Processes and Threads

CSCI 3431: Operating Systems

Agenda

- Assignment 1 due September 29!
  - Linux/Unix environment available on the CS machine
  - ssh cs.smu.ca
- Today's lecture
  - Thread Model
  - Java thread libraries
  - Issues related to multithreaded programming
- Reading: Sections 4.1-4.5
- Next :Java thread libraries, synchronization

Just checking….

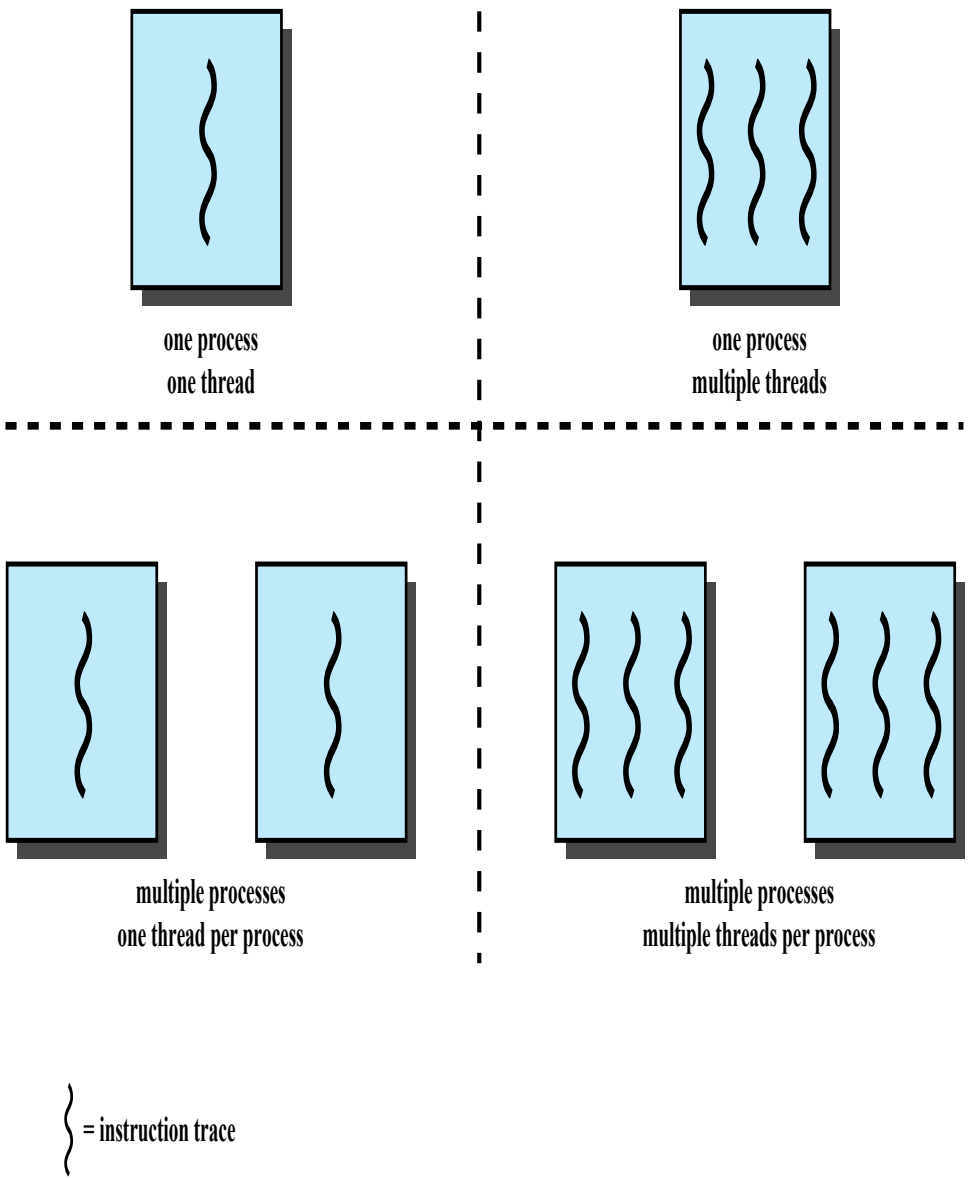
- What is a process?
- What information does the OS need to manage a process?
- What data structures are used?
- What is the difference between I/O-bound and CPU-bound processes?
- Why do we care?

Motivation

- How do we handle complex applications?

Processes and Threads

- Single Threaded Approaches:
  - A single thread of execution per process, in which the concept of a thread is not recognized, is referred to as a single-threaded approach
  - MS-DOS is an example
- Multithreaded Approaches:
  - One process with multiple threads (JRE)
  - Multiple processes , each supporting multiple threads (Windows, Solaris, modern versions of UNIX)



Processes and Thread

Resource Ownership

- **Process image**: collection of program, data, stack, and attributes defined in the process control block.
- The OS performs a protection function to prevent unwanted interference between processes with respect to resources.

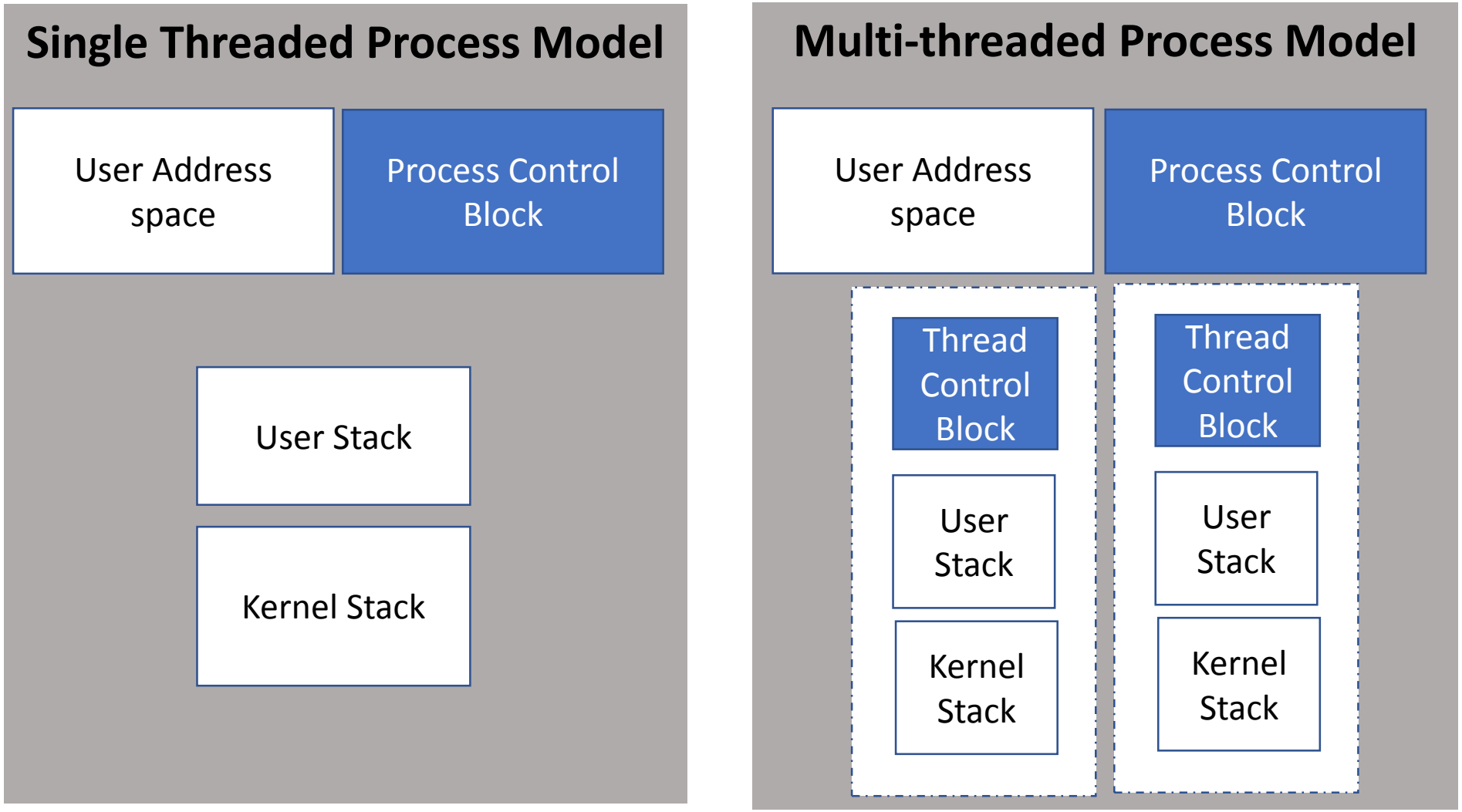
Scheduling/Execution

- Execution state (Running, Ready, etc.)
- The OS takes care of dispatching and scheduling a process
  - Priority
  - Execution may be interleaved with that of other processes

Processes and Threads

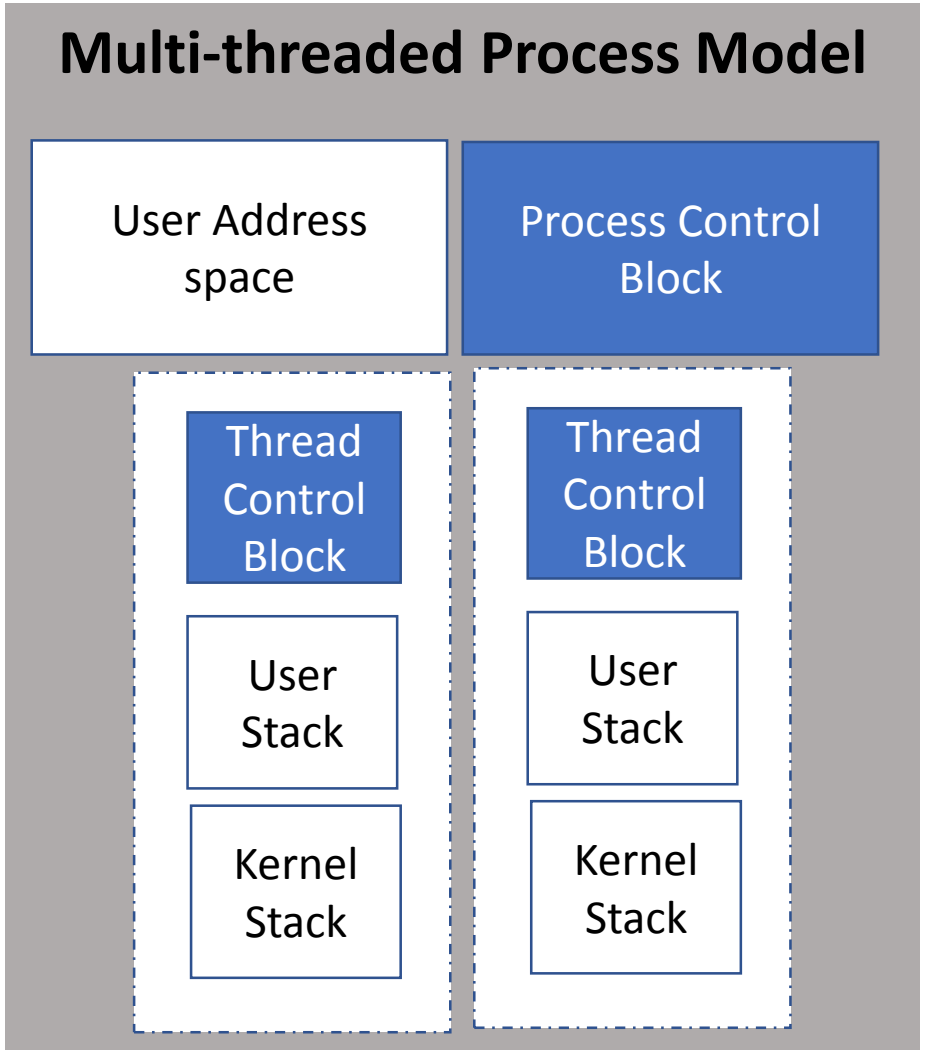
- **Process** or **task** refers to the unit of resource ownership
- **Thread** or **lightweight process** refers to the unit of dispatching
- **Multithreading** refers to the ability of an OS to support multiple, concurrent execution within a single process

Single and Multi-threaded Processes



One or More Threads in a Process

- Each thread has:
  - Access to the memory and resources of its process, shared with other threads in that process
  - CPU context
    - CPU registers
    - PC
  - Its own execution **stack**
    - Local variables
  - Thread Control Block (TCB)
    - Thread context (CPU context)
    - State (Running, Ready, etc.)



Key Benefits of Threads

- Responsiveness
- Resource Sharing
- Economy
- Scalability

Key Benefits of Threads

- Responsiveness
  - Loading webpage content made faster using threads
- Discuss the performance of Single Thread Web Browser with a colleague.
  - You click on a link
  - Decide to click on the stop button before you received a response

Key Benefits of Threads

- Resource Sharing
  - Process state and resources are shared among process threads
  - Same address space with access to same data
  - E.g., If one thread opens a file with read privileges, other threads in the same process can also read from that file
  - Enhanced communication between programs
- Discuss the implementation of a word processor
  - Handling Input
  - Formatting and displaying content
  - Backup

Key Benefits of Threads

- Economy
  - It takes less time to create/terminate a thread than a process (10x-100x faster)
  - The memory is already allocated for the process
  - No data or code needs to be copies
  - Thread creation is cheap
- It takes less time to switch between two threads within the same process than to switch between processes.

Key Benefits of Threads

- Scalability
  - Threads are already designed to run concurrently
  - Threads can be scheduled to run on multiple CPUs
  - The OS is responsible for scheduling threads
  - The application cannot tell if it is running on a single or multiprocessor system
  - Work can be distributed among multiple processors (when available)
- Increased parallelism on multiprocessor machines

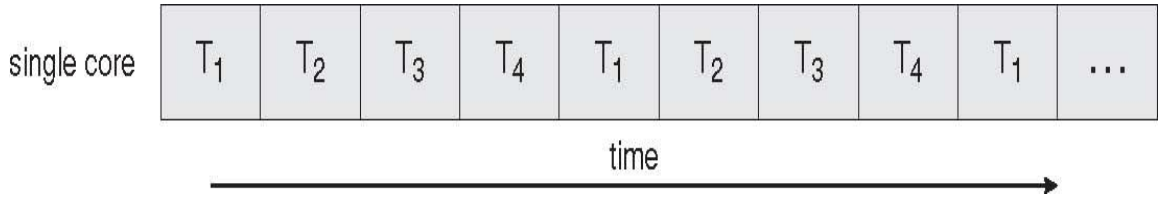
Concurrent Processes

- Most OS provide process and thread abstractions to allow creation of multithread and multi-process application
- Hence, we talk about **concurrent processes**

Concurrent Processes

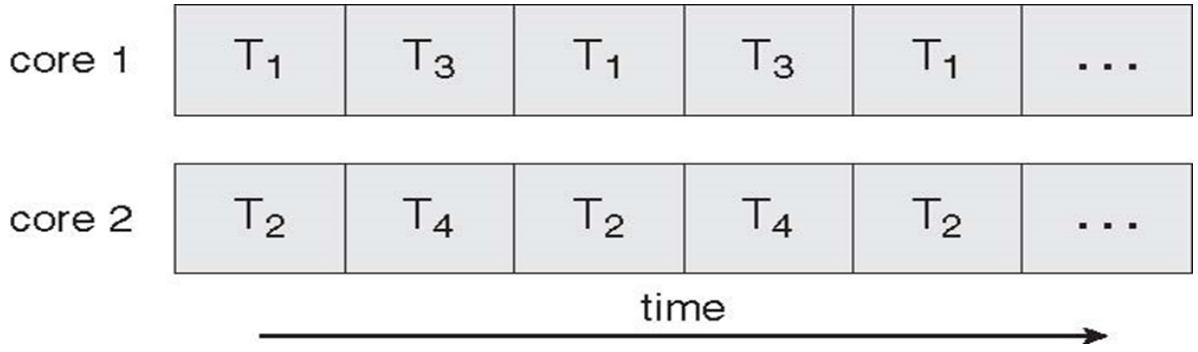
Concurrent Execution

- 2 or more tasks **seem** to be performed simultaneously
- Implemented in most modern PCs
- How does it work?
- Recall: Scheduling based on Time-slicing and pre-emption



Parallel Execution

- 2 or more tasks **are** performed simultaneously
- Requires cooperation from hardware
  - Multiple CPUs
  - CPU with multicores
  - Multithreaded CPU



Threads

User Threads

- Thread management done by user-level threads library
- Three primary thread libraries:
  - POSIX **Pthreads**
  - Win32 threads
  - Java threads

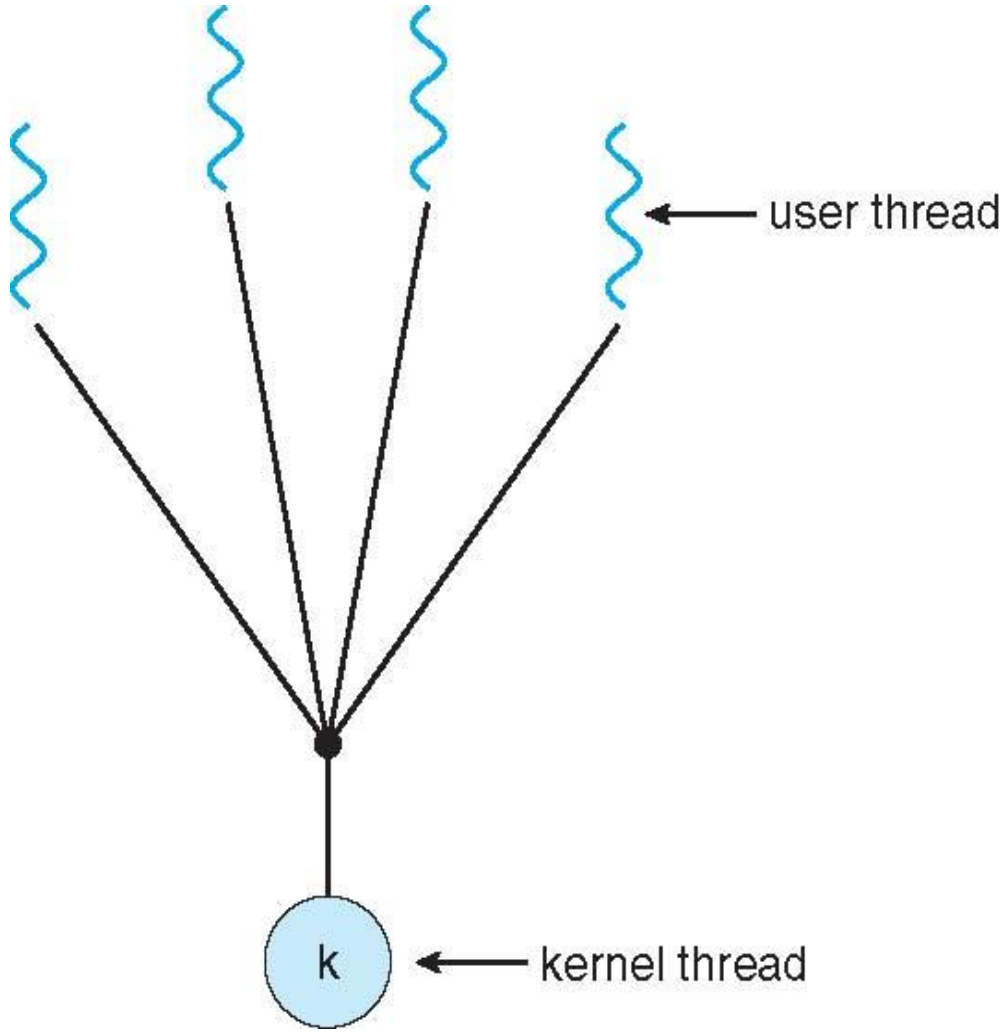
Kernel Threads

- Supported by the Kernel
- Examples
  - Windows XP/2000
  - Solaris
  - Linux
  - Tru64 UNIX
  - Mac OS X

Question: What kind of relationship exists between User Threads and Kernel Threads?

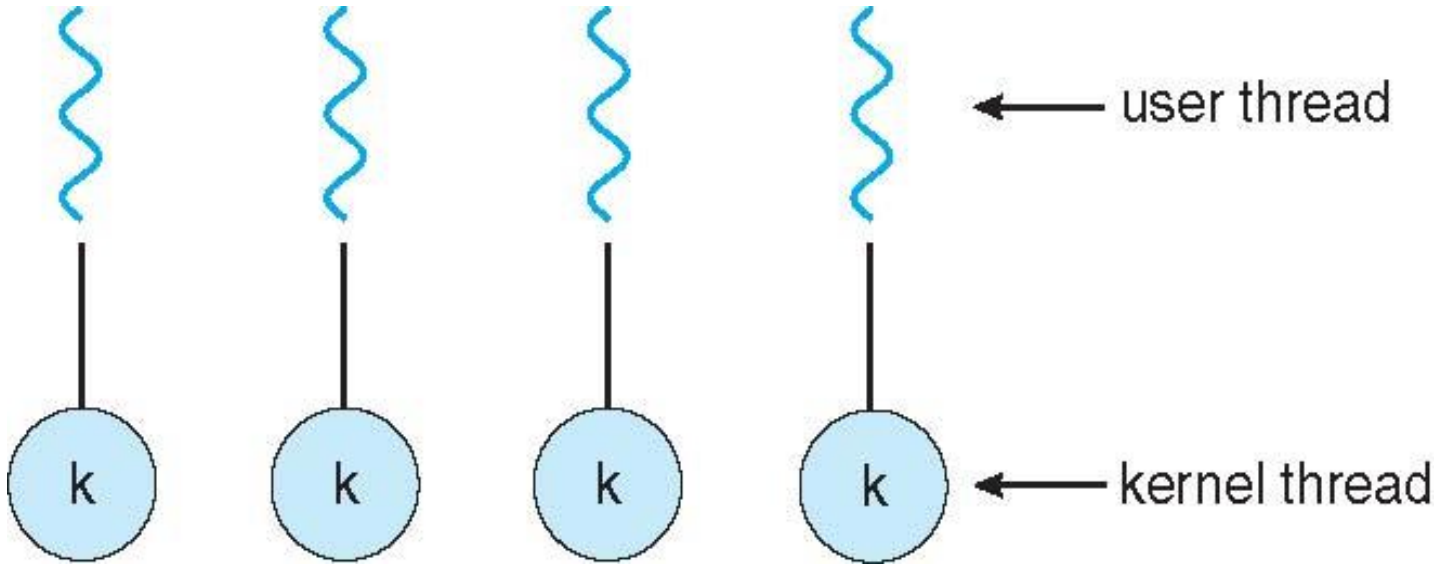
Many-to-One Model

- **M:1 Model - Many** user threads mapped to a kernel thread
- Thread management done by thread library in user space
  - As many user threads can be created
- Only one thread can access the kernel
  - No support for parallel execution on multiprocessors
  - Process is blocked when one thread makes a system call



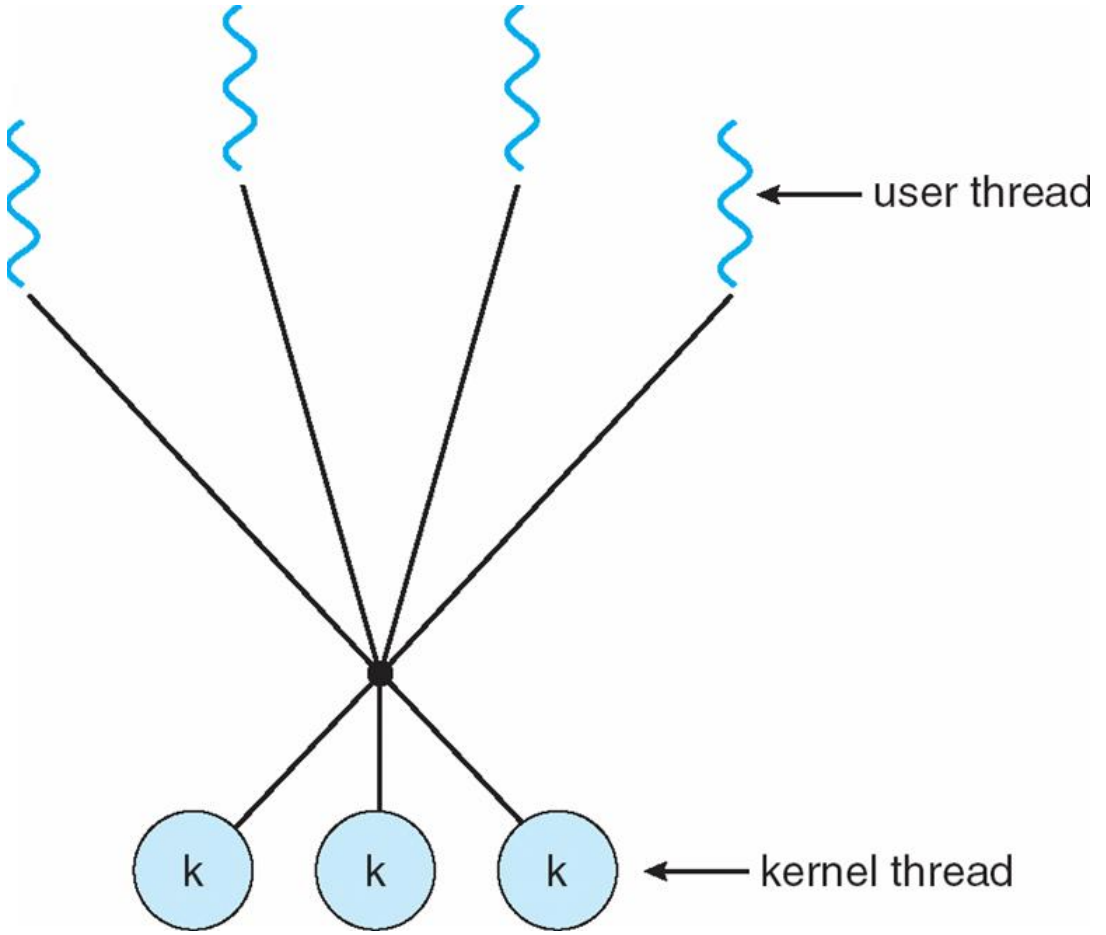
One-to-One Model

- **1:1-Model - One** user thread mapped to **one** kernel thread
- Pros: better support for concurrency
  - Process is NOT blocked when one thread makes a system call
  - Multiple threads run in parallel on multiprocessor
- Cons: requires a kernel thread for each new user thread
  - Degrade performance
  - # user threads can be limited



Many-to-Many Model

- **M:M – Model - Many** user threads mapped to **many** kernel thread
  - # kernel threads <= # user threads
  - No limit on the # threads
- Support for concurrency
- *Two-level* model supports mapping one user thread to one kernel thread



Multicore Programming

- Multicore systems putting pressure on programmers, challenges include:
  - **Dividing activities**
    - **Separate tasks**
  - **Balance**
    - **Tasks have equal work**
  - **Data splitting**
    - **Separate data**
  - **Data dependency**
    - **Ensure synchronization**
  - **Testing and debugging**