Project 2 - System Call & CPU Scheduling

Part 1. System call - Sleep()

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

Part 1. System call - Sleep()

- Alarm - Software alarm clock, that generates an interrupt every X time ticks
  - threads/alarm.h
- The **WaitUntil()** will be called when a thread going to sleep
- Call the **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

Part 1. System call - Sleep()

- Write your own **test code** like test1 and test2
  - Create test.c in test/
  - Modify the Makefile in test/
  - Compile in the same way as test1 and test2

Part 2. CPU scheduling

- Choose **at least one** of the following to implement
  - First-Come-First-Service(FCFS)
  - Shortest-Job-First(SJF)
  - Priority
  - Other scheduling method taught in class
- The extra implementation will be considered as BONUS
- Design your own test code
  - Modify Thread::SelfTest() or create your own method for testing

Part 2. CPU scheduling

- Design at least 2 test case to proof your result
  - Specify your scheduling method in the report
  - Specify the test case setting and output in your report
  - Design the NachOS interface to switch different scheduling method, if you implement more than one scheduling method
    - threads/main.cc

Report

- Motivation
  - Motivation and the problem analysis
  - Wha’t your plan to deal with the problem
- Implementation
  - How do you implement to solve the problem in NachOS
  - You can include some of your code and explain it
- Result
  - Experiment result and some discussion
  - Extra effort or observation
- Save your report as the format report.pdf

Submission format

- Create a folder for the source code and report
  - {Student ID}_nachos2/
    - nachos-4.0/
    - report.pdf
- Submission format: {Student ID}_nachos2.tar.gz
  - e.g. r11223344_nachos2.tar.gz
- Compress your folder
  - tar zcvf {Student ID}_nachos2.tar.gz {Student ID}_nachos2/
- Submit your compressed file to NTU cool
- Deadline: 2022/11/18 23:59

Grading policy

- NachOS source code:(40%)
  - Part 1.(20%)
  - Part 2.(20%)
  - Bonus of Part 2.(10%)
    - Get the bonus if you implement more than one scheduling method
- Report: (40%)
- Correct format: (20%)
- No plagiarisim

Late policy

- 10% penalty per day
- After 7 days, you will get 70% penalty, but no more penalty after that

Reference

- The University of Chicago - Com Sci 230 System Calls and Other Exceptions
- Purdue University - CS 354 Thread Scheduling and Sleeping