

# CS342301 2022 MP2 – Multi-Programming

Deadline: 2022/11/13 23:59

## ★ Goal

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

## ★ Assignment

### ○ Trace code

- Starting from “threads/kernel.cc **Kernel::ExecAll()**”, “threads/thread.cc **thread::Sleep()**” until “machine/mipsim.cc **Machine::Run()**” is called for executing the first instruction from the user program.
  - You need to explain at least the function mentioned below in the report.
- 1-1. threads/thread.cc

```
Thread::Sleep()
Thread::StackAllocate()
Thread::Finish()
Thread::Fork()
```

1-2. userprog/addrspace.cc

```
AddrSpace::AddrSpace()
AddrSpace::Execute()
AddrSpace::Load()
```

1-3. threads/kernel.cc

```
Kernel::Kernel()
Kernel::ExecAll()
Kernel::Exec()
Kernel::ForkExecute()
```

1-4. threads/scheduler.cc

```
Scheduler::ReadyToRun()
Scheduler::Run()
```

- Implement page table in NachOS

2-1. Working item: Modify its memory management code to make NachOS support multiprogramming.

2-2. Verification:

- Wrong results without multiprogramming

```
[ta@lsalab ~/2020/riya/MP2_test/code/test]$ ../build.linux/nachos -e consoleIO_test1 -e consoleIO_test2
consoleIO_test1
consoleIO_test2
9
16
15
18
19
1return value:0
7
return value:0
```

- Correct results with multiprogramming

```
[ta@lsalab ~/2020/riya/MP2_sol/code/test]$ ../build.linux/nachos -e consoleIO_test1 -e consoleIO_test2
consoleIO_test1
consoleIO_test2
9
8
7
6
1return value:0
5
16
17
18
19
return value:0
```

- Correctly handle the exception about insufficient memory (Details in 2-3 requirement)

```
[ta@localhost test]$ ../build.linux/nachos -e consoleIO_test1 -e consoleIO_test4
consoleIO_test1
consoleIO_test4
9Unexpected user mode exception 8
Assertion failed: line 201 file ../userprog/exception.cc
Aborted
```

2-3. Requirement:

- Be careful that program size might exceeds a pagesize
- You must put the data structure recording used physical memory in **kernel.h** / **kernel.cc**
- You must set up “valid, readOnly, use, and dirty” field for your page table, which is defined under “**TranslationEntry class**” in translate.h
- You must call ExceptionHandler to handle the exception when there is insufficient memory for a thread. (i.e. MemoryLimitException).

Since there is no MemoryLimitException in ExceptionType class, you need to add it in machine.h and place it right before **NumExceptionTypes**. **(0 points for using existing exception type or placing it in the wrong index.)**

Also, you need to create the test cases to verify the correctness .TAs will use hidden test cases to test.

- When the thread is finished, make sure to release the address space and restore physical page status.

2-4. Hint: The following files “may” be modified...

- **userprog/addrspace.\***
- **threads/kernel.\***

- Report

- Cover page, including team members, Team member contribution.
- Explain your implementation.
- Explain how NachOS creates a thread (process), load it into memory and place it into the scheduling queue as requested in the **Trace code** part.

Your explanation on the functions along the code path should at least cover answer for the questions below:

- How does Nachos allocate the memory space for a new thread(process)?
- How does Nachos initialize the memory content of a thread(process), including loading the user binary code in the memory?
- How does Nachos create and manage the page table?
- How does Nachos translate addresses?
- 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?

## ★ Instructions

1. Copy your code for MP1 to a new folder

```
$ cp -r NachOS-4.0_MP1 NachOS-4.0_MP2
```

2. Copy test file

```
$ cp /home/os2022/share/consoleIO_test* NachOS-4.0_MP2/code/test/
```

3. Compile / Rebuild NachOS

```
$ cd NachOS-4.0_MP2/code/build.linux
```

```
$ make clean
```

```
$ make
```

4. Test your program

```
$ cd NachOS-4.0_MP2/code/test
```

```
$ ../build.linux/nachos -e consoleIO_test1 -e consoleIO_test2
```

## ★ Grading

1. Implementation correctness – 60%
  - (a) Execute “../build.linux/nachos -e consoleIO\_test1 -e consoleIO\_test2” correctly
  - (b) There will be hidden test cases to test the requirements. Make sure your code meets all the requirements.
  - (c) Your working directory will be copied for validation after the deadline.
2. Report – 20%
  - (a) Upload to eeclass with the filename **MP2\_report\_<Group Number>.pdf**
3. Demo – 20%
  - (a) Demonstrate your implementation, and answer questions from TAs in **15 minutes**.
  - (b) Some random test cases will be used for correctness verification.
  - (c) Demos will take place on our server, so you are responsible to make sure your code works on our server.
4. Refer to the syllabus for late submission penalty.
5. **Plagiarism**
  - (a) **NEVER SHOW YOUR CODE** to others.
  - (b) If the codes are similar to other people (**including your upperclassman**) and you can't answer questions properly during the demo, you will be identified as plagiarism.