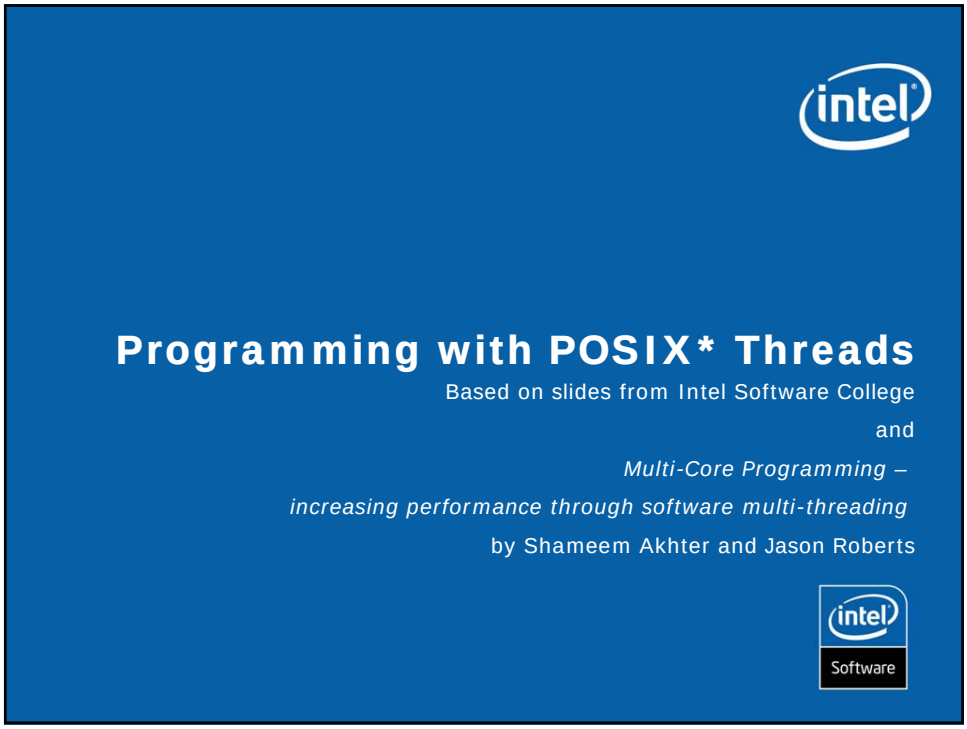

Objectives

Explore Pthreads “core” functions to create and synchronize threads

What is Pthreads?

POSIX.1c standard

C language interface

Threads exist within same process

All threads are peers

- No explicit parent-child model
- Exception: “main thread” holds process information

pthread_create

```
int pthread_create(tid, attr, function, arg);
```

pthread_t *tid

- handle of created thread

const pthread_attr_t *attr

- attributes of thread to be created

void *(*function)(void *)

- function to be mapped to thread

void *arg

- single argument to function

pthread_create Explained

Spawn a thread running the function

Thread handle returned via **pthread_t** structure

- Specify **NULL** to use default attributes

Single argument sent to function

- If no arguments to function, specify **NULL**

Check error codes!

EAGAIN - insufficient resources to create thread
EINVAL - invalid attribute

Example: Thread Creation

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

void *hello (void * arg) {
      printf("Hello Thread\n");
}

main() {
  pthread_t tid;

  pthread_create(&tid, NULL, hello, NULL);

}
```

What Happens?

Waiting for a Thread

```
int pthread_join(tid, val_ptr);
```

pthread_t tid

- handle of *joinable* thread

void **val_ptr

- exit value returned by joined thread

pthread_join Explained

Calling thread waits for thread with handle **tid** to terminate

- Only one thread can be joined
- Thread must be *joinable*

Exit value is returned from joined thread

- Type returned is **(void *)**
- Use **NULL** if no return value expected

ESRCH - thread (pthread_t) not found
EINVAL - thread (pthread_t) not joinable

Thread States

Pthreads threads have two states

- *joinable* and *detached*

Threads are joinable by default

- Resources are kept until **pthread_join**
- Can be reset with attributes or API call

Detached threads cannot be joined

- Resources can be reclaimed at termination
- Cannot reset to be *joinable*

Example: Multiple Threads

```
#include <stdio.h>
#include <pthread.h>
#define NUM_THREADS 4

void *hello (void *arg) {
      printf("Hello Thread\n");
}

main() {
  pthread_t tid[NUM_THREADS];
  for (int i = 0; i < NUM_THREADS; i++)
    pthread_create(&tid[i], NULL, hello, NULL);

  for (int i = 0; i < NUM_THREADS; i++)
    pthread_join(tid[i], NULL);
}
```

What's Wrong?

What is printed for myNum?

```
void *threadFunc(void *pArg) {
  int* p = (int*)pArg;
  int myNum = *p;
  printf( "Thread number %d\n", myNum);
}
. . .
// from main():
for (int i = 0; i < numThreads; i++) {
   pthread_create(&tid[i], NULL, threadFunc, &i);
}
```


Solution – "Local" Storage

```
void *threadFunc(void *pArg)
{
  int myNum = *((int*)pArg);
  printf( "Thread number %d\n", myNum);
}
. . .

// from main():
for (int i = 0; i < numThreads; i++) {
  tNum[i] = i;
  pthread_create(&tid[i], NULL, threadFunc, &tNum[i]);
}
```