# Operating Systems

Not Project 2 CPU Scheduling

Another

Completely

Heuristic

Operating

System

Speaker : Wei-Ting Lu

# Before we implement CPU scheduling

## We need to implement system call : Sleep()

1. userprog/syscall.h - Define a system call number of Sleep
2. test/start.s - Prepare registers for Sleep
3. userprog/exception.cc - Add a new case for Sleep in ExceptionHandler and Note the use of kernel->alarm->WaitUntil()

## You can try implement some other kinds of system call

If so, please explain what system call you implement and write in your report
It will be a bonus!

# Details about implementation

## test/start.s

Please follow the implemented system calls in start.s

Some symbols you may not know(like *addiu*), please refer to the following link that show how assembly language assist to make system calls to the Nachos kernel
http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html

## userprog/exception.cc

You also need to modify */code/threads/alarm.h* and *alarm.cc* files and the function WaitUntil() will be called when a thread going to sleep. After that, please call the function CallBack() to check which thread should wake up.

See the comments in the file, they are helpful

Don’t forget the useful data structure like list in the *lib* folder

# How to write my own test code?

Write it in the same way as other test code

1. Create *.*c* in code/test/
2. Modify the Makefile in code/test/ in the same way as test1 and test2
3. Type “make” in code/test/

* could be the file name you like

Now, we can start to implement CPU Scheduling!

# CPU Scheduling

## Choose at least ONE of the following to implement:

1. First-Come-First-Service(FCFS)
2. Shortest-Job-First(SJF)
3. Priority
4. Otherwise

## The extra implementation will be a bonus

1. You can also choose any scheduling algorithm from lecture
2. Specify your algorithm in the report

## Design your own test code:

1. You can find Class::SelfTest() in many classes.
2. Implement some test code, and call it in SelfTest()
3. You may modify the following files:
   - 3.1 Recall sleep(): threads/thread.cc and thread.h
   - 3.2 To call your test code in ThreadedKernel : threads/kernel.cc
   - 3.3 To change the program interface : threads/main.cc
   - 3.4 To make your own SelfTest() function : threads/alarm.cc and alarm.h
   - 3.5 To implement schedulers : threads/scheduler.cc and scheduler.h
   - 3.6 Useful data structure : lib/list.h for SortedList

# CPU Scheduling

Design at least 2 test cases to proof your result

1. Specify the test case setting and plot that screenshot in your report

Design the nachos interface to switch different scheduling algorithm if you implement more than one

# Report

1. What scheduling algorithm you implemented
2. How you implement the scheduling algorithm in Nachos
3. Experiment result and some discussion
4. Extra effort (e.g. bonus part) or observation
5. Remember there are two parts(system call and scheduling) in project2
6. Must screenshop the result in your report

Please saved as [Student ID]_NachOS_report.pdf
E.g. r123456789_NachOS_report.pdf

# Bonus

Implement more than one scheduling algorithm

1. Explaining how to switch different scheduling algorithm in your program in the report or the README
2. You can implement any kind of scheduling algorithm

Extra observation or modification on nachos

The bonus works must write in report

# Hand in report

## Code

tar zcvf b99xxxxxx.tar.gz ./nachos-4.0

## Mail your code and report to TA before the deadline:

argonmisir@gmail.com

# Grading Policy
# With presentation

Nachos source code: 40 %

Sleep() : 20%

Scheduling : 20%

Report : 30%

Correct format : 10%

Presentation: 20%

Bonus

# Grading Policy
# Without presentation

Nachos source code: 50 %

- Sleep() : 25%
- Scheduling : 25%

Report : 40%

Correct format : 10%

Bonus

# Reference

Nachos System Call Implementation Sample:
http://puremonkey2010.blogspot.tw/2013/05/nachos-40-nachos-system-call.html

Nachos Beginner’s Guide:
https://www.ida.liu.se/~TDDI12/material/begguide/