Pthreads
A quite short introduction (Part II)

Agenda
- Last Tutorial (Feb.09)
  - Short introduction to threads
  - Creation and termination
- Today (Feb.14)
  - Synchronization

Synchronization
- Ensures data consistency
- Methods
  - Join
    - Wait for other threads to complete
  - Mutual Exclusion Locks (Mutex)
    - Block access (only one thread at a time)
  - Conditional Variables
    - Block access until condition is true

Join
- Suspends execution of current thread until specified thread is completed
  - Example:
    ```c
    pthread_join(thread_id, NULL);
    ```

Mutual Exclusion Locks
- Prevent “race conditions”
- Control access to data
- “Normal” sequence
  1. Create and initialize mutex variables
  2. Several threads attempt to lock, just one succeed (owner)
  3. After performing tasks, owner unlocks mutex
  4. Back to (2) until all threads are executed
  5. Mutex is destroyed

Mutex Basic Operations
- Initialization
  - Static
    - `pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;`
  - Dynamic
    - `pthread_mutex_t *mutex;`
      - `n = (pthread_mutex_t *)malloc(sizeof(pthread_mutex_t));`
      - `pthread_mutex_init(n, NULL);`
- Destroy
  - `pthread_mutex_destroy(mutex);`
Mutex Basic Operations

- **Lock**
  - `pthread_mutex_lock(&mutex);`
  - Current thread waits until mutex is unlocked

- **Unlock**
  - `pthread_mutex_unlock(&mutex);`

- **“Safer” lock**
  - `pthread_mutex_trylock(&mutex);`
  - Returns an error if mutex is locked
  - Helps avoiding deadlock

Conditional Variables

- Synchronization based on data
- Alternative: threads should continuously checking for specific condition (which might also require access to “protected” variables)
- Used in conjunction with Mutex

CV Basic Operations

- **Initialization**
  - `int pthread_cond_init(pthread_cond_t *cond, pthread_condattr_t *cond_attr);`
  - `pthread_cond_t cond = PTHREAD_COND_INITIALIZER`

- **Wait for condition**
  - `pthread_cond_wait(&cond, &mutex);`

- **Send signal to one thread**
  - `pthread_cond_signal(&cond);`

- **Send broadcast**
  - `pthread_cond_broadcast(&cond);`

Example

```c
void *thread1(void *arg) {
  //...
  pthread_mutex_lock(&count_lock);
  while (count < MAX_COUNT) {
    pthread_cond_wait(&count_cond, &count_lock);
  }
  pthread_mutex_unlock(&count_lock);
  // ...
  pthread_exit(NULL);
}

void *thread2(void *arg) {
  //...
  pthread_mutex_lock(&count_lock);
  if (count == MAX_COUNT) {
    pthread_mutex_unlock(&count_lock);
    pthread_cond_signal(&count_cond);
  } else {
    pthread_mutex_unlock(&count_lock);
    // }
  // ...
  pthread_exit(NULL);
}
```

Exercise

- Copy source code and compile
  - `cp /home/grads/dmncastr/457/09/mutex.tar.gz`
  - `tar -zxvf mutex.tar.gz`
  - `cd cpc457-09`

- Review and compile “mutex.c”
  - Execute it a few times
  - Notice that counter might be duplicated and not always add up to 10 (never?).
- Fix it by adding a mutex called “lock”

mutex.c

```c
#include <pthread.h>
...
int global_counter = 0;
void *print_message(void *thread_id) {
  int tid = (int)thread_id;
  int counter = global_counter;
  counter += 1;
  printf("Thread %d starting... Counter=%d\n", tid, counter);
  counter +=1;
  printf("Thread %d finishing... Counter=%d\n", tid, counter );
  global_counter = counter;
  return NULL;
}
```