OPERATING SYSTEMS

Basir University, 2020-2021

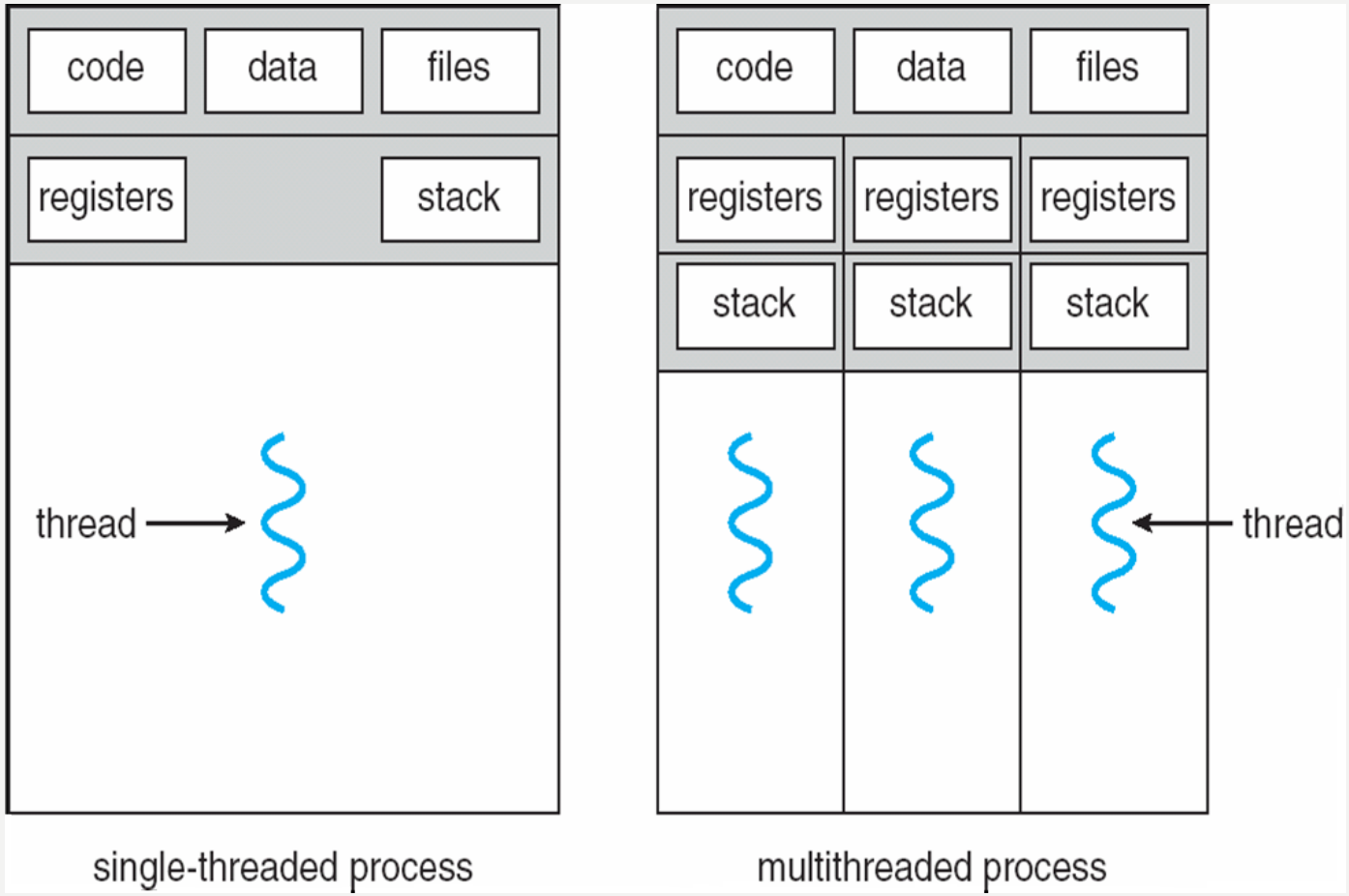
By: Prof. Dr. Mohammad Hajarian

- Session 4

THREADS

THREADS

SINGLE AND MULTITHREADED PROCESSES



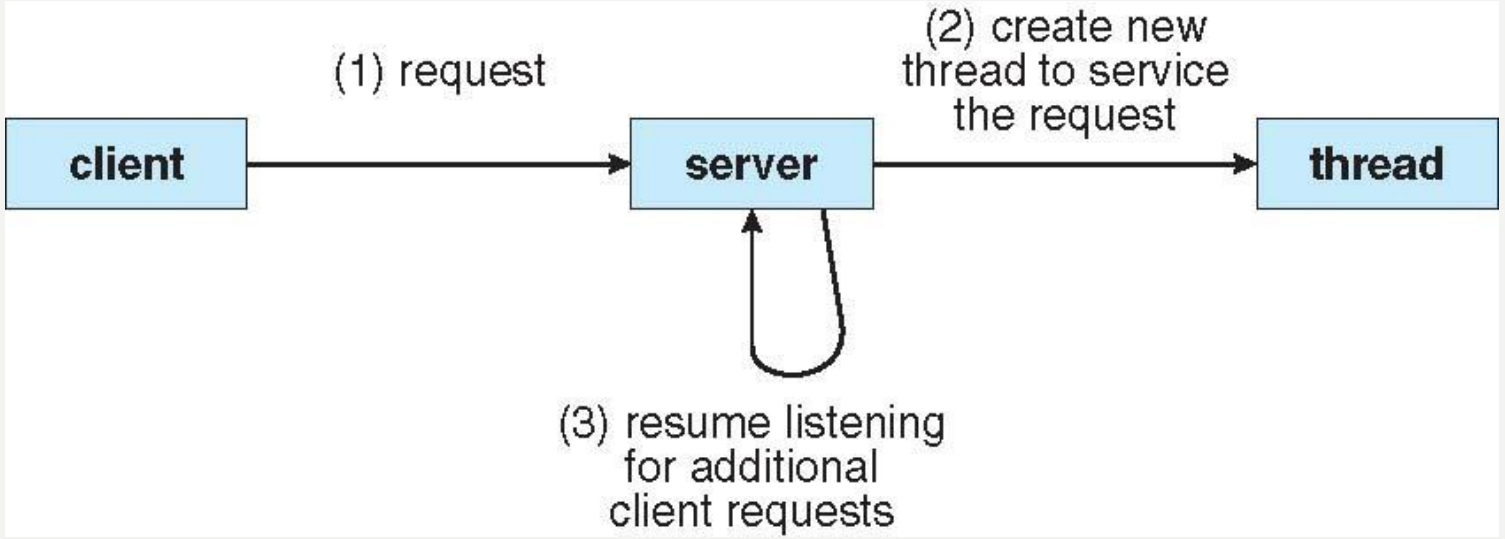
BENEFITS

- Responsiveness
- Resource Sharing
- Economy
- Scalability

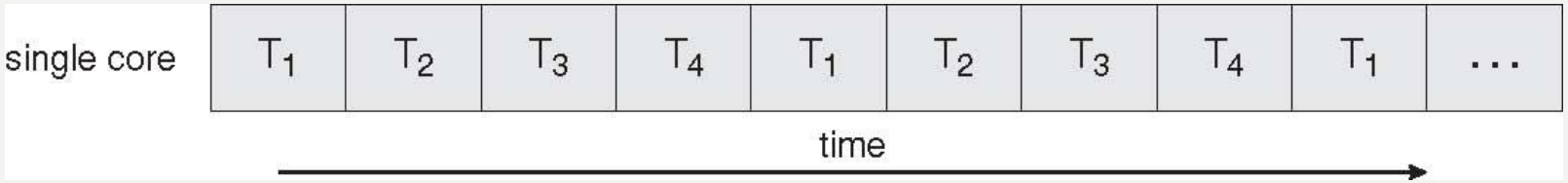
MULTICORE PROGRAMMING

- Multicore systems putting pressure on programmers, challenges include
  - **Dividing activities**
  - **Balance**
  - **Data splitting**
  - **Data dependency**
  - **Testing and debugging**

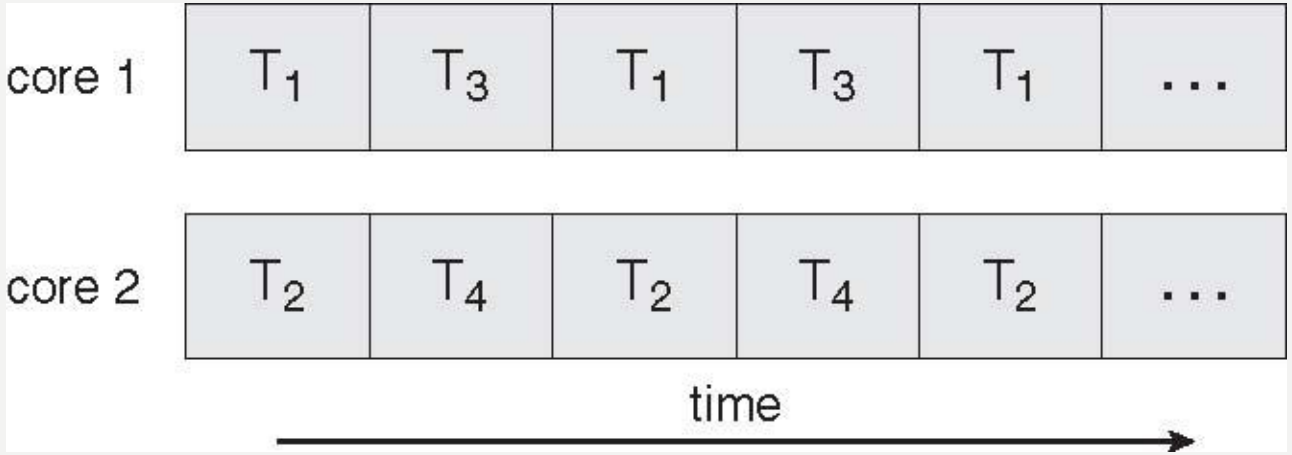
MULTITHREADED SERVER ARCHITECTURE



CONCURRENT EXECUTION ON A SINGLE-CORE SYSTEM



PARALLEL EXECUTION ON A MULTICORE SYSTEM



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

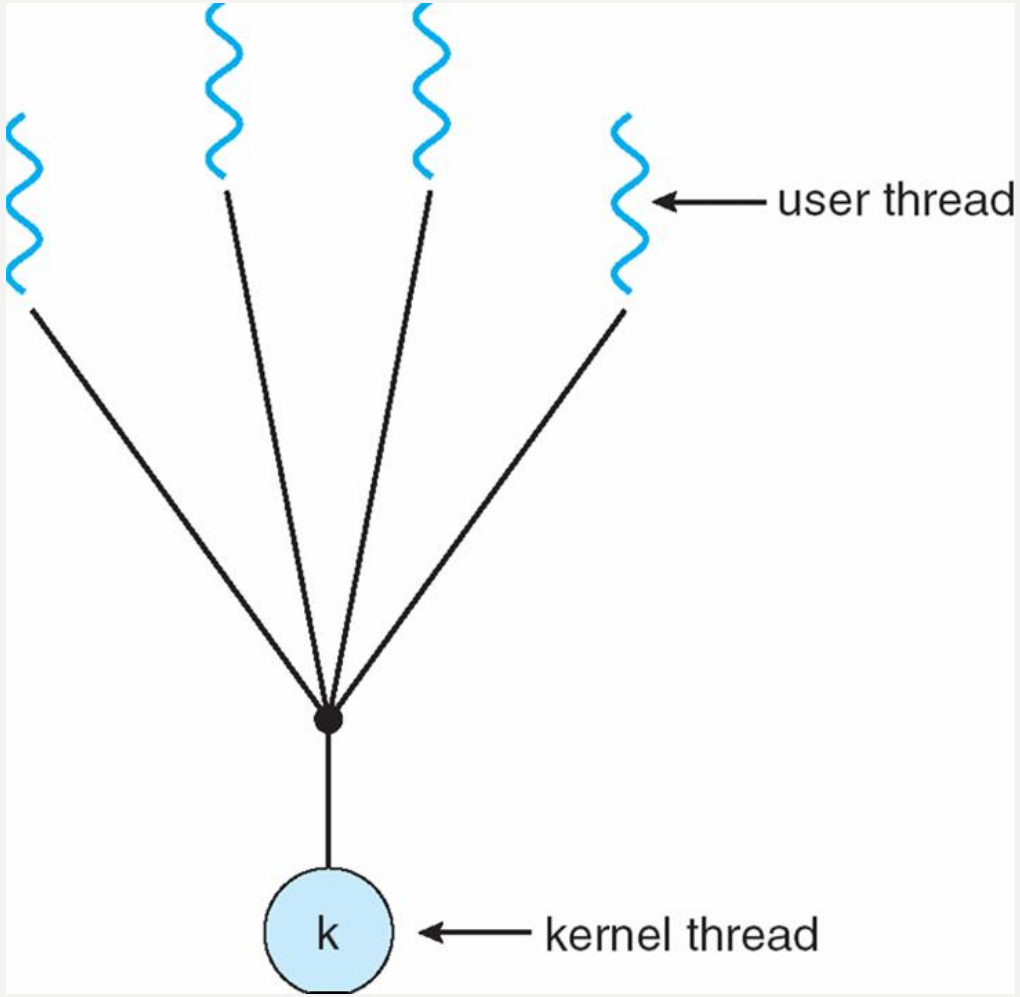
MULTITHREADING MODELS

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

MANY-TO-ONE

- Many user-level threads mapped to single kernel thread

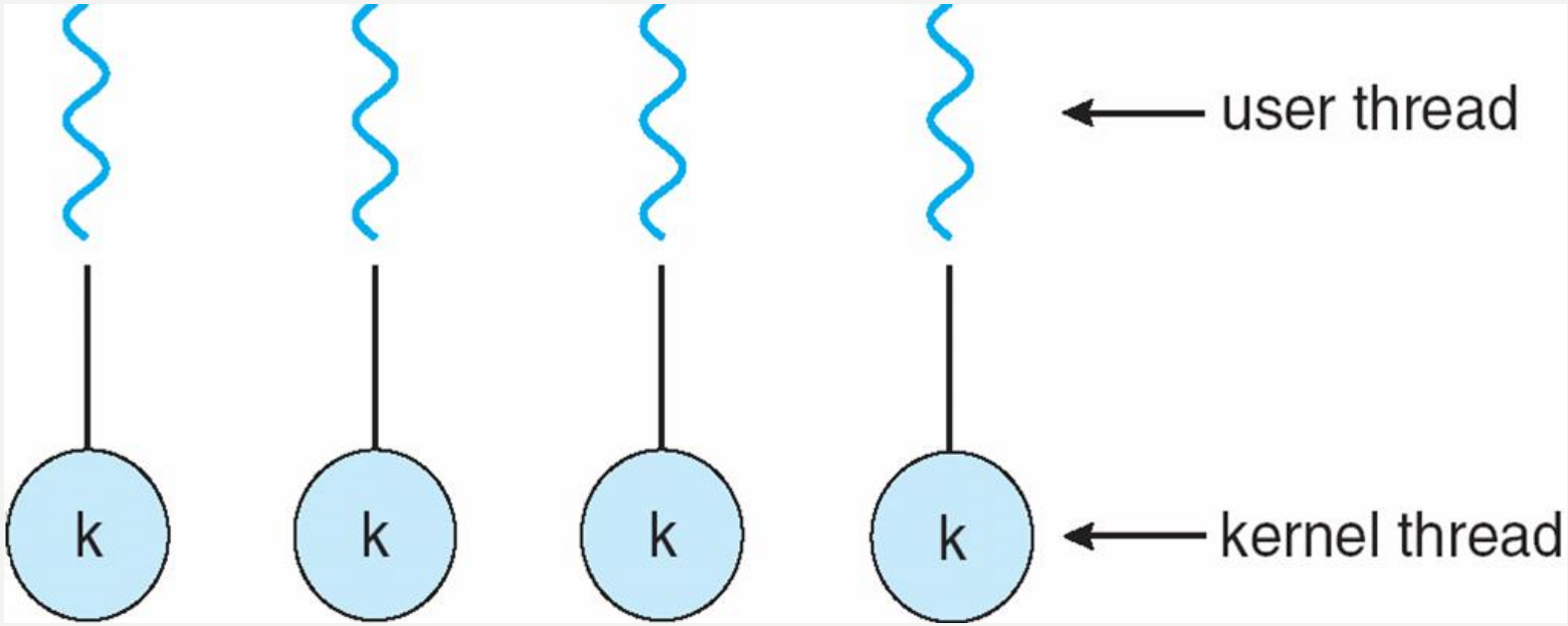
MANY-TO-ONE MODEL



ONE-TO-ONE

- Each user-level thread maps to kernel thread
- Examples
  - Windows NT/XP/2000
  - Linux
  - Solaris 9 and later

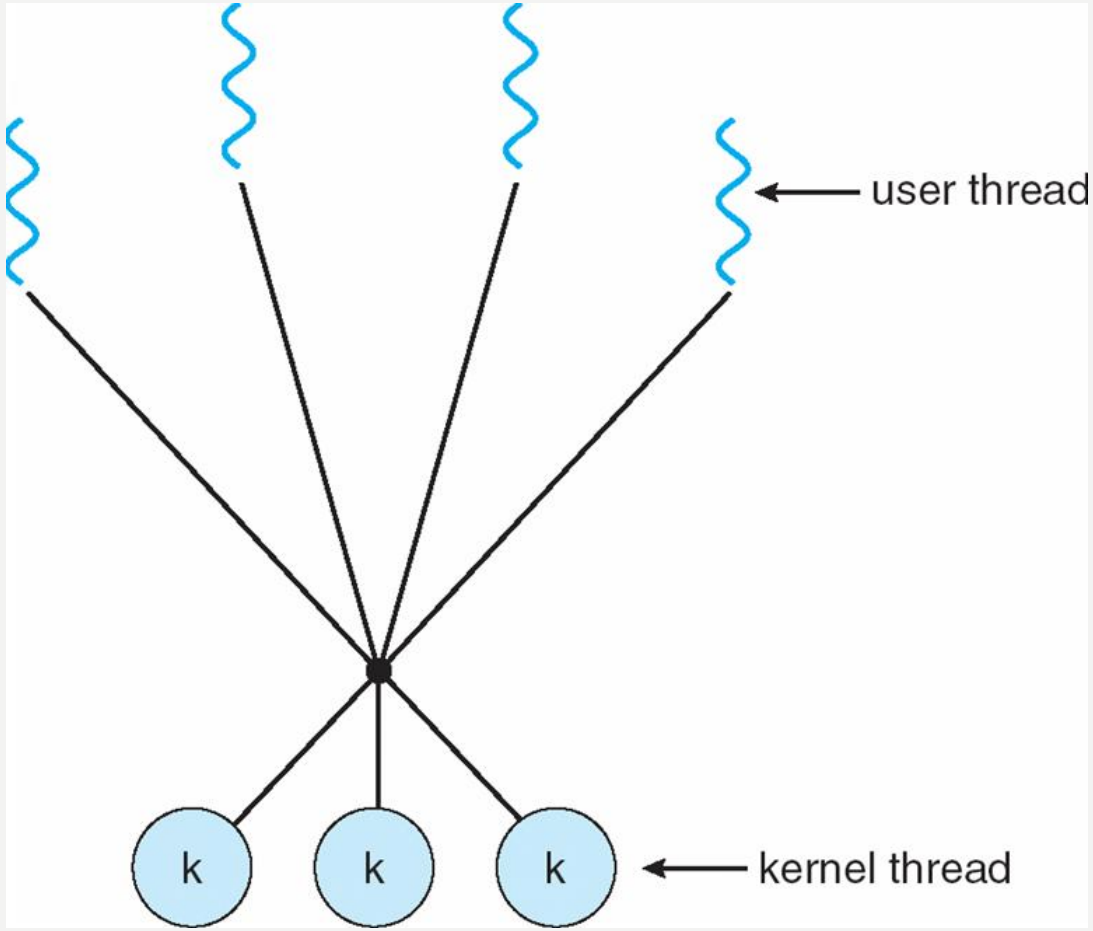
ONE-TO-ONE MODEL



MANY-TO-MANY MODEL

- Allows many user level threads to be mapped to many kernel threads
- Allows the operating system to create a sufficient number of kernel threads
- Solaris prior to version 9
- Windows NT/2000 with the *ThreadFiber* package

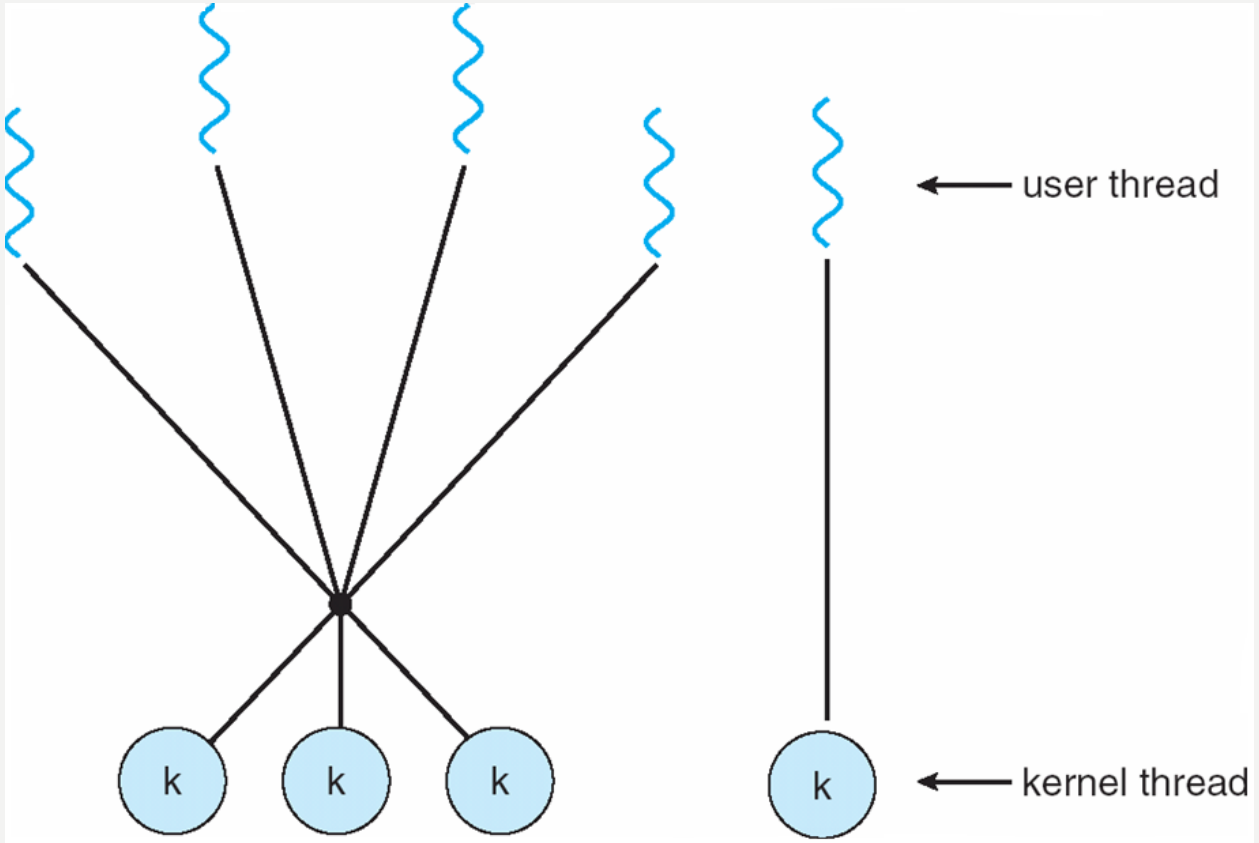
MANY-TO-MANY MODEL



TWO-LEVEL MODEL

- Similar to M:M, except that it allows a user thread to be **bound** to kernel thread
- Examples
  - IRIX
  - HP-UX
  - Tru64 UNIX
  - Solaris 8 and earlier

TWO-LEVEL MODEL



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

JAVA THREADS

- Java threads are managed by the JVM
- Typically implemented using the threads model provided by underlying OS
- Java threads may be created by:
  - Extending Thread class
  - Implementing the Runnable interface

THREADING ISSUES

- Semantics of **fork()** and **exec()** system calls
- Thread cancellation of target thread
  - Asynchronous or deferred
- Signal handling
- Thread pools
- Thread-specific data
- Scheduler activations

SEMANTICS OF FORK() AND EXEC()

- Does **fork()** duplicate only the calling thread or all threads?

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

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

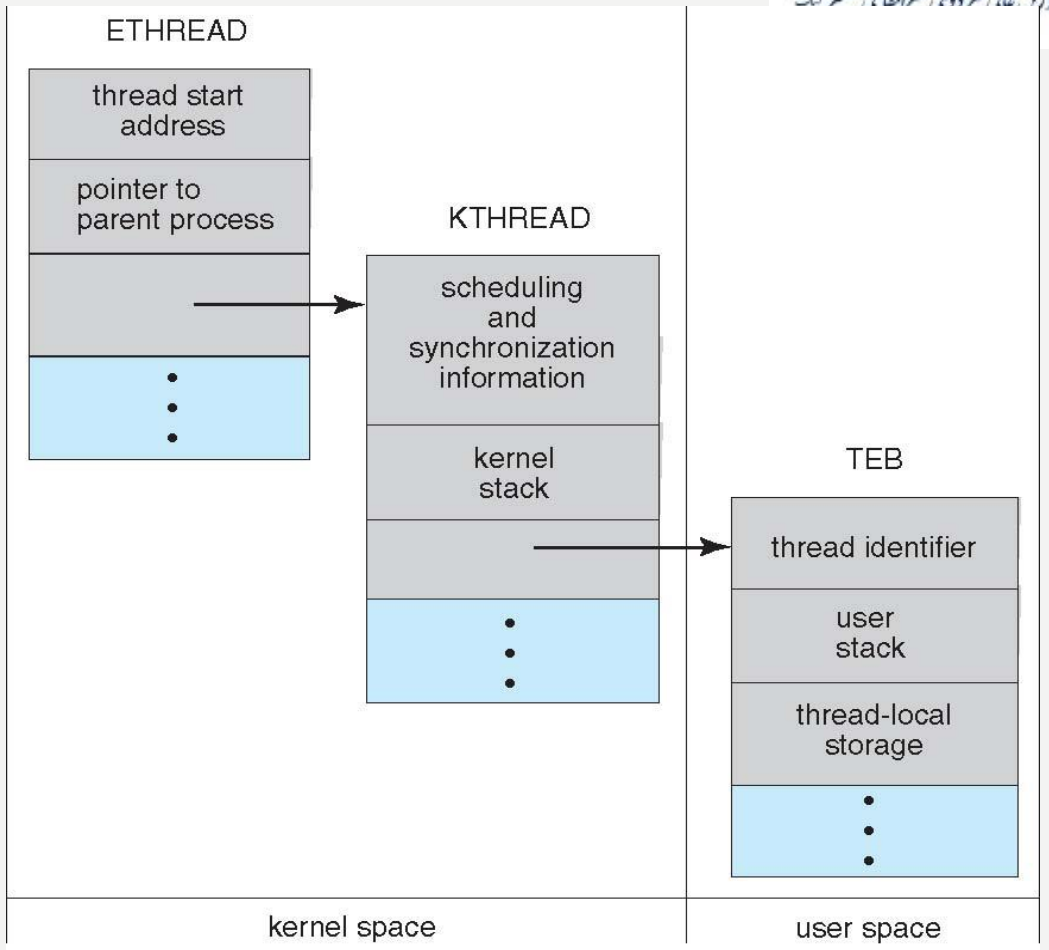
- Allows each thread to have its own copy of data
- Useful when you do not have control over the thread creation process (i.e., when using a thread pool)

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

WINDOWS XP THREADS

- Implements the one-to-one mapping, kernel-level
- Each thread contains
  - A thread id
  - Register set
  - Separate user and kernel stacks
  - Private data storage area
- The register set, stacks, and private storage area are known as the context of the threads
- The primary data structures of a thread include:
  - ETHREAD (executive thread block)
  - KTHREAD (kernel thread block)
  - TEB (thread environment block)



Q/A

- End of Session 4

THANK YOU!