CSE 451 Homework 3

May 7 2009
USEFUL/OBVIOUS TIPS

• Due Monday May 11th before class starts

• Read the write-up *multiple* times 😊
• Read the GoPosts about the HW
• Post questions to the GoPost
• Write strong tests (at least 1/3 of points)
Part 1 - Semaphore Package

• Use pthread condition variables and mutex
  – Pthread_cond_wait(), pthread_cond_broadcast(),
    pthread_mutex_lock(), pthread_mutex_unlock()

• You may **not** use any semaphore libraries

• Decide on an interface:
  – typedef struct _semaphore{ ... } semaphore;
  – initialize(s, start_value)
  – wait(s) //P
  – signal(s) //V
  – etc...
Part 2 – Bounded Buffer Pro/Con

• Use your semaphore package here
• Nice if this package is generic enough to handle all types, though not required
• Interface should handle an arbitrary amount of producers and consumers
• Examples: `add_to_buffer(buf_t buf, item_t item)`, `item_t consume_from_buffer(buf_t buf)`
Part 2 - continued

• A couple of counting semaphores to handle empty and full buffers
• A binary semaphore to enforce mutual exclusion
• See lecture 8 (specifically slide 10)
Part 3 – Crack the key

• One producer thread and multiple consumer threads (one per core)
• You are trying to crack 4 bytes
  – Write-up says use blocks of 1024 keys
    
    ```
    add_to_buffer(buf, 0);
    add_to_buffer(buf, 1024);
    add_to_buffer(buf, 2048);
    ```
  
  Note: You don’t have to add every key to the buffer
• Consume from a buffer and try all 1024 keys starting from the value just consumed
• See example code for how to use AES encryption and decryption
  
MIDTERM

Wednesday May 13, 2009

- Kernel vs User
  - system calls, protection bit

- Processes/threads
  - Address space, process state, fork, context switches, kernel vs user, shared memory or message passing

- Scheduling
  - Tradeoffs between different algorithms, avg turnaround time & avg wait time, fairness, preemption

- Synchronization
  - Critical sections, atomic instructions, locks, mutex, semaphores, monitors

- MM
  - Fragmentation, paging, virtual memory, TLBs, page replacement