OPERATING SYSTEMS

Threads

Jerry Breecher

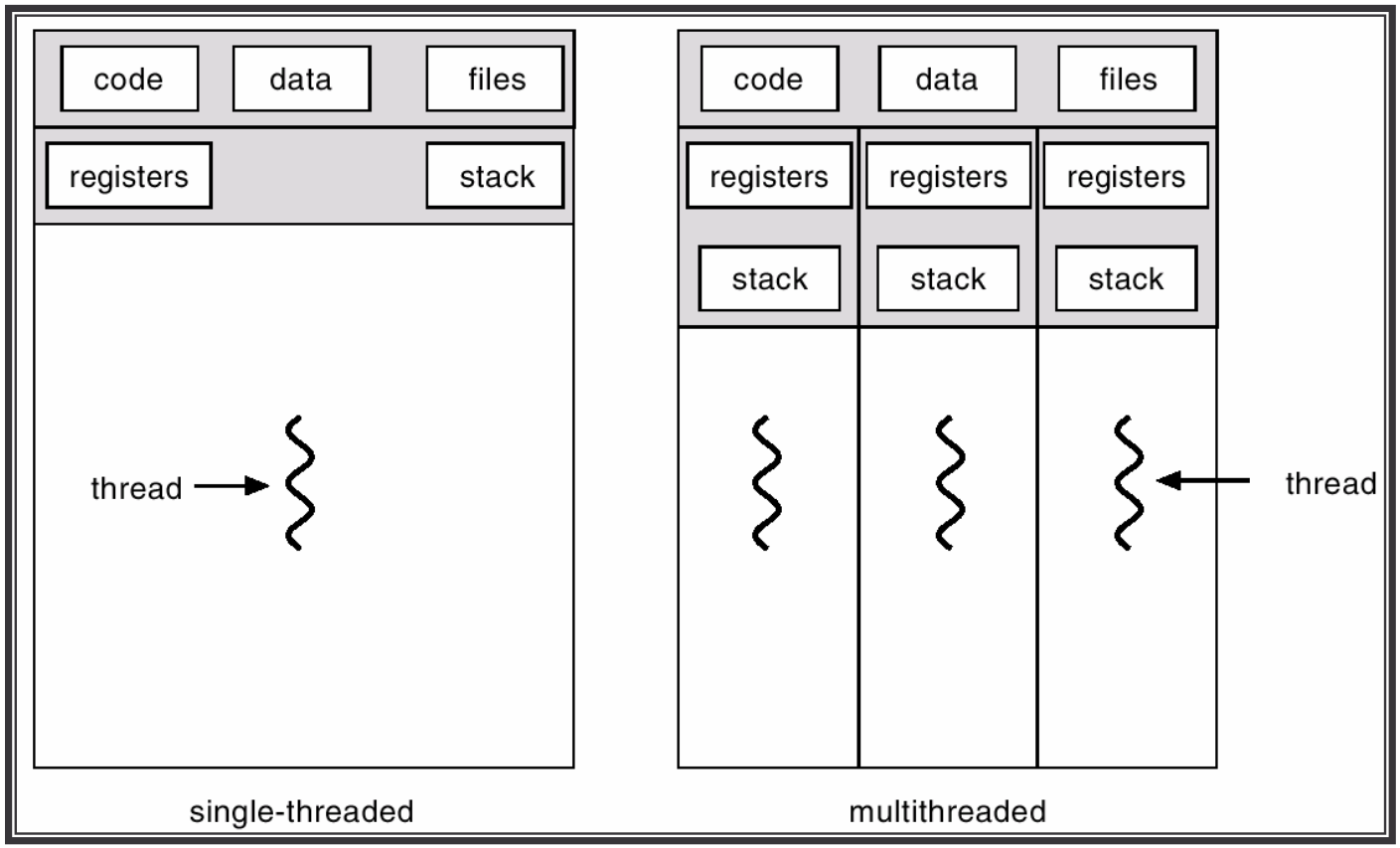
OPERATING SYSTEM
Threads

What Is In This Chapter?

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

THREADS

Single and Multithreaded Processes



THREADS

Benefits

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

THREADS

User Threads

- Thread management done by user-level threads library

- Examples
  - POSIX *Pthreads*
  - Mach *C-threads*
  - Solaris *threads*

Kernel Threads

- Supported by the Kernel

- Examples
  - Windows 95/98/NT/2000
  - Solaris
  - Tru64 UNIX
  - BeOS
  - Linux

THREADS

Multithreading Models

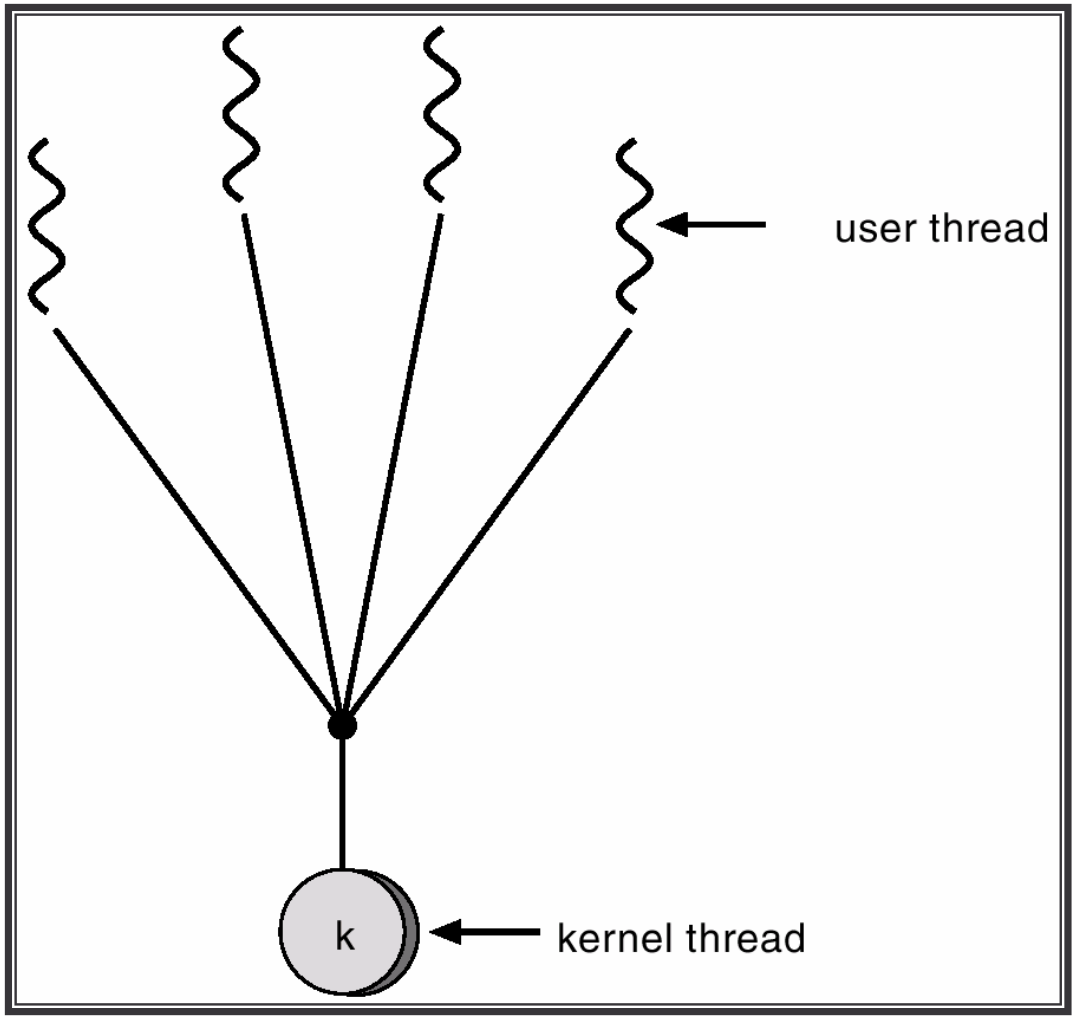
- Many-to-One
- One-to-One
- Many-to-Many

How do user and kernel threads map into each other?

THREADS

Many-to-One

- Many user-level threads mapped to single kernel thread.
- Used on systems that do not support kernel threads.
- Examples:
  - Solaris Green Threads
  - GNU Portable Threads

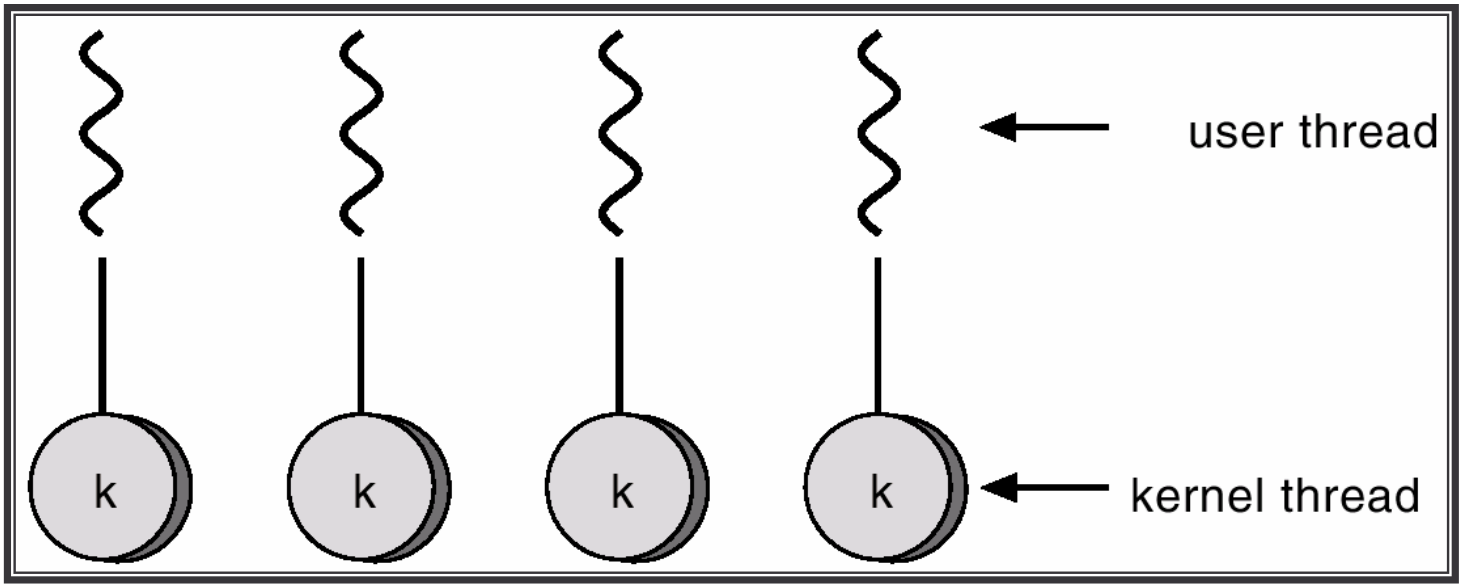


THREADS

One-to-One

- Each user-level thread maps to kernel thread.

- Examples
  - Windows 95/98/NT/2000
  - Linux



THREADS

Semantics of fork() and exec() system calls

- 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

Threading Issues

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 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

THREADS

Threading Issues

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

- Many:Many 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

THREADS

Various Implementations

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)

Windows 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

THREADS

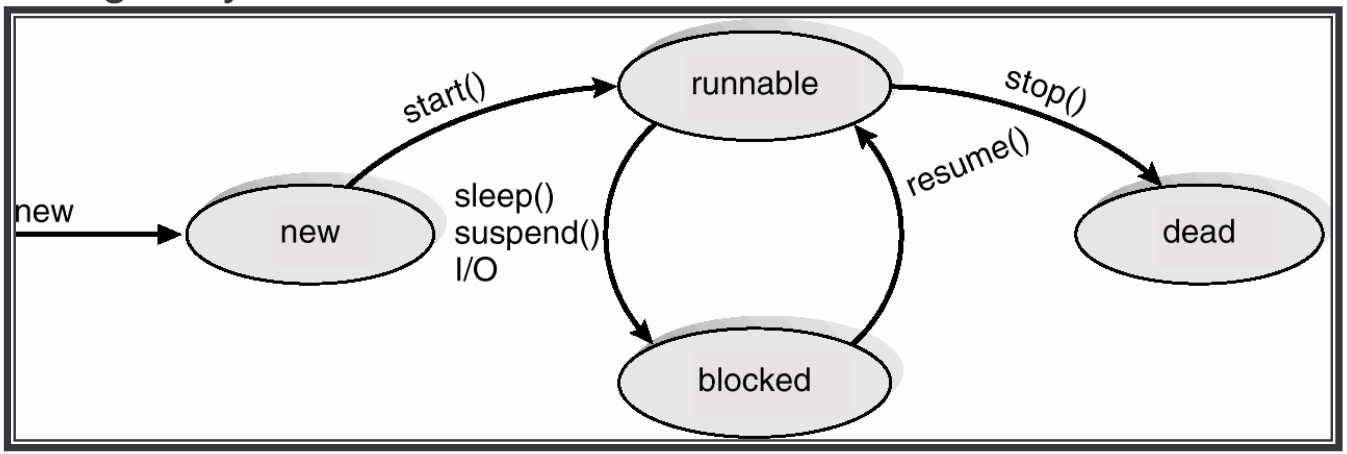
Various Implementations

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



Threads

WRAPUP

We've looked in detail at how threads work. Specifically we've looked at:

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