Project Overview

Task 1: Server Access (Task 0 –Sever access.pdf)

- SSH key generation
  - Linux and Mac: *ssh-keygen -t rsa*
  - Windows: PuttyGen
- Login the course server:
  - Linux and Mac: *ssh f20-Xl-gY@klwin00.eng.ucmerced.edu*
  - Windows: Putty
- Type bash as soon as you login
- Group representative: Navigate to the *.ssh* directory by executing cd *.ssh* and copy the contents of *id_rsa.pub* from your team member to the *authorized_keys* file. Make sure that different public keys appear on different lines.

Task 2: KThread.join()

Two threads, A and B

- A thread must not attempt to join with itself (A and B are not the same thread)
- B joins with A. Check to see if the status of the thread A is already finished
  - If it is, thread B returns immediately
  - If not, thread B waits inside of the join until A finishes, then it resumes B
- Join can be called on a thread at most once. If thread B calls join on A, then it is an error for B or any other thread C to call join on A again
- Prevent two threads from trying to join with another thread at the same time

Task 2: KThread.join()

- Only one other aspect is needed to finish implementation; in KThread.finish() , the currentThread will go through each thread waiting on it and wake them. This will successfully restore all waiting threads to a ready state
- KThread.java
  - join()
  - finish()

Task 3 : Implement condition variables using interrupt enable and disable

- Figure out condition.java (using semaphore)
- Reimplement condition.java in condition2.java
  - sleep()
  - wake()
  - wakeAll()

Task 4: Alarm

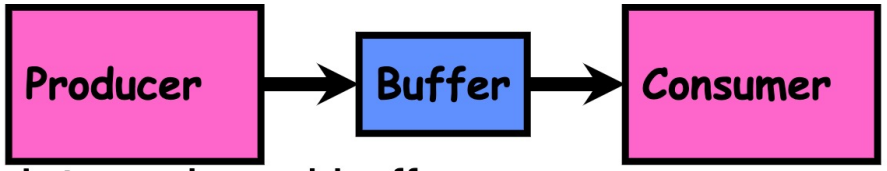
- Whenever a thread calls waitUntil() , we take the current system time and add the desire wait time to it. This means we end up with an absolute system time at which the thread wants to resume.
- Need an appropriate data structure to sort the absolute waiting time
- Alarm.java

  timerInterrupt()

  waitUtil()

Task 5: Communicator

- Producer-consumer with a bounded buffer(slides 8, page 18)



  - Problem Definition
    - Producer puts things into a shared buffer
    - Consumer takes them out
    - Need synchronization to coordinate producer/consumer
  - Don’t want producer and consumer to have to work in lockstep, so put a fixed-size buffer between them
    - Need to synchronize access to this buffer
    - Producer needs to wait if buffer is full
    - Consumer needs to wait if buffer is empty
  - Example: Coke machine
    - Producer can put limited number of cokes in machin
    - Consumer can’t take cokes out if machine is empty
    - They cannot work at the same time.

Task 6: Boat

- With a singular thread perspective, the solution for the boat problem requires the following procedure:
- While (adults on Oahu)

  Send boat with two children from Oahu to Molokai

  Return boat with one child from Molokai to Oahu

  Send boat with one Adult from Oahu to Molokai
- While (children on Oahu > 2)

  Send boat with two children from Oahu to Molokai

  Return boat with one child from Molokai to Oahu

  Send boat with two children from Oahu to Molokai

Group Projects

- Communication and cooperation will be essential
  - Regular meetings.
  - Slack/Messenger/whatever doesn't replace face-to-face!

- Everyone should do work and have clear responsibilities
  - You will evaluate your teammates at the end of each project.
  - Dividing up by Task is not a good approach. Work as a team.