CMSC412 DISCUSSION

Project 3 [Due Friday, October 19 @ 6:00pm]

Review

- Questions about Project 2

Project 3 Requirements

- Add a custom scheduler
  - Sys_SetSchedulingPolicy
  - Sys_GetTimeOfDay
- Add semaphores
  - Note: These functions must all be done *atomically*
  - Sys_Open_Semaphore
  - Sys_Close_Semaphore
  - Sys_P
  - Sys_V

Current Scheduler

- Round Robin
  - At every time slice, or “quantum”, the next thread in the runnable thread list is chosen to run.
- Issues
  - This punishes IO heavy processes, and rewards CPU intensive processes.

Implementation – Custom Scheduler

- In /src/geekos/syscall.c:
  - Implement Sys_SetSchedulingPolicy()
    - Change some global flag to indicate which scheduler should be operating (either custom or round-robin)
  - Implement Sys_GetTimeOfDay()
    - Return g_numTicks from timer.c
- In /src/geekos/kthread.c:
  - Change Get_Next_Runnable() to chose the next thread based on the global flag set earlier
    - It is up to you how you chose the next thread in your custom implementation, but it is necessary to be faster than the default scheduler.
    - Suggest looking at http://en.wikipedia.org/wiki/Scheduling_(computing)
- Write a README.scheduler per the Project 3 Specification

Implementation - Semaphores

- In /src/geekos/sem.c:
  - Add some data structure for a semaphore containing the following:
    - *name*: at most 25 characters
    - *semaphore id* (or *SID*): integer
    - *value*: non-negative integer
    - *open users*: set of user threads that currently have the semaphore “open”
    - Anything else you deem necessary
  - Implement Sys_Open_Semaphore:
    - Look at the current open semaphores to see if a semaphore with the specified name already exists
    - If not, create a new one (unless there are too many [20] semaphores already, then return ENOSPACE)
    - With the found or newly created semaphore, add the current process to the list of “open users”
  - Implement Sys_Close_Semaphore:
    - Remove the current process from the list of “open users”
    - If no more users exist, return semaphore to pool of available semaphores

Implementation – Semaphores (cont.)

- In /src/geekos/sem.c:
  - Implement Sys_P:
    - *Atomically* check the value of the specified semaphore
    - If value > 0:
      - Return 0, after decrementing value by 1
    - Otherwise:
      - Block – thread should not be in run queue anymore (use Wait() in kthread.c)
      - Once value > 0, perform action as described above
  - Implement Sys_V:
    - *Atomically* increment the value of the specified semaphore
    - This should release a (or any) blocked thread(s) waiting on the semaphore (use Wake_Up_One() or Wake_Up() in kthread.c)
      - You have some freedom in how to design this, look at Project Specification for more details
- In /src/geekos/kthread.c:
  - Exit() must now release (aka Close_Semaphore()) any semaphores a process has open.