HW2 - CPU Scheduler

Speaker: Wei-Xiang Wang

M10715010@mail.ntust.edu.tw

SelfTest

- 有不少 class 都會有自己的測試程式，可以在這個程式區塊寫上自己要測試的程式碼，而可以暫時不用透過test資料夾下的測試程式來測試。
- 例如 thread/thread.cc 裡面就有 Thread::SelfTest()

Debugger

- Internal Debugging
  - 透過 DEBUG 函數來實現，並可定義自己的 debug 類型
- External Debugger
  - 透過 GDB 進行偵錯，可實現單步執行、中斷點等
- Reference:
  - http://puremonkey2010.blogspot.com/2013/03/nachos-40-debugging-nachos.html
  - https://welkinchen.pixnet.net/blog/post/60935296-%E9%99%A4%E9%8C%AF%E5%B7%A5%E5%85%B7-gdb-%28the-gnu-project-debugger%29

Git pull

- 我有在原先的 NachOS 專案進行一些更新，請在你的專案，利用 git pull 拉取更新。
  - 在更新時可能會遇到衝突，請解決衝突後再推上你們的 github

- Reference:
  - https://ithelp.ithome.com.tw/articles/10209306

Homework Requirements - 1

- 實現 multi-programming (multi-thread)
- 目前 NachOS 載入兩個程式起來後，會覆蓋在同一個記憶體位子(code segment)，需要為實體記憶體位址做一些轉換。
- 記憶體衝突的程式碼主要在 userprog/addrspace.cc

```
ubuntu@ubuntu:/tmp/NachOS/code/userprog$ ./nachos -e ../test/test1 -e ../test/te
st2
Total threads number is 2
Thread ../test/test1 is executing.
Thread ../test/test2 is executing.
Print integer:9
Print integer:8
Print integer:7
Print integer:20
Print integer:21
Print integer:22
Print integer:23
Print integer:24
Print integer:6
Print integer:7
Print integer:8
```

Homework Requirements - 2

- 先進先出排程 (First-Come-First-Service, FCFS)
- 最短工作優先排程 (Shortest-Job-First, SJF)
  - 不可搶占
  - 預估下一次時間長度
    - 利用前幾次 CPU bursts 所測得的值的指數平均數來做為排班依據
    - $\tau_{n+1}=\alpha\, t_n + (1-\alpha)\, \tau_n$
- 最短剩餘時間排程 (shortest remaining time first ,SRTF)
  - 類似於 SJF，但是可被搶占行程
- (可選題) 輸出每個測試程式使用的分割時間(brust time)，平均等待時間(average waiting time)
  - Tip: kernel->stats->userTicks 是執行 user program 的時間累加器

Homework Requirements - 2

› Example – First-Come-First-Service scheduling

```
ubuntu@ubuntu:~/NachOS/code/userprog$ ./nachos -d w -e ../test/test1 -prio 4 -e ../test/test2 -prio 3 -FCFS
Total threads number is 2
Thread ../test/test1 is executing.
Thread ../test/test2 is executing.
Print integer:9
Print integer:8
Print integer:7
Print integer:6
return value:0
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 300, idle 28, system 50, user 222
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
ubuntu@ubuntu:~/NachOS/code/userprog$
```

Homework Requirements - 2

- Example – Priority scheduling

```
ubuntu@ubuntu:~/NachOS/code/userprog$ ./nachos -e ../test/test1 -prio 4 -e ../test/test2 -prio 2 -PRIORITY
Total threads number is 2
Thread ../test/test1 is executing.
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
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 300, idle 8, system 70, user 222
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
ubuntu@ubuntu:~/NachOS/code/userprog$
```

Tip1

- Trace the following files
  - code/threads/alarm.cc
  - code/threads/thread.cc
  - code/threads/scheduler.cc
  - code/threads/kernel.cc
  - code/threads/main.cc
  - code/machine/interrupt.cc
  - code/userprog/addrspace.cc
  - code/userprog/userkernel.cc
  - code/lib/list.h for SortedList

Tip2

- 若測試程式同時執行太多個可能會造成 stack overflow，透過把 code/machine/machine.h 的 NumPhysPages 加大即可。
- CPU 的 timer 計數時間量定義在 code/machine/stats.h
  - const int TimerTicks = 100;

Hand in source code & report

- Source Code
  - Push it to your GitHub
- Report
  - The work distribution of team members
  - The idea you take to implement the problem in NachOS
  - Some important codes and comments
  - Experiment result (Screenshot)
  - Saved as [Team ID]_HW2_report.pdf
    - e.g. 2_HW2_report.pdf
  - Upload the report to moodle