

CSc 360

Operating Systems

Pthread

Jianping Pan
Summer 2006

5/29/06

CSc 360

1

Review: threads

- Threads
 - a basic unit of CPU utilization
 - thread vs process
- User vs kernel-level threads
 - thread model
- Issues with threading
 - fork(), exec()
 - kill()

5/29/06

CSc 360

2

Pthread library

- Create a thread
 - `int pthread_create (thread, attributes, start_routine, arguments);`
 - PC: `start_routine(arguments);`
 - default attributes: joinable and non-realtime
- Exit from a (created) thread
 - `void pthread_exit (return_value);`
 - cleanup handlers by `pthread_cleanup_push ();`
 - stack-like “reverse” execution order

5/29/06

CSc 360

3

Pthread library: more

- Wait a target thread to exit: *synchronize*
 - `int pthread_join (thread, return_value);`
 - release resource allocated to the target thread
- Put a target thread in detached state
 - `int pthread_detach (thread);`
 - no other threads can “join” this one
 - no “`pthread_attach`”
 - resource released once the thread exits
 - thread can be created in detached state

5/29/06

CSc 360

4

Pthread: further more

- Cancel another thread
 - `int pthread_cancel (thread);`
 - calling thread: send a request
 - target thread: `pthread_setcancelstate();`
 - ignore the request
 - terminate immediately
 - asynchronous cancellation
 - check whether it should be cancelled periodically
 - deferred cancellation

5/29/06

CSc 360

5

Example: producer-consumer

- Multi-process
 - shared memory solution
 - message passing solution
- Single-process, multi-thread

```
#include <pthread.h>
...

void *producer (void *args);
void *consumer (void *args);

typedef struct {...} queue;
```

5/29/06

CSc 360

6

Main thread

```
queue *queueInit (void);
void queueDelete (queue *q);
void queueAdd (queue *q, int in);
void queueDel (queue *q, int *out);

int main ()
{
    queue *fifo;
    pthread_t pro, con;

    fifo = queueInit ();
    if (fifo == NULL) {
        fprintf (stderr, "main: Queue Init failed.\n");
        exit (1);
    }
    pthread_create (&pro, NULL, producer, fifo);
    pthread_create (&con, NULL, consumer, fifo);
    pthread_join (pro, NULL);
    pthread_join (con, NULL);
    queueDelete (fifo);

    return 0;
}
```

5/29/06

CSc 360

7

Producer thread

```
void *producer (void *q)
{
    queue *fifo;
    int i;

    fifo = (queue *)q;

    for (i = 0; i < LOOP; i++) {
        /* produce LOOP items, inserting them into
         * the "fifo" queue.
         */
        ...
    }
    return (NULL);
}
```

5/29/06

CSc 360

8

Consumer thread

```
void *consumer (void *q)
{
    queue *fifo;
    int i, d;

    fifo = (queue *)q;

    for (i = 0; i < LOOP; i++) {
        /* Consumer LOOP items from the
         * "fifo" queue.
         */
        ...
    }
    return (NULL);
}
```

5/29/06

CSc 360

9

This lecture

- Pthread library
 - create and terminate threads
 - passing arguments: start_routine (arguments);
 - join and detach threads
 - synchronize
- Explore further
 - Pthread tutorial
<http://www.llnl.gov/computing/tutorials/pthreads/>
 - Assignment 2!

5/29/06

CSc 360

10

Next lecture

- Pthread
 - threads: sharing data in a process
 - read-write, write-write conflicts
 - mutex and condition variables
 - <http://www.llnl.gov/computing/tutorials/pthreads/>