Recitation 11
11/19/2018
Today

- Mutex Locks
- Conditional Variables
Synchronization

A section of code which works with shared resources can have unpredictable results depending on the order in which the threads execute.

This is called a critical section.
Mutual Exclusion

A **mutex** lets you lock a code section so that only one thread at a time executes a critical section.

First, you have to identify your critical sections.

Example: `sum = sum + 1;` in `race.c`

Then create a mutex for each shared resource, and lock and unlock them before and after each critical section.
POSIX Mutex

Initialize

```c
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
```

Lock

```c
pthread_mutex_lock(&mutex);
```

Unlock

```c
pthread_mutex_unlock(&mutex);
```
Posix condition variables

**Initialize**

```c
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
```

**Signal**

```c
pthread_cond_signal(pthread_cond_t* cond);
```

**Wait**

```c
pthread_cond_wait(pthread_cond_t* cond,
                  pthread_mutex_t* mutex);
```

**Broadcast**

```c
pthread_broadcast(pthread_cond_t* cond);
```
Questions?