國立陽明交通大學
NATIONAL YANG MING CHIAO TUNG UNIVERSITY

Institute of Artificial Intelligence Innovation
Department of Computer Science

Operating System

Homework 03: CPU Scheduling (part1)

Shuo-Han Chen (陳碩漢),
shch@nycu.edu.tw

Thur. 13:20 - 16:20 ED305

Goal

1. The default CPU scheduling algorithm of Nachos is a simple round-robin scheduler for every 100 ticks.
2. The goal of HW3 is to replace it with a priority scheduling strategy.

Part 1 Trace Code

- Explain the purposes and details of the following 6 code paths to understand how nachos manages the lifecycle of a process (or thread) as described in the Diagram of Process State.

1. New -> Ready

   Kernel::ExecAll -> Kernel::Exec -> Thread::Fork -> Thread::StackAllocate -> Scheduler::ReadyToRun

2. Running -> Ready

   Machine::Run -> Interrupt::OneTick -> Thread::Yield -> Scheduler::Find NextToRun -> Scheduler::ReadyToRun -> Scheduler::Run

3. Running -> Waiting
4. Waiting -> Ready
5. Running -> Terminated
6. Ready -> Running

Part 1 Trace Code (cont'd)

- Explain the purposes and details of the following 6 code paths to understand how nachos manages the lifecycle of a process (or thread) as described in the Diagram of Process State.
  1. New -> Ready
  2. Running -> Ready
  3. Running -> Waiting

     SynchConsoleOutput::PutChar -> Semaphore::P -> List<T>::Append ->
     Thread::Sleep -> Scheduler::FindNextToRun -> Scheduler::Run
  4. Waiting -> Ready

     Semaphore::V -> Scheduler::ReadyToRun
  5. Running -> Terminated

     ExceptionHandler(ExceptionType) case SC_Exit -> Thread::Finish() ->
     Thread::Sleep -> Scheduler::FindNextToRun -> Scheduler::Run
  6. Ready -> Running

Part 1 Trace Code

- Explain the purposes and details of the following 6 code paths to understand how nachos manages the lifecycle of a process (or thread) as described in the Diagram of Process State.
  1. New -> Ready
  2. Running -> Ready
  3. Running -> Waiting
  4. Waiting -> Ready
  5. Running -> Terminated
  6. Ready -> Running

     Scheduler::FindNextToRun -> Scheduler::Run -> SWITCH(Thread*, Thread*) -> (depends on the previous process state, e.g., [New,Running,Waiting]→Ready) -> for loop in Machine::Run()

Part 2 Prerequisite

- Before you start implement this homework feature, you need to ensure your previous NachOS assignments meet the following requirements.
    1. Complete the requirements for Homework 2 to ensure that your Nachos system is capable of running multiple processes correctly.

Part 2 Modification

- To observe scheduling easily by PrintInt(), change ConsoleTime to 1 in machine/stats.h

```
const int ConsoleTime = 1;
```

- Comment out current Alarm::Callback to cancel the Round Robin Mechanism.

```
  // if (status != IdleMode) {
  // interrupt->YieldOnReturn();
  // }
```

- Comment out postOffice at Kernel::Initialize() and Kernel::~Kernel() in kernel.cc

```
// postOfficeIn = new PostOfficeInput(10);
// postOfficeOut = new PostOfficeOutput(reliability);
// delete postOfficeIn;
// delete postOfficeOut;
```

Part 2 Modification (cont'd)

- Modified and add test case for better debugging.

ConsoleIO_test1

```c
#include "syscall.h"

int
main()
{
    int n;

    for (n=0; n < 4; n++) {
        PrintInt(1);
    }
    return 0;
}
```

ConsoleIO_test2

```c
#include "syscall.h"

int
main()
{
    int n;

    for (n=0; n < 5; n++) {
        PrintInt(2);
    }
    return 0;
}
```

ConsoleIO_test3

```c
#include "syscall.h"

int
main()
{
    int n;

    for (n=0; n < 12; n++) {
        PrintInt(3);
    }
    return 0;
}
```

Part 2 Implementation

- Implement a **Priority Queue** Scheduler as described below:
  1. All processes must have a valid scheduling priority between 0 to 149. Higher values means higher priority. So 149 is the highest priority, and 0 is the lowest priority.
  2. In Homework 3, you are required to implement only the Priority Queue Strategy, meaning all threads will be in the same queue. However, in the future, we plan to enhance this to include three different strategies, each with its own queue. Please keep this in mind while implementing your work.
  3. **Priority Queue** uses a non-preemptive priority scheduling algorithm. A thread won't preempt other threads. If two threads enter the queue with the same priority, either one of them can execute first.

Part 2 Implementation (cont'd)

- Add a command line argument "-ep" for nachos to initialize priority of process.
- E.g., the command below will launch 2 processes: test1 with priority 40, and test2 with priority 80.

```
$ ../build.linux/nachos –ep test1 40 –ep test2 80
```

Part 2 Implementation (cont'd)

- Add a debugging flag "z" and use the DEBUG('z', expr) macro(defined in debug.h) to print following messages. Replace "{...}" to the corresponding value.
  1. Whenever a process is inserted into a priority queue

     [A] Tick [{current total tick}]: Thread [{thread ID}] is inserted into queue
  2. Whenever a process is removed from a queue

     [B] Tick [{current total tick}]: Thread [{thread ID}] is removed from queue
  3. Whenever a process changes its scheduling priority

     [C] Tick [{current total tick}]: Thread [{thread ID}] changes its priority from [{old value}] to [{new value}]
  4. Whenever a context switch occurs

     [D] Tick [{current total tick}]: Thread [{new thread ID}] is now selected for execution, thread [{prev thread ID}] is replaced, and it has executed [{accumulated ticks}] ticks

Part2 Verification Example

```
machine@machine:~/code/test$ timeout
1 ../build.linux/nachos -ep conso
leIO_test1 70 -ep consoleIO_test3 80 -ep
consoleIO_test2 50
consoleIO_test1 with priority 70
consoleIO_test3 with priority 80
consoleIO_test2 with priority 50
3
3
3
3
3
3
3
3
3
3
return value:0
1
1
1
1
return value:0
2
2
2
2
2
return value:0
```

```
machine@machine:~/code/test$ timeout 1 ../build.linux/nachos -d z -ep c
onsoleIO_test1 70 -ep consoleIO_test3 80 -ep consoleIO_test2 50
consoleIO_test3 with priority 80
consoleIO_test2 with priority 50
[D] Tick [0]: Thread [0] (main thread) starts its execution
[A] Tick [10]: Thread [1] is inserted into queue L[2]
[A] Tick [20]: Thread [2] is inserted into queue L[2]
[B] Tick [30]: Thread [1] is removed from queue L[2]
[D] Tick [30]: Thread [1] is now selected for execution, thread [0] is replaced,
and it has executed [30] ticks
3[B] Tick [69]: Thread [2] is removed from queue L[2]
[D] Tick [69]: Thread [2] is now selected for execution, thread [1] is replaced,
and it has executed [39] ticks
[A] Tick [79]: Thread [1] is inserted into queue L[2]
[B] Tick [98]: Thread [1] is removed from queue L[2]
[D] Tick [98]: Thread [1] is now selected for execution, thread [2] is replaced,
and it has executed [29] ticks

// ...

[A] Tick [988]: Thread [2] is inserted into queue L[2]
[B] Tick [988]: Thread [2] is removed from queue L[2]
[E] Tick [988]: Thread [2] is now selected for execution, thread [2] is replaced,
and it has executed [273] ticks
return value:0
```

Hint

- The following files “may” be modified...
  - threads/kernel.*
  - threads/thread.*
  - threads/scheduler.*
  - threads/alarm.*
  - lib/debug.*

Jenkins verification

- The TA's job will involve running three tests.
  1. consoleIO_test1 60 consoleIO_test2 70

     2 2 2 2 2 1 1 1 1 …
  2. consoleIO_test1 70 consoleIO_test2 60

     1 1 1 1 2 2 2 2 2 ….
  3. consoleIO_test1 70 consoleIO_test3 80 consoleIO_test2 50

     3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 2 2 2 2 2 …

Grading

- Part1 (Trace) – 36%
    1. Each path – 6%
- Part2 (Implementation) – 72%
    1. Priority Queue Correctness – 60%
    2. Debug Flag – 12%
- Report Format – 2%
- Deadline: 12/14 (23:59)

Report Format

- Please follow the word file to form your report for HW03
- Format guide
  - Content format: should be set with 12pt front,16pt row height, and align to the left.
  - Caption format: 18pt and Bold font.
  - Font format: Times New Roman, 標楷體
  - Figure: center with single line row height.
  - Change the title to your student ID and name in Chinese.
  - Upload pdf file with the file name format : OS_HW03_GROUP_X.pdf (change X to your group ID)

Reminder

- 0 will given to cheaters. Do not copy & paste!
  - TA will check your repository
- Feel free to ask TA questions
  - Teams Message(Recommended): 廖永誠
  - Email: yongchengliaw.ii12@nycu.edu.tw

Q & A

Thank you for your attention