Chapter 4: Threads

Chapter 4: Threads

- Overview
- Multithreading Models
- Threading Issues
- Java Threads
- Pthreads

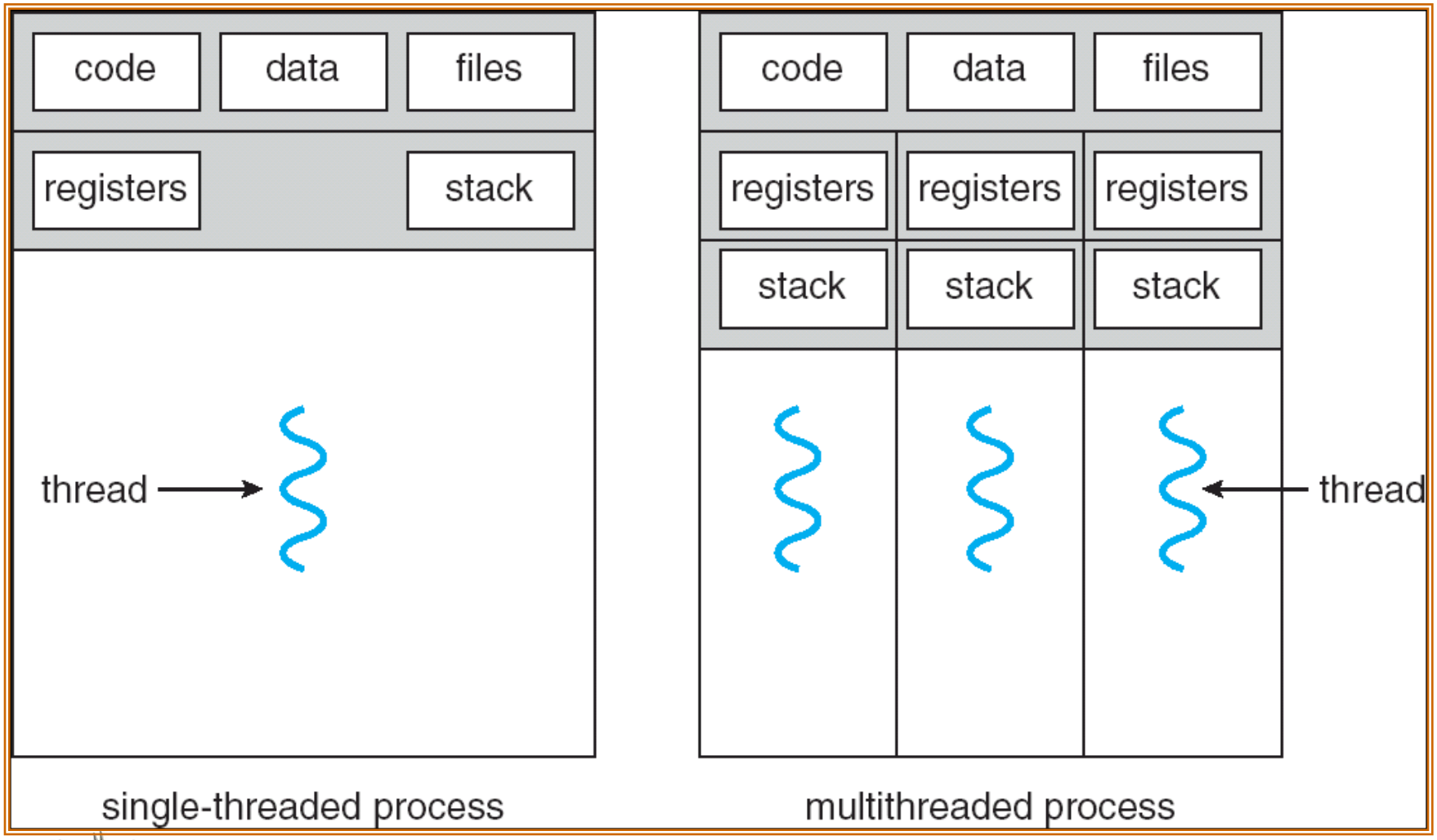
Process/Thread

- Resource ownership - process includes a virtual address space to hold the process image
- Scheduling/execution - follows an execution path that may be interleaved with other processes
- These two characteristics are treated independently by the operating system
  - Process: Ownership of memory, files, other resources
    - 资源分配的基本单元
  - Thread: Unit of execution we use to dispatch
    - 指令执行的基本单元

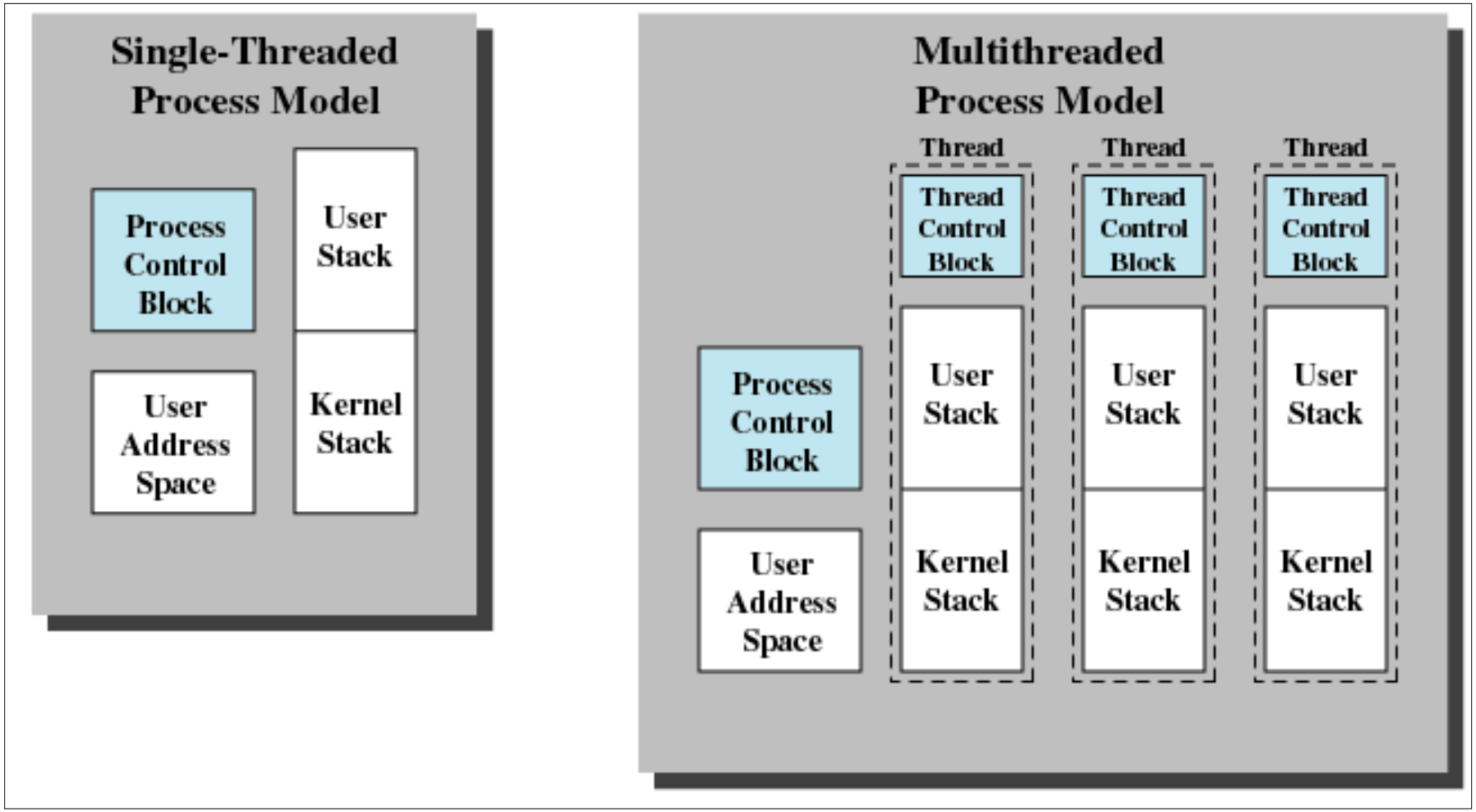
Thread Control Block

- Each Thread has a Thread Control Block (TCB)
  - Execution State: CPU registers, program counter, pointer to stack
  - Scheduling Info: State, priority, CPU time
  - Accounting Info
  - Various Pointers (for implementing scheduling queues)
  - Pointer to enclosing process (PCB)

Single and Multithreaded Processes



Single and Multithreaded Processes



Benefits

- Responsiveness (响应速度快)
- Resource Sharing (共享进程的资源)
- Economy (经济)
- Utilization of MP (多处理器) Architectures

User and Kernel Threads

- User threads - Thread management done by user-level threads library.
  - Three primary thread libraries:
    - POSIX Pthreads
    - Win32 threads
    - Java threads
- Kernel threads - Threads directly supported by the kernel.
  - Downside of kernel threads: a bit expensive

Kernel Threads

Examples (多进程，多线程):

- Windows XP/2000
- Solaris
- Linux
- Tru64 UNIX
- Mac OS X

Multithreading Models

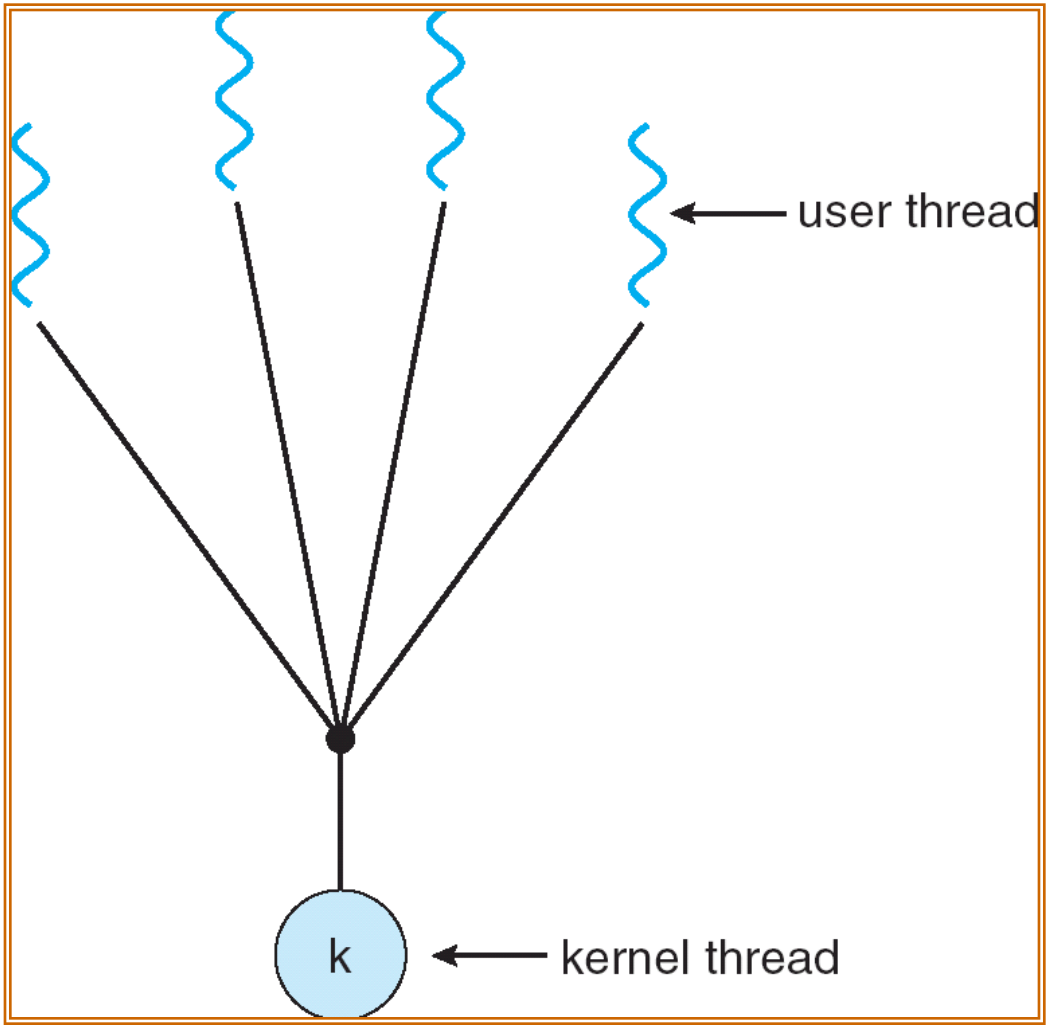
Mapping user threads to kernel threads:

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

Many-to-One

- Many user-level threads mapped to single kernel thread
- Examples:
  - Solaris Green Threads
  - GNU Portable Threads

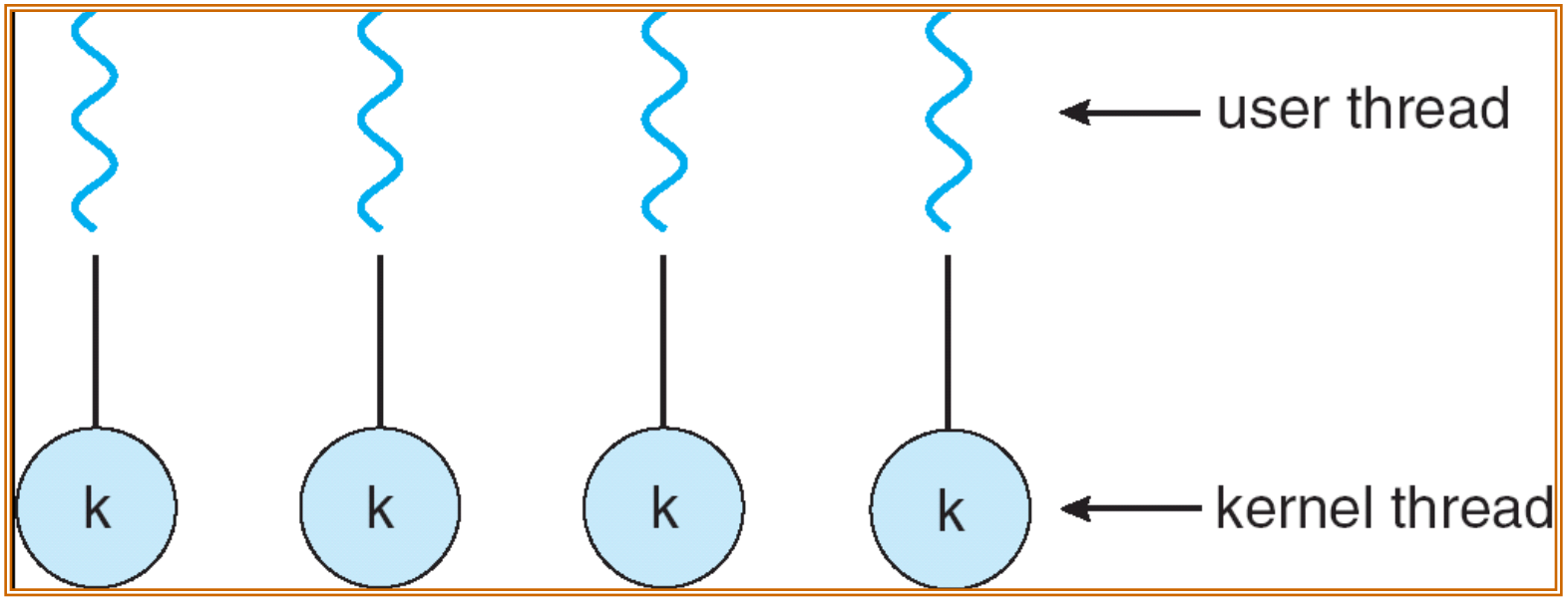
Many-to-One Model



One-to-One

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

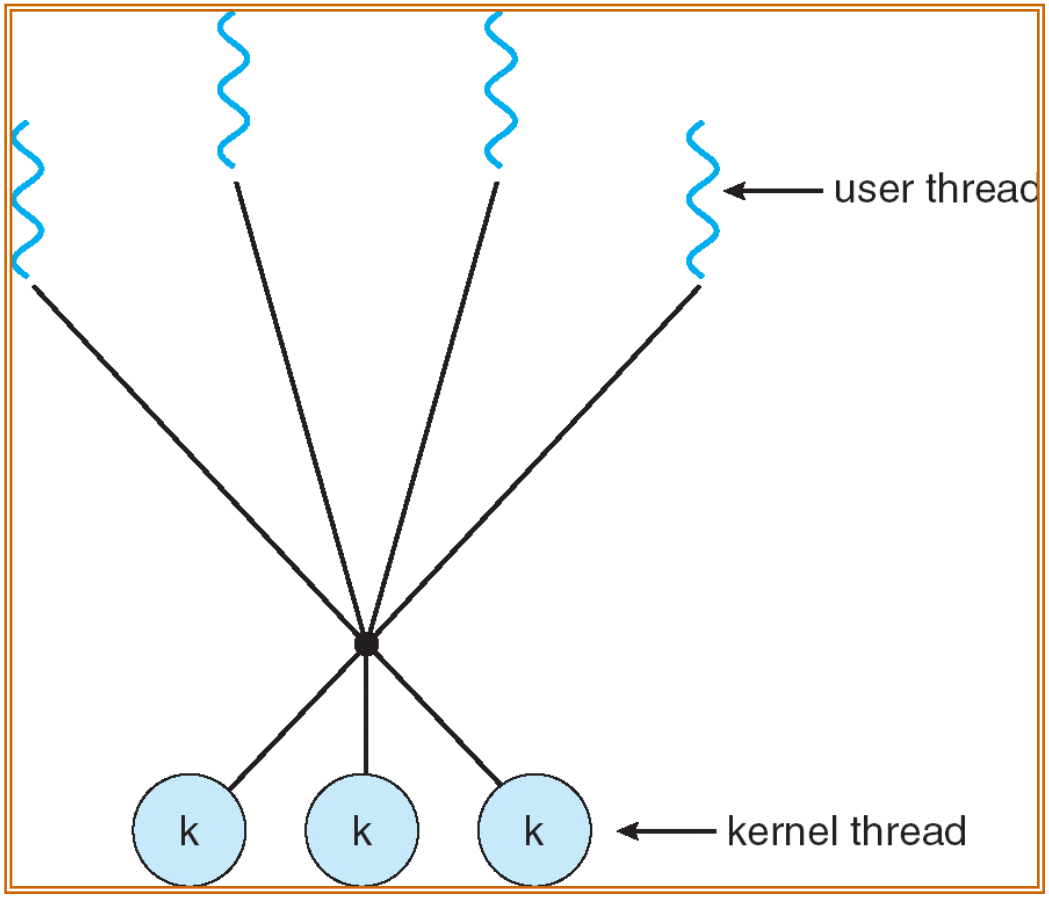
One-to-one Model



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

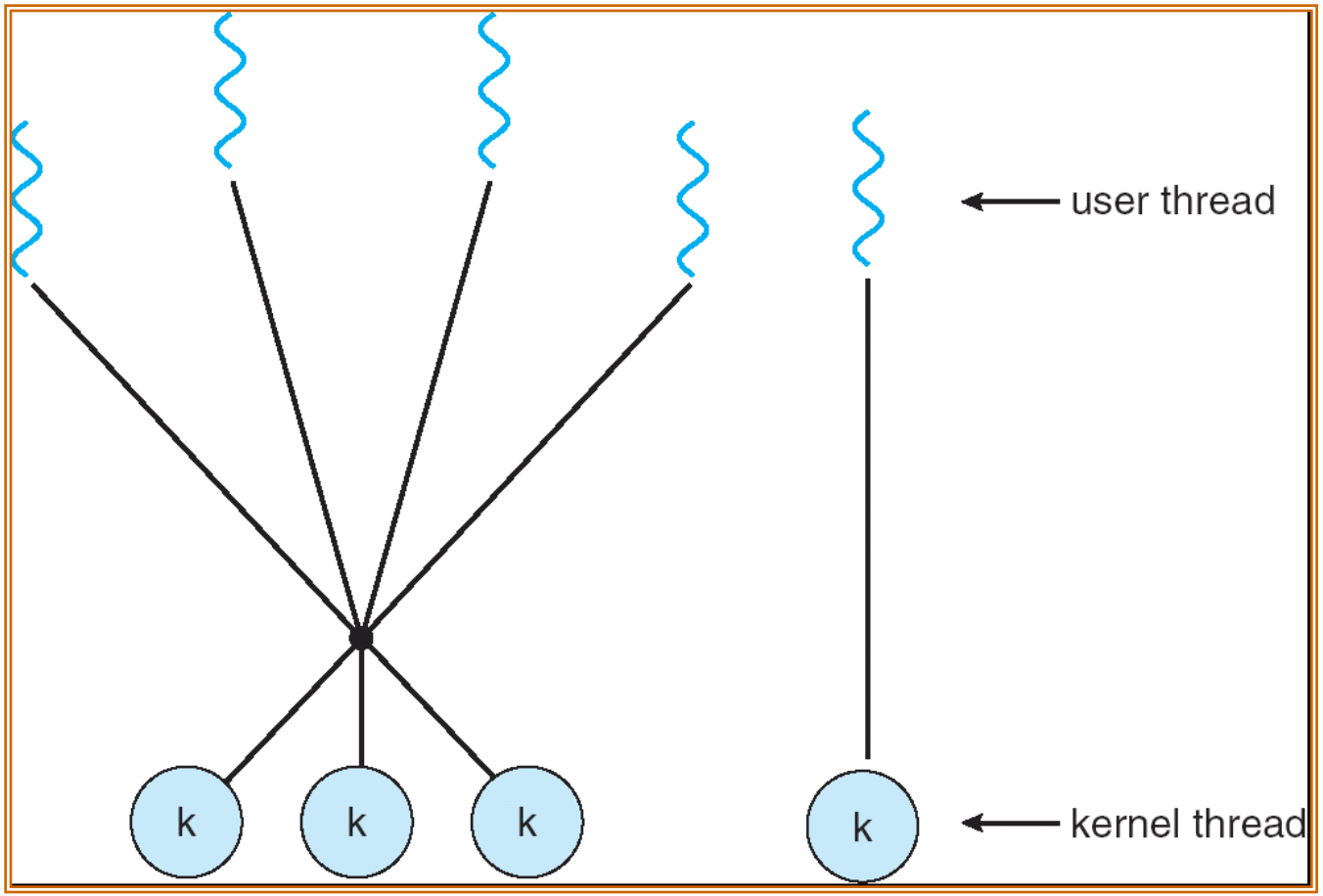
Many-to-Many Model



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



Java Threads

- Java threads are managed by the JVM
- Java threads may be created by:
  - Implementing the Runnable interface

```
public interface Runnable
{
    public abstract void run();
}
```

Java Threads - Example Program

```
class MutableInteger
{
   private int value;
   public int getValue() {
     return value;
   }
   public void setValue(int value) {
     this.value = value;
   }
}

class Summation implements Runnable
{
   private int upper;
   private MutableInteger sumValue;
   public Summation(int upper, MutableInteger sumValue) {
     this.upper = upper;
     this.sumValue = sumValue;
   }
   public void run() {
     int sum = 0;
     for (int i = 0; i <= upper; i++)
        sum += i;
     sumValue.setValue(sum);
   }
}
```

Java Threads - Example Program

```
public class Driver
{
  public static void main(String[] args) {
    if (args.length > 0) {
      if (Integer.parseInt(args[0]) < 0)
        System.err.println(args[0] + " must be >= 0.");
      else {
        // create the object to be shared
        MutableInteger sum = new MutableInteger();
        int upper = Integer.parseInt(args[0]);
        Thread thrd = new Thread(new Summation(upper, sum));
        thrd.start();
        try {
          thrd.join();
          System.out.println
                  ("The sum of "+upper+" is "+sum.getValue());
        } catch (InterruptedException ie) { }
      }
    }
    else
      System.err.println("Usage: Summation <integer value>");
    }
}
```

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

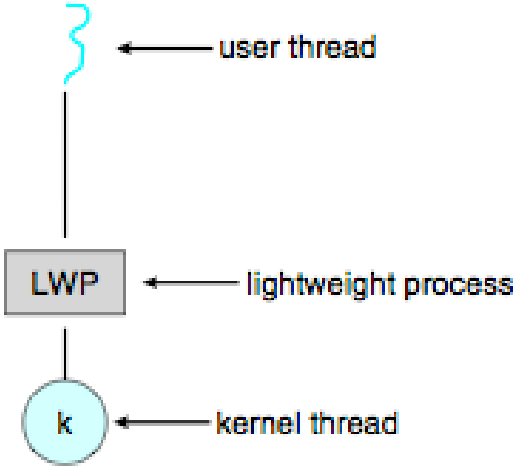
Thread Pools

- Java provides 3 thread pool architectures:

1. **Single thread executor** - pool of size 1.
   - `static ExecutorService newSingleThreadExecutor()`
2. **Fixed thread executor** - pool of fixed size.
   - `static ExecutorService newFixedThreadPool(int nThreads)`
3. **Cached thread pool** - pool of unbounded size
   - `static ExecutorService newCachedThreadPool()`

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)

End of Chapter 4