Processes and Threads

- Multiple threads can reside within a single process
Threads and Processes

- **Process**: Execution stream + Program State
- **Thread**: Execution stream + Thread State (in same process address space)

- **Thread State**: Bare minimum required to support a stream
  - Thread is also called a “lightweight process”
#include <stdio.h>
#include <pthread.h>

void *do_stuff(void *p) {
    printf("Hello from thread %d\n", *(int *)p);
}

int main() {
    pthread_t thread;
    int id, arg1, arg2;

    arg1 = 1;
    id = pthread_create(&thread, NULL, do_stuff, (void *)&arg1);
    pthread_join(thread, NULL);
    arg2 = 2;
    do_stuff((void *)&arg2);

    return 0;
}
Mutex

• MUTual EXclusion: a lock that only one thread can acquire

• To protect a critical section:
  1. Acquire lock at the beginning
     • All other threads attempting to enter are blocked
  2. Release lock at the end
     • Another thread can now enter after acquiring lock
Critical Sections

Shared Data:

<table>
<thead>
<tr>
<th>tail</th>
<th>5</th>
</tr>
</thead>
</table>

mutex

A[]

1 8 5 6 20 9

Enqueue():

lock(&mutex);
A[tail] = 20;
tail++;
unlock(&mutex);

Thread 0

Blocked!

lock(&mutex);
A[tail] = 9;
tail++;
unlock(&mutex);

Thread 1
#include <stdio.h>
#include <pthread.h>

int tail = 0;
int A[20];

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

void enqueue(int value)
{
    pthread_mutex_lock(&mutex);
    A[tail] = value;
    tail++;
    pthread_mutex_unlock(&mutex);
}