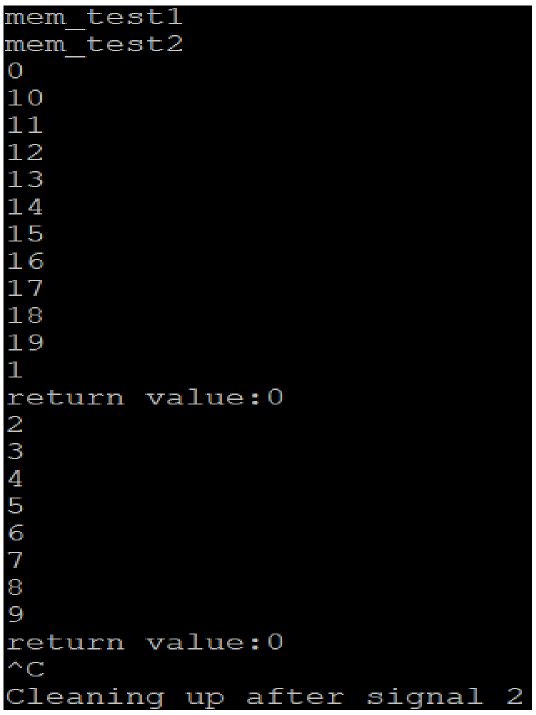
1. Trace code
   - Starting from "threads/kernel.cc **Kernel::ExecAll()**", until "machine/mipssim.cc **Machine::Run()**" is called for executing the first instruction from the user program.

2. Implement page table in NachOS
   - Working item: Modify its memory management code to make NachOS support multi-programming.
   - Verification:
     - ➢ Run "mem_test1" and "mem_test2" at the same time. "mem_test1" prints the number from 0 to 9, and "mem_test2" prints the number from 10 to 19
     - ➢ The original kernel will show a wrong output results like the one below. The correct results with multi-programming will print the numbers from 0 to 19 in output console. Noted, the numbers printed from the two program can be shown out of order on the screen.



```
mem_test1
mem_test2
0
10
12
13
11
15
16
14
18
19
1return value:0
7
return value:0
^C
Cleaning up after signal 2
```

Wrong results



```
mem_test1
mem_test2
0
10
11
12
13
14
15
16
17
18
19
1
return value:0
2
3
4
5
6
7
8
9
return value:0
^C
Cleaning up after signal 2
```

Correct results

   - Hint: The following files "may" be modified…
     - ➢ userprog/addrspace.*
     - ➢ threads/kernel.*

3. Report
   - Cover page, including team members, Team member contribution.
   - Explain your implementation as requested in Part II-2.
   - Explain how a NachOS thread(process) is created, loaded into memory and placed into scheduling queue as requested in Part II-1. Your explanation on the functions along the code path should **at least** cover the answers for the following questions:
     - ➢ How Nachos allocates the memory space for new thread(process)?
     - ➢ How Nachos initializes the memory content of a thread(process), including loading the user binary code in the memory?
     - ➢ How Nachos creates and manages the page table?
     - ➢ How Nachos translates address?
     - ➢ How Nachos initializes the machine status (registers, etc) before running a thread(process)
     - ➢ Which **object** in Nachos acts the role of **process control block**
     - ➢ When and how does a thread get added into the ReadyToRun queue of Nachos CPU scheduler?

III. Instruction

1. Copy your code for MP1 to a new folder
```
$ cp -r NachOS-4.0_MP1 NachOS-4.0_MP2
```
2. Copy the following files on iLMS to your “test” folder
   - mem_test1.c, mem_test2.c and Makefile
3. Test your program
```
$ cd NachOS-4.0_MP2/code/test
$ make clean; make
$ ../build.linux/nachos –e mem_test1 -e mem_test2
```
4. Terminate NachOS
```
$ Crtl +C
```

IV. Grading

1. Implementation correctness – 60%
   - Execute “../build.linux/nachos -e test1 -e test2” with correct output
2. Report – 20%
   - Upload it to iLMS with the Filename: **MP2_report_[GroupNumber].pdf**.
3. Demo– 20%
   - Answer questions during demo.
   - Demo will take place on our server, so you are responsible to make sure your code works on our server.

***Refer to syllabus for late submission penalty.**