

# Threads (3)

---

Dr. Jun Zheng

CSE325 Principles of Operating  
Systems

9/25/2019



# Thread Libraries

- ❑ **Thread library** provides programmer with API for creating and managing threads
- ❑ Two primary ways of implementing
  - ❑ Library entirely in user space
  - ❑ Kernel-level library supported by the OS
- ❑ Three main thread libraries in use today
  - ❑ POSIX threads (Pthreads)
  - ❑ Win32
  - ❑ Java

# Pthreads

- ❑ The standard interface for C threads
- ❑ <https://computing.llnl.gov/tutorials/pthreads/>
- ❑ To create threads
  - ❑ **int pthread\_create(pthread\_t \*tid, pthread\_attr\_t \*attr, void \*(\*func)(void \*), void \*arg);**
  - ❑ **tid** is a pointer to an allocated (dynamic or otherwise) **pthread\_t** (opaque type) that will have the thread ID of the new thread placed in it
  - ❑ **attr** is a pointer that can be used to change the attributes of the new thread (but we'll usually just use **NULL**)
  - ❑ **func** is a function pointer to the new thread's function to execute
  - ❑ **arg** is a pointer that will be passed to the new thread's routine when the thread is created; this is the way you pass arguments to a thread
  - ❑ returns **0** on success, **nonzero** on error
  - ❑ we can use structures to pass multiple values to the function executed by the thread

# Thread Termination

- ❑ Threads can be terminated in one of four ways:
  - ❑ **Implicit termination**: thread routine returns; usually what we'll use
  - ❑ **Explicit termination**: the thread calls **pthread\_exit()**
  - ❑ **Process exit**: any thread calls **exit()**, which terminates the process and all associated threads; maybe not what you really want
  - ❑ **Thread cancellation**: another thread calls **pthread\_cancel()** to terminate a specific thread

# Thread Reaping

- ❑ When a thread has terminated, information about it, including the thread return value, is still kept in memory until reaped by another thread
- ❑ Threads are reaped by **pthread\_join()**:
  - ❑ **int pthread\_join(pthread\_t tid, void \*\*val);**
  - ❑ Reaps thread with thread ID **tid**
  - ❑ Blocks until thread **tid** terminates
  - ❑ Frees memory resources held by thread **tid**
  - ❑ Returns 0 on success, nonzero on error
  - ❑ On success, **\*val** is the return value of the terminated thread

# Compiling Pthreads Code

- ❑ To compile the example programs, you need to tell **gcc** to use the pthread library when linking your executable
- ❑ To do this, use the **-l** switch to **gcc**:
  - ❑ **gcc -o threadex threadex.c -lpthread**
  - ❑ The **-l** switch is needed to use many other libraries
    - ❑ math functions, for example
  - ❑ If you forget this, your program will not compile, and you will get an error like this:
    - ❑ **/tmp/cc3VuzAe.o(.text+0x3a): In function `main':  
: undefined reference to `pthread\_create'**
  - ❑ You can add this flag to the **LDFLAGS** variable in your **Makefile** to have it work correctly
- ❑ All of the functions we'll talk about that begin with "pthread\_" require including **<pthread.h>**
- ❑ **Example:** **threads\_basics.c, threads\_ret\_sqr\_value.c**

# Several Threads

**Example:** `threads_several.c`

- ❑ Why do we create an array called **ids**? Why not just pass in **&i** as the argument?
  - ❑ The value of **i** changes; if it changes before the new thread can access the memory, it's a problem.
  - ❑ What will the output be? Do we know what the first line to be printed out will be? How about the last?
  - ❑ Most of the output will be interleaved

**Example:** `threads_no_sync.c`