COS 431
Operating Systems

 pthreads

Department of Computer Science
University of Maine
Why Threads?

- Sometimes need to do multiple things at a time within a program
- Examples:
  - Web browser
  - Web server
  - Database server
- Could use multiple processes; e.g.:
  - Web browser forks processes to fetch images, etc.
  - Web (or DB) server forks processes for each request
- Problems:
  - Overhead of process start-up
  - Can’t share memory (e.g., window for Web browser)
Threads

- Threads offer a light-weight means to have multiple things done at once
- Unlike processes, which run separate programs, think of threads as running separate *functions* within the same program.
- Easy/quick to start
- Share memory (e.g., global variables)
- Supported in Unix by pthread library
pthreads

- Header file:
  
  ```c
  #include <pthread.h>
  ```

- Type:
  
  ```c
  pthread_t myThread;
  ```

- Creating a thread:
  
  ```c
  pthread_create(&myThread, NULL, myFunction, (void *) arg);
  ```

- Waiting for a thread:
  
  ```c
  pthread_join(&myThread, &retval); // or NULL for retval
  ```
Example

// From [Molay, 2003]
#include <stdio.h>
#include <pthread.h>
#define NUM 5
main() {
    pthread_t t1, t2;
    void *print_msg(void *
    pthread_create(&t1,NULL,print_msg,(void *)"hello ");
    pthread_create(&t2,NULL,print_msg,(void *)"world\n");
    pthread_join(t1,NULL);
    pthread_join(t2,NULL);}

void *print_msg(void *m) {
    char *cp = (char *) m;
    int i;
    for (i=1;i<NUM;i++) {
        printf("%s",m);
        fflush(stdout);
        sleep(1);
    }
    return NULL;}
Problem: Thread Synchronization

- Could use semaphores, but...
- ...what would happen if a thread did a down on a semaphore and blocked?
- Instead, use built-in support for *mutex locks*. 


#include <pthreads.h>
...

pthread_t t1, t2;

pthread_mutex_t lock;
...

pthread_create(&t1,NULL,function,(void *)args);
pthread_create(&t2,NULL,function,(void *)args);
...

//inside of function...

pthread_mutex_lock(&lock);
...critical region

pthread_mutex_unlock(&lock);
...