CS 353 Project 2
Deterministic Scheduler

Dohyeong Kim
Deterministic Scheduler

● Goal
  ○ Always execute a concurrent program in the exact same order.
  ○ Execute one thread at a time.

● Implementation details
  ○ Global Lock
  ○ Next Thread
Global Lock

- One global mutex lock for all threads.
- Only the thread holding the lock can be executed.
  - Every thread should acquire the lock before it begins its own execution.
    - e.g. Before a new thread starts, before a thread wakes up from waiting, ...
  - Every thread should release the lock before it ends or pause its own execution.
    - e.g. Before a thread finishes, before a thread waits for a mutex, ...
Next Thread

- Only the selected next thread can hold the global lock.
  - If current thread is not the selected next thread, wait till current thread is selected.

- LOCK(GL)
  - while (true)
    - PLOCK(GL)
      - if (currentThread == nextThread)
          - break;
      - else
          - PRELEASE(GL);
Next Thread

- Every thread should select a next *available* thread before it releases the global lock.
  - The selected thread should be *available* to execute.
    - it should not be waiting for a mutex or another thread.
Algorithm

● Enter a thread / a new thread starts
  ○ LOCK(GL)
  ○ // begin execution

● Leave a thread / a thread finishes
  ○ // select a next available thread
  ○ UNLOCK(GL)
  ○ // terminate execution
Algorithm

- `pthread_join(joinee)`
  - if (joinee is still running)
    - // select a next available thread
    - UNLOCK(GL)
    - // wait till joinee is terminated
    - LOCK(GL)
Algorithm

- `pthread_mutex_lock(L)`
  - if (L is held by another thread)
    - // select a next available thread
    - UNLOCK(GL)
    - // wait till L is not held by any other threads
    - LOCK(GL)
    - LOCK(L) // this should always succeed
  - else
    - LOCK(L)

- `pthread_mutex_unlock(L)`
  - UNLOCK(L)
Algorithm

- sched_yield()
  - if (another available thread exists)
    ▪ // select a next available thread
    ▪ UNLOCK(GL)
    ▪ LOCK(GL)
You may also need to implement

- A list of threads
- Status of threads
  - e.g. available, waiting for mutex, ...
- Status of mutex locks
  - e.g. available, held by a thread, ...
- ...

...
Code template

- Code template is available on the project web page.
- You can modify everywhere in chess.cpp
  - Please ignore // TODO comment.
Code template

- chess.cpp re-defines pthread functions.
  - When a test program call pthread functions it will call the function in chess.cpp instead of the original pthread functions.
  - If you need to use original pthread functions inside chess.cpp, use original_pthread_xyz() instead.
    - e.g. original_pthread_mutex_lock() instead of pthread_mutex_lock(), original_pthread_mutex_unlock() instead of pthread_mutex_unlock(), ...
Questions?

If you have more questions while doing projects, use piazza

http://www.piazza.com/purdue/fall2014/cs353