Announcements

• Homework #5 was posted
Pthread Programming

Useful links:

- https://computing.llnl.gov/tutorials/pthreads/
- http://www.cs.cf.ac.uk/Dave/C/node31.html
Example code

every code is posted on course website.
Simple Pthread Example

See pthread_simple.c

- Harcodes 5 threads
- Do they run in any specific order?
Simple Init Example

See pthread_init.c.

- Initializes 256MB of data. Number of threads from command line.
  Is this the most efficient way to init memory?
- Why do we have the sleep call? Note: you’d never want to write a real program using a sleep like that.
- Why errors if run on odd number?
  Be sure when splitting up problem handle remainders.
Simple Join Example

Can use join to make the master thread wait for the others to finish.

See pthread_join.c
Stack Example

How to see how much stack is available, and how to change it if not enough.

See pthread_stack.c
Mutex Example

See pthread_mutex.c for code w/o mutex (run with a num greater than 1)
Then see pthread_mutex2.c for core w mutex

Creates a “thread pool” and the threads can request more work when they finish.
Mutex Info

- Can create mutexes two ways,
  - Statically, when declared
    
    ```c
    pthread_mutex_t our_mutex = PTHREAD_MUTEX_INITIALIZER;
    ```
  - Dynamically with `pthread_mutex_init()` which allows setting mutex object attributes, `attr`.
- The mutex is initially unlocked.
- Can specify protocol, priority ceiling, and if it’s shared/private.
- `lock`, `unlock`, `trylock`. Lock will spin until available,
trylock is non-blocking.
Deadlock

When you have more than one lock, it is possible to end up nesting locks in ways that lockup a program with both threads getting stuck.

<table>
<thead>
<tr>
<th>Thread 1</th>
<th>Thread 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_mutex_lock(&amp;mutex1);</td>
<td>pthread_mutex_lock(&amp;mutex2);</td>
</tr>
<tr>
<td>pthread_mutex_lock(&amp;mutex2);</td>
<td>pthread_mutex_lock(&amp;mutex1);</td>
</tr>
</tbody>
</table>
Condition Variable Example?

See `pthread_mutex.c`

- Can have a thread start up sleeping on a lock, and wake up when signalled by another thread.
PAPI Example

See pthread_papi.c

• Initialize with:
  PAPI_library_init(PAPI_VER_CURRENT);

• You can/should check all functions to see if return PAPI_OK

• If using pthreads need to do:
  PAPI_thread_init(pthread_thread_self);
• Eventsets are just integers
  int eventset=PAPI_NULL;

• Gathered results are typically 64-bit integers
  long long values[1];

• Create an eventset:
  PAPI_create_eventset(&eventset);

• Add an event. Available events can be seen with the
  papi_avail and papi_native_avail commands.

• PAPI_add_named_event(eventset,"PAPI_TOT_INS");
• Before the code of interest do a
  \texttt{PAPI\_start(eventset);} \\
• Afterward do a
  \texttt{PAPI\_stop(eventset,values);} \\
  and you can print the value or save it for later.