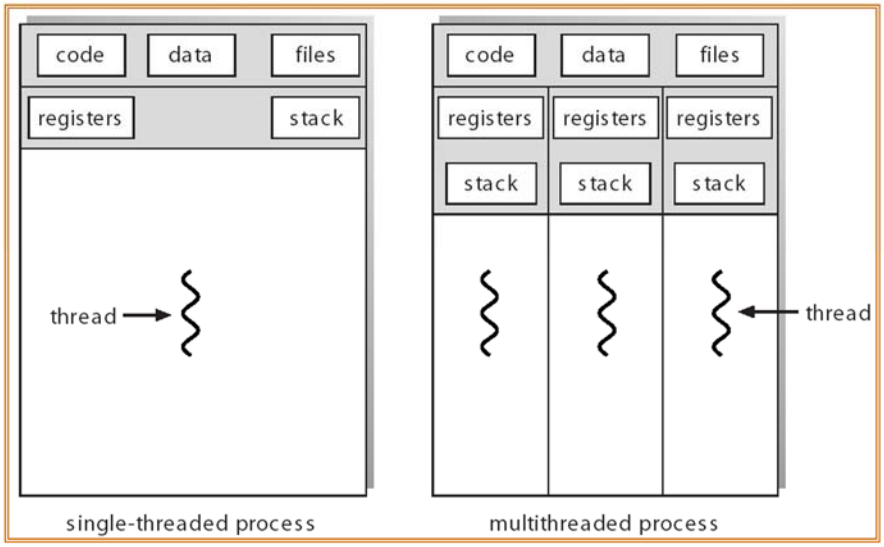
BUCKNELL UNIVERSITY
Computer Science

CSCI 315 Operating Systems Design

Threads

Notice: The slides for this lecture have been largely based on those accompanying an earlier edition of the course text *Operating Systems Concepts with Java*, by Silberschatz, Galvin, and Gagne. Many, if not all, the illustrations contained in this presentation come from this source.

Multithreading



Benefits

- Responsiveness
- Resource Sharing
- Economy
- Utilization of MP Architectures

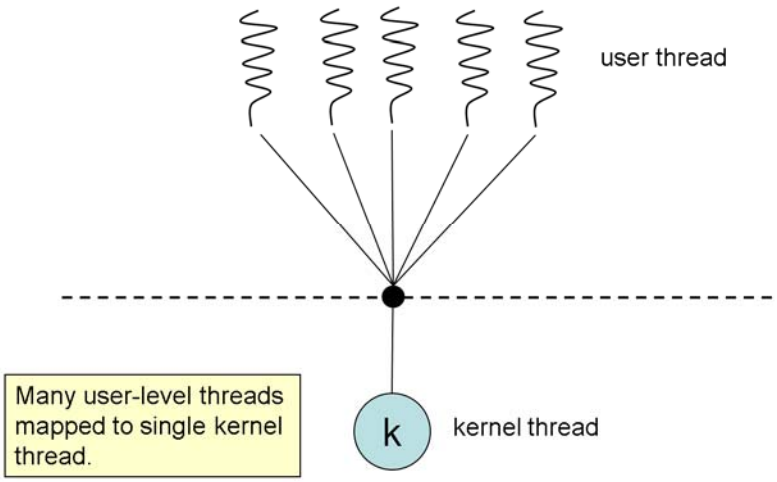
The 2 Types of Threads

- User Threads:
  - Thread management done by user-level threads library.
  - Three primary thread libraries:
    - POSIX Pthreads
    - Java threads
    - Win32 threads
- Kernel Threads:
  - Thread management done by the kernel.

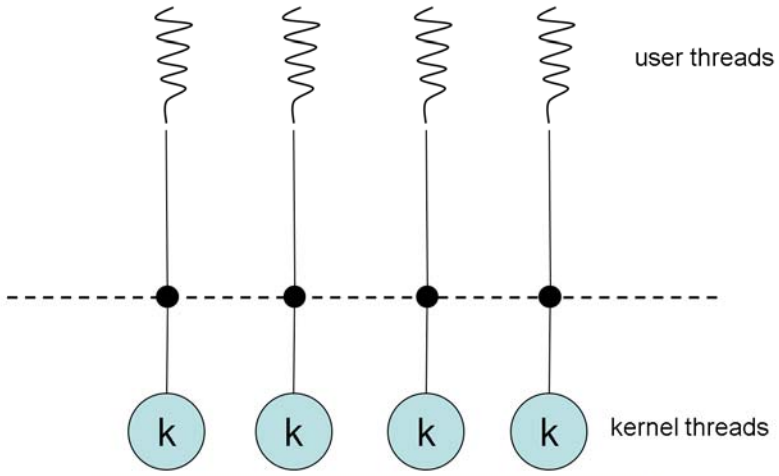
Multithreading Models

- Many-to-One
- One-to-One
- Many-to-Many

Many-to-One Model



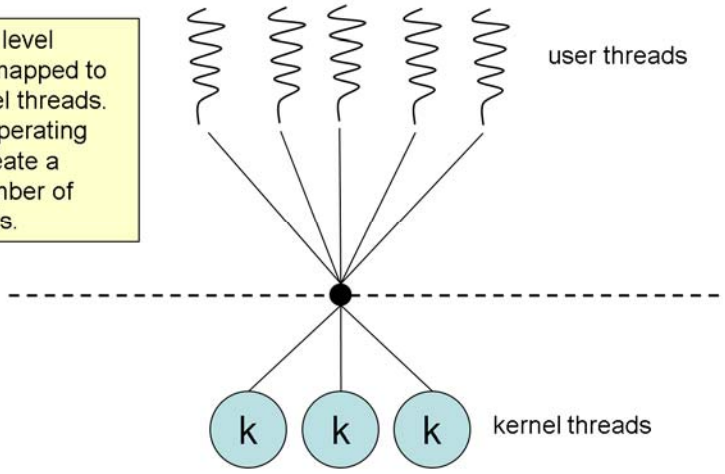
One-to-One Model



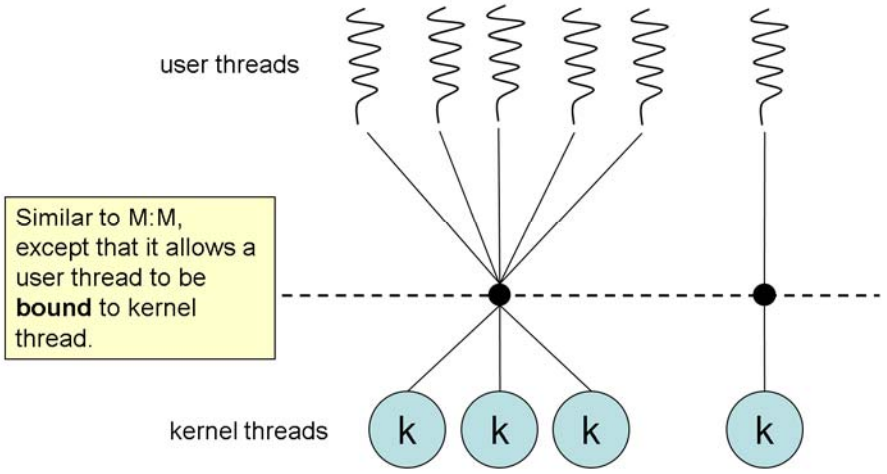
Each user-level thread maps to kernel thread.

Many-to-Many Model

Several user level threads are mapped to several kernel threads. Allows the operating system to create a sufficient number of kernel threads.



Two-Level Model



Threading Issues

- Semantics of **fork()** and **exec()** system calls (does **fork()** duplicate only the calling thread or all threads?)
- Thread cancellation
- Signal handling
- Thread pools
- Thread specific data
- Scheduler activations

Thread Cancellation

- Terminating a thread before it has finished.
- Two general approaches:
  - **Asynchronous cancellation** terminates the target thread immediately.
  - **Deferred cancellation** allows the target thread to periodically check if it should be cancelled.

Signal Handling

- Signals are used in UNIX systems to notify a process that a particular event has occurred.
- A **signal handler** is used to process signals:
  1. Signal is generated by particular event.
  2. Signal is delivered to a process.
  3. Signal is handled.
- Options:
  - Deliver the signal to the thread to which the signal applies.
  - Deliver the signal to every thread in the process.
  - Deliver the signal to certain threads in the process.
  - Assign a specific thread to receive all signals for the process.

Thread Pools

- Create a number of threads in a pool where they await work.
- Advantages:
  - Usually slightly faster to service a request with an existing thread than create a new thread.
  - Allows the number of threads in the application(s) to be bound to the size of the pool.

Thread Specific Data

- Allows each thread to have its own copy of data.
- Useful when you do not have control over the thread creation process (i.e., when using a thread pool).

Scheduler Activations

- Both M:M and Two-level models require communication to maintain the appropriate number of kernel threads allocated to the application.
- Scheduler activations provide **upcalls** - a communication mechanism from the kernel to the thread library.
- This communication allows an application to maintain the correct number kernel threads.

Pthreads

- A POSIX standard (IEEE 1003.1c) API for thread creation and synchronization.
- API specifies behavior of the thread library, implementation is up to development of the library.
- Common in UNIX operating systems (Solaris, Linux, Mac OS X).

Pthreads

```
int sum; /* this data is shared by the thread(s) */
void *runner(void *param); /* the thread */

main(int argc, char *argv[]) {
  pthread_t tid; /* the thread identifier */
  pthread_attr_t attr; /* set of attributes for the thread */
  /* get the default attributes */
  pthread_attr_init(&attr);
  /* create the thread */
  pthread_create(&tid,&attr,runner,argv[1]);
  /* now wait for the thread to exit */
  pthread_join(tid,NULL);
  printf("sum = %d\n",sum); }

void *runner(void *param) {
  int upper = atoi(param);
  int i;
  sum = 0;
  if (upper > 0) {
    for (i = 1; i <= upper; i++)
      sum += i;
  }
  pthread_exit(0);}
```

Linux Threads

- Linux refers to them as *tasks* rather than *threads.*
- Thread creation is done through **clone()** system call.
- **clone()** allows a child task to share the address space of the parent task (process).