Project 1
Soumya Basu
Department of Computer Science
Cornell University
February 5, 2016
• Project 1 will be released tomorrow
• Due on February 18th at noon
• We’re using Github!
• Read the assignment before lecture
Github

• CMS quiz due tonight at midnight!

• Instructions:
  • Make an account on Github
  • Find a partner
  • Type in your and your partner’s Github usernames
Purpose

• Gain experience in C

• Learn how threading and scheduling works

• Implement basic synchronization primitives

• Large project!

  • Bad coding style will come back to bite later…
Project Overview

- Sample Application
- Minithreads
- Scheduler
- Semaphores
- FIFO Queues
Project Overview

- Sample Application
- Minithreads
- Scheduler
- Semaphores
- FIFO Queues
Queues

• Simple FIFO queue
• Interface described in queue.c
• Use a linked list queue
• queue_prepend, queue_append, queue_dequeue should be FAST
  • constant time
Making Queues Faster

struct queue_node {
    queue_node* next;
};

struct minithread {
    minithread* next;
    . . . <other things>
};

DON’T DO THIS!!
Project Overview

Sample Application

Minithreads

Scheduler

Semaphores

FIFO Queues
Concurrency 101

• You decide how many threads can hold a semaphore

• Semaphore value is manipulated atomically
  • `semaphore_P` decrements the value by 1
  • `semaphore_V` increments the value by 1

• Threads wait on a semaphore
  • if count is 0, `semaphore_P` blocks
  • if count is 0, `semaphore_V` wakes up waiting threads
Project Overview

Sample Application

Minithreads

Scheduler

Semaphores

FIFO Queues
Scheduler

- First come first serve
  - Just take the next thread off of the FIFO queue
- This will get a lot more complicated in P2!
  - **Code style is important!!!!**
Minithreads

• This is the “meat” of this project

• Need a Thread Control Block for each thread

• TCB contains…
  • Stack top pointer
  • Stack base pointer
  • Thread identifier
  • Anything else you need for bookkeeping!!
Minithreads

• Lots of useful functions given… but make sure you understand them!

• Allocating the stack:
  • Use `minithread Allocate_stack`

• Initializing the stack:
  • Use `minithread initialize_stack`

• Context switching between threads:
  • Use `minithread switch`
void minithread_system_initialize

- This bootstraps the system
- Initialize global data structures (queues, global semaphores, etc)
Bootstrapping

• What happens when there’s no user threads?
  • System shouldn’t crash! It’s an operating system
  • Run the **idle thread**
  • In our case, the kernel thread is the idle thread
    • No need to allocate the stack
    • TCB’s stacktop and stackbase should be NULL
More Scheduling!

- What happens when a user thread runs forever?
  - In P1, we let it!
  - Assume that all threads will voluntarily yield
    - Call `minithread_yield`
Testing

• We supply some *simple* tests
  • Useful to figure out how minithreads work

• Your code will not work the first time!
  • If it does, you won the CS lottery
  • Don’t let us find bugs while grading!
Coding Style

• Avoid unnecessary polling

```python
while (condition == False) {
    minithread_yield();
}
```

Moral: Unnecessary context switches are bad!
• Comments are good

```python
# Polling because I want slow code
while (condition == False) {
    minithread_yield();
}
```
...but you can have too much of a good thing.

```c
int x = 0;
# X is probably 1 at this point...
x = x + 1;
# So... I think I want to say y = 2, but
# I'm not sure if computers can add
# properly.
y = 2*x;
# Checking for cosmic radiation
# Because this actually is not an
# invariant
assert (y == 2);
```
Coding Style

• Names are a little important.

# BAD: zargle is a made up, meaningless word
int zargle = 5;

# num_threads is concise and explicit
int NUM_THREADS = 5;
# ALL CAPS MEANS THAT THE VALUE IS CONSTANT

...and yes, the commenting is bad here too.
Coding Style

• Not really a style thing, but here's a helpful pointer

```
int *int_ptr = (int*) malloc(sizeof(int));
int_ptr = 5;
```

What does this do?
Acknowledgements

• Slides adapted from previous TA: Kai Mast

• …who adapted it from many other TAs

• Sean Ogden, Robert Escriva, Z. Teo, Ayush Dubey, …
Questions?