Chapter 5: Threads

- Overview
- Multithreading Models
- Threading Issues
- Pthreads
- Windows XP Threads
- Linux Threads
- Java Threads

More About Processes

- A process encapsulates a running program, providing an execution state along with certain resources, including file handles and registers, along with:
  - a program counter (Instruction Pointer)
  - a process id, a process group id, etc.
  - a process stack
  - one or more data segments
  - a heap for dynamic memory allocation
  - a process state (running, ready, waiting, etc.)
- Informally, a process is an executing program

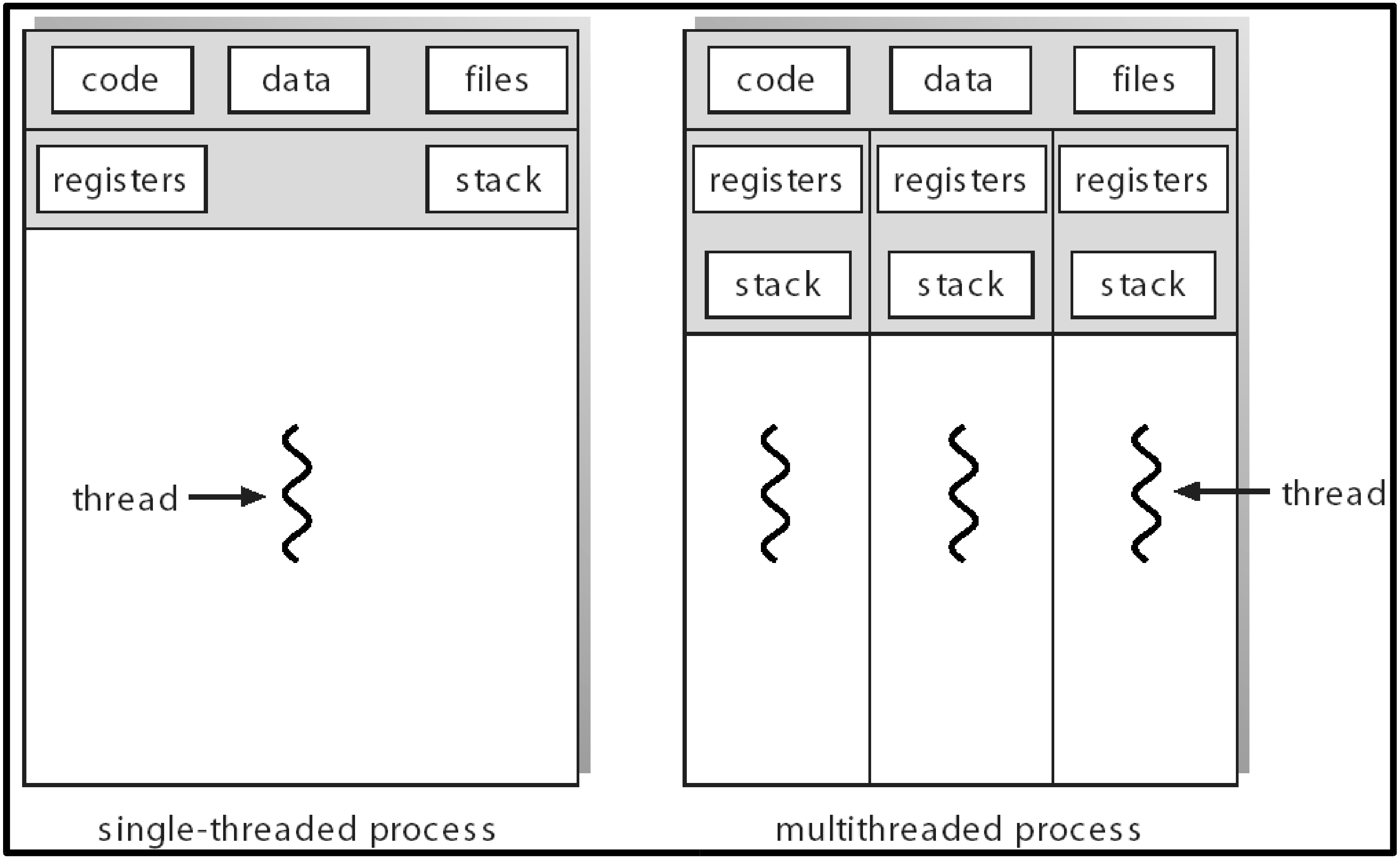
Multiprocessing

- A multiprocessing or multitasking operating system (like Unix, as opposed to DOS) can have more than one process executing at any given time
- This simultaneous execution may either be
  - concurrent, meaning that multiple processes in a run state can be swapped in and out by the OS
  - parallel, meaning that multiple processes are actually running at the same time on multiple processors

What is a Thread?

- A thread is an encapsulation of some flow of control in a program, that can be independently scheduled
- Each process is given a single thread by default
- A thread is sometimes called a lightweight process, because it is similar to a process in that it has its own thread id, stack, stack pointer, a signal mask, program counter, registers, etc.
- All threads within a given process share resource handles, memory segments (heap and data segments), and code.

Single and Multithreaded Processes



Process/Thread

A PROCESS	A THREAD
Process ID	Thread ID
Program Counter	Program Counter
Signal Dispatch Table	Signal Dispatch Table
Registers	Registers
Process Priority	Thread Priority
Stack Pointer & Stack	Stack Pointer & Stack
Heap	All threads share the same memory, heap, and file handles (and offsets)
Memory Map	
File Descriptor Table	

Benefits

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

Processes and Threads: Creation Times

- Because threads are by definition lightweight, they can be created more quickly that “heavy” processes:
  - Sun Ultra5, 320 Meg Ram, 1 CPU
    - 94 forks()/second
    - 1,737 threads/second (18x faster)
  - Sun Sparc Ultra 1, 256 Meg Ram , 1 CPU
    - 67 forks()/second
    - 1,359 threads/second (20x faster)
  - Sun Enterprise 420R, 5 Gig Ram, 4 CPUs
    - 146 forks()/second
    - 35,640 threads/second (244x faster)
  - Linux 2.4 Kernel, .5 Gig Ram, 2 CPUs
    - 1,811 forks()/second
    - 227,611 threads/second (125x faster)

Benefits of Multithreading

- Performance gains
  - Amdahl's Law: $\text{speedup} = 1 / ((1 - p) + (p/n))$
  - the speedup generated from parallelizing code is the time executing the parallelizable work (p) divided by the number of processors (n) plus 1 minus the parallelizable work (1-p)
  - The more code that can run in parallel, the faster the overall program will run
  - If you can apply multiple processors for 75% of your program's execution time, and you're running on a dual processor box:
    - $1 / ((1 - .75) + (.75 / 2))$ = 60% improvement
  - Why is it not strictly linear? How do you calculate p?

User Threads

- Thread management done by user-level threads library
- Three primary thread libraries:
  - POSIX (IEEE Portable Operating System Interface) Pthreads
  - Java threads
  - Win32 threads

Kernel Threads

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

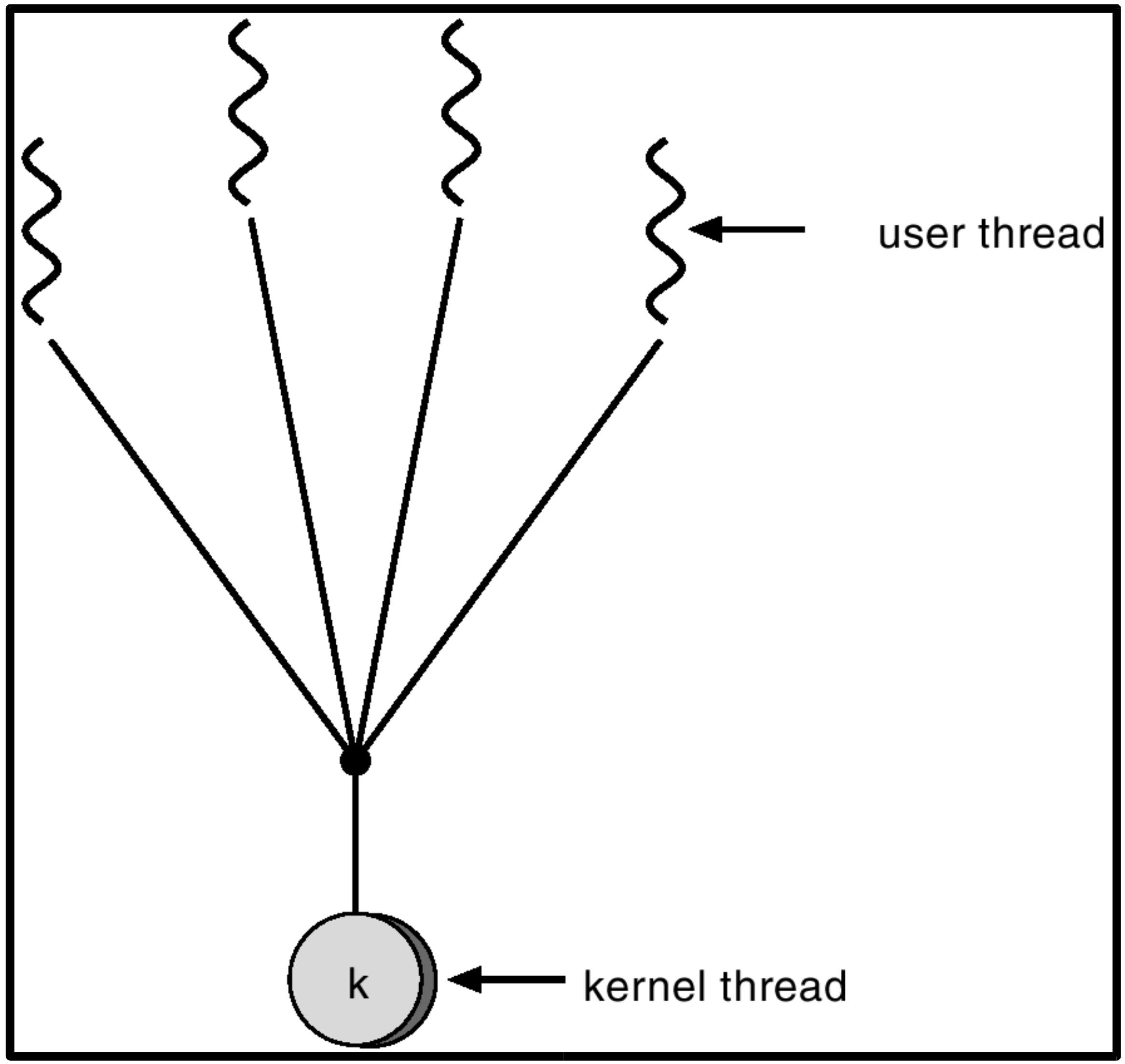
Multithreading Models

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

Many-to-One

- Many user-level threads mapped to single kernel thread
- Examples
  - Solaris Green Threads
    - used by early JVMs
  - GNU Portable Threads

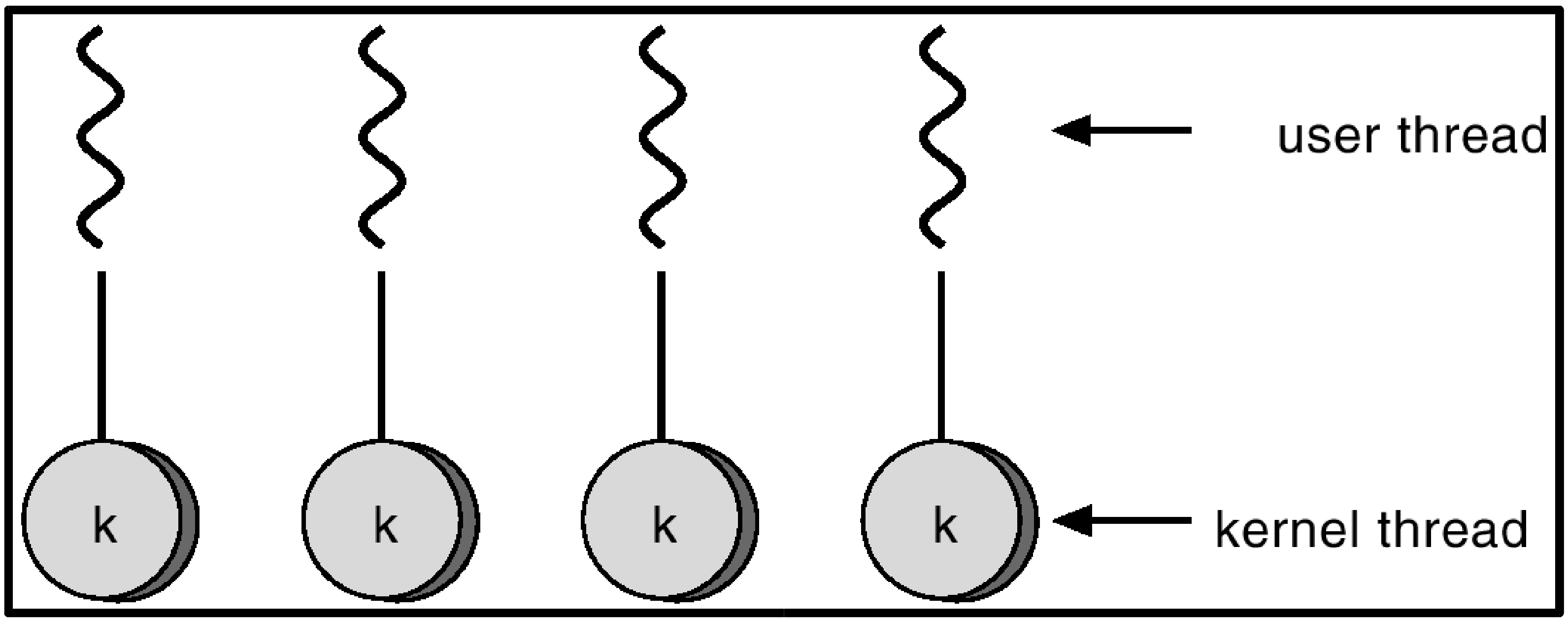
Many-to-One Model



One-to-One

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

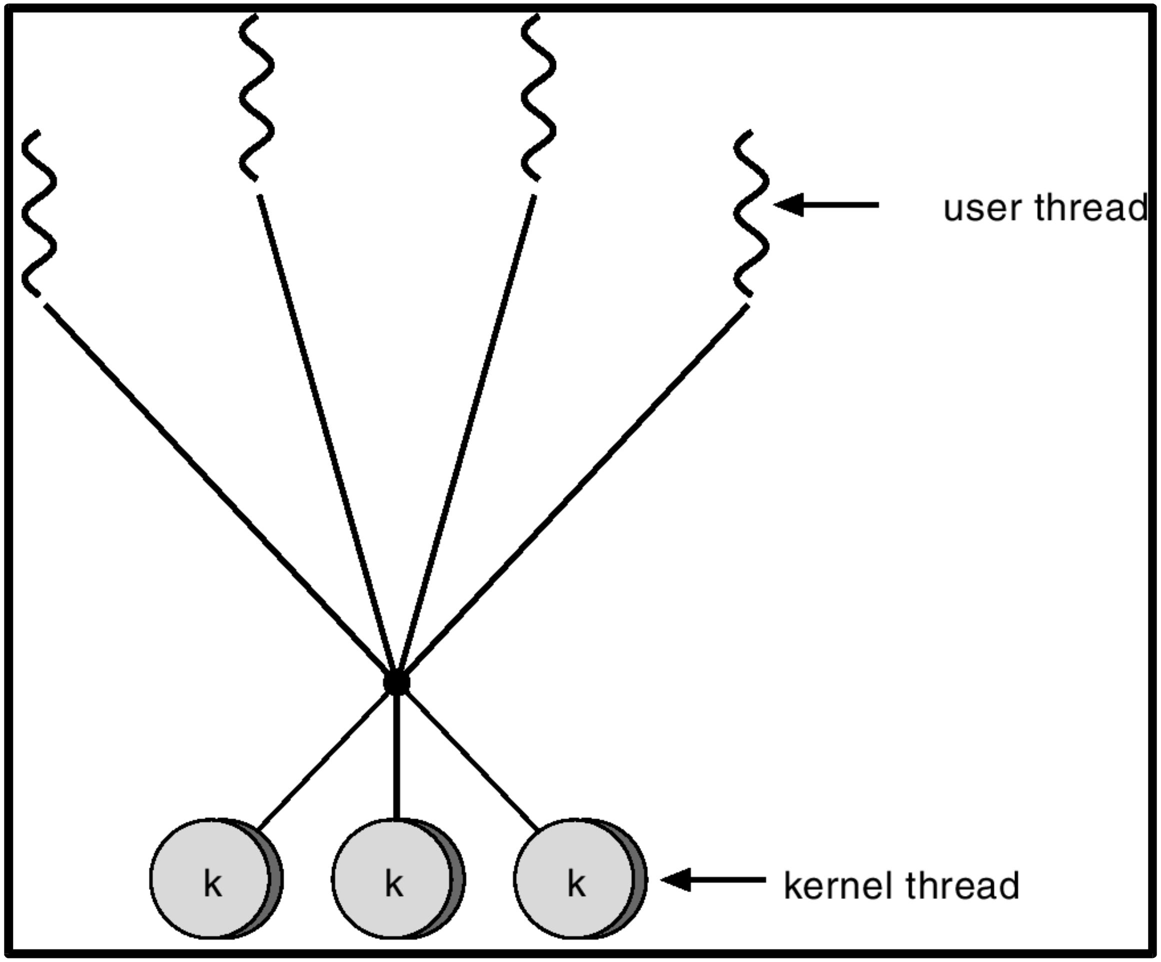
One-to-one Model



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

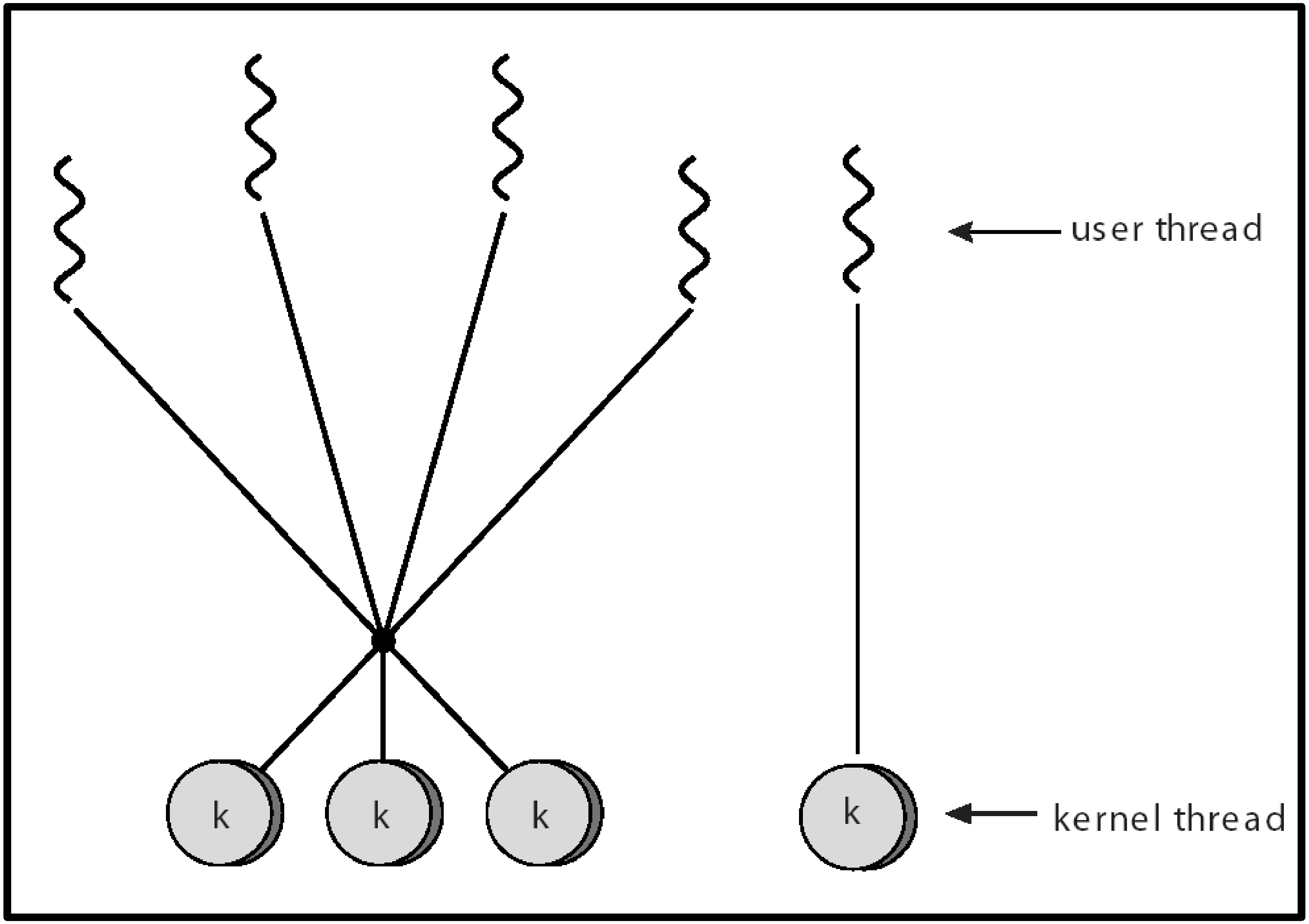
Many-to-Many Model



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



Threading Issues

- Semantics of fork() and exec() system calls
  - Thread cancellation
  - Signal handling
  - Thread pools
  - Thread specific data
  - Scheduler activations
- 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
  - Signal is generated by particular event
    - CPU interrupt, I/O completion, mouse click, ...
  - Signal is delivered to a process
  - 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 threa 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

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)

POSIX

- Each OS had its own thread library and style
- That made writing multithreaded programs difficult because:
  - you had to learn a new API with each new OS
  - you had to modify your code with each port to a new OS
- POSIX (IEEE 1003.1c-1995) provided a standard known as Pthreads
- Unix International (UI) threads (Solaris threads) are available on Solaris (which also supports POSIX threads)

Windows XP Threads

- Implements the one-to-one mapping
- 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)

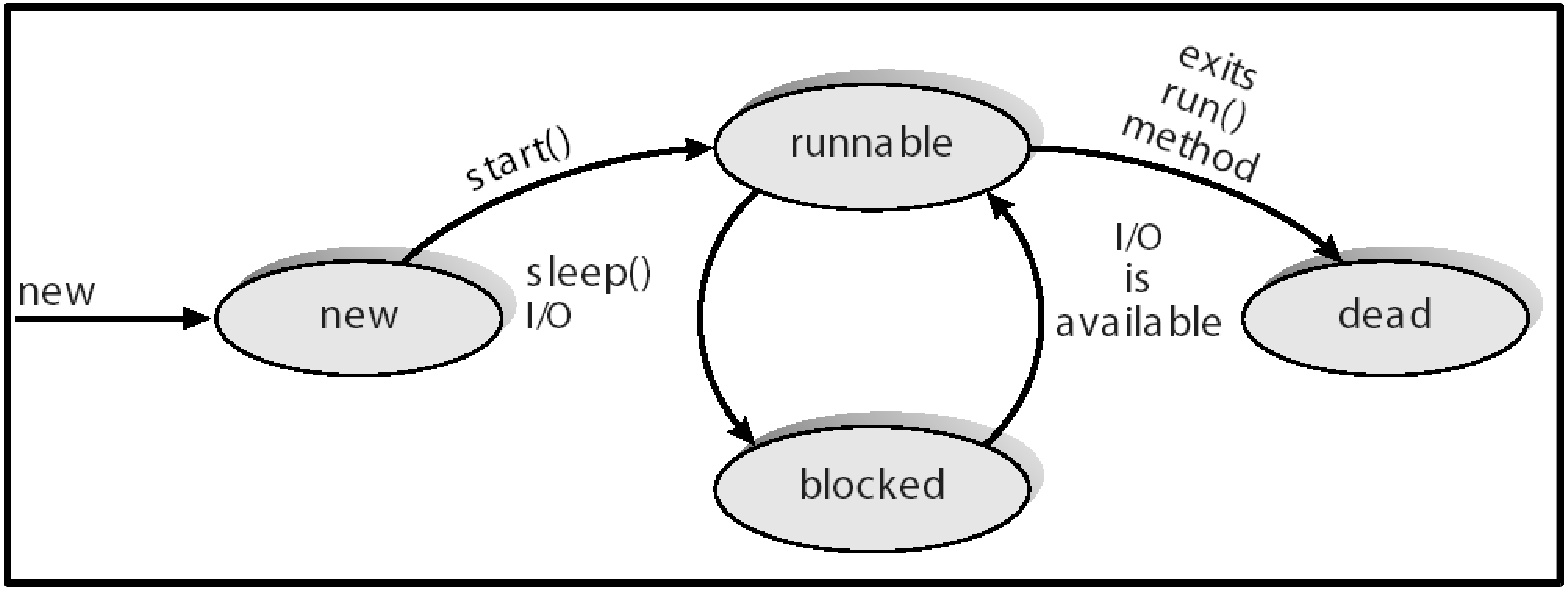
Linux Threads

- 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)

Java Threads

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

Java Thread States



On the Scheduling of Threads

- Threads may be scheduled by the system scheduler (OS) or by a scheduler in the thread library (depending on the threading model).
- The scheduler in the thread library:
  - will preempt currently running threads on the basis of priority
  - does NOT time-slice (i.e., is not fair). A running thread will continue to run forever unless:
    - a thread call is made into the thread library
    - a blocking call is made
    - the running thread calls sched_yield()

Chapter 5 Homework

- Write a multithreaded program
  - Java, or Pthreads
  - pg 169, 5.9, 5.10, 5.11, OR an MT program of your choice
  - write, compile, run, and monitor your program as it runs
    - show source code, output of run
    - show results of monitoring the program run...execution time, memory use, thread execution
  - “write a program” means read/research/understand existing code fragments and examples, prepare a source file, compile and run the program, explain the execution and output
    - it does NOT mean simply copy/modify other's solution to the assignment