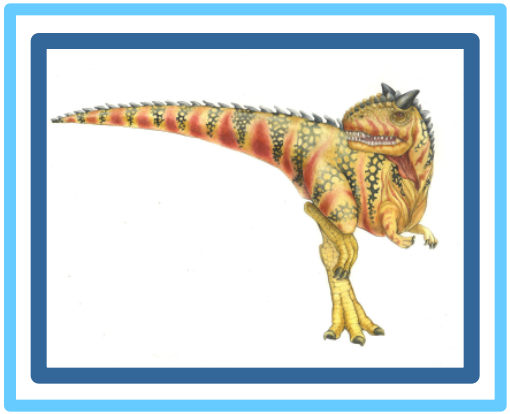
Chapter 4: Threads & Concurrency



Chapter 4: Threads

- Overview
- Multithreading Models
- Threading Issues

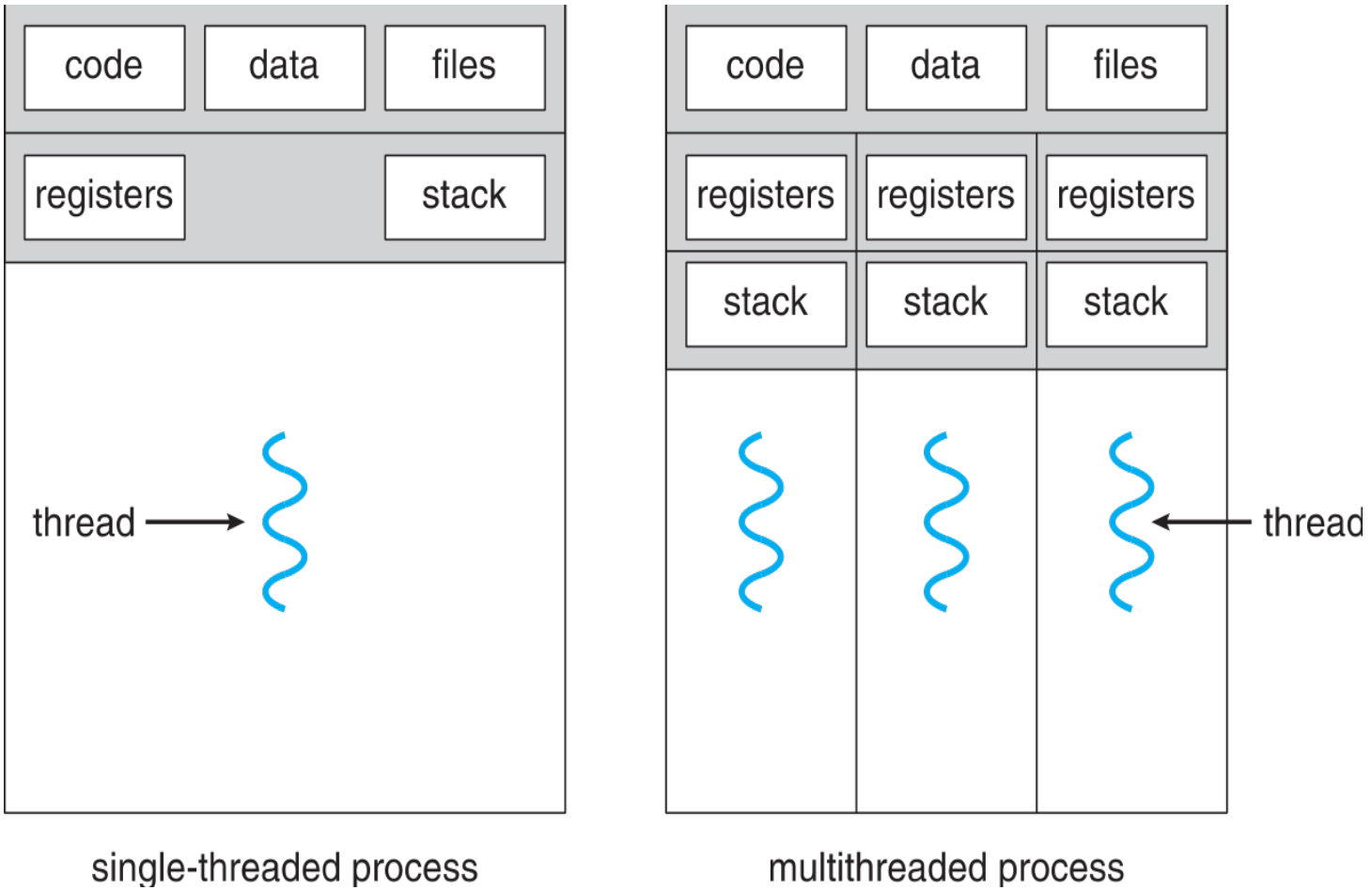
Objectives

- To introduce the notion of a thread—a fundamental unit of CPU utilization that forms the basis of multithreaded computer systems
- To discuss multithreading models
- To examine issues related to multithreaded programming

Thread

- Unit of execution
- Comprises of,
  - A thread ID
  - A program counter
  - A register set
  - A stack
- Share code section, data section, OS resources such as files and signals with other threads belonging to the same processes.
- Single process can have multiple thread and each thread can handle one task at a time.

Single and Multithreaded Processes



single-threaded process

multithreaded process

Benefits

- **Responsiveness –** may allow continued execution if part of process is blocked, especially important for user interfaces
- **Resource Sharing –** threads share resources of process, easier than shared memory or message passing
- **Economy –** cheaper than process creation, thread switching lower overhead than context switching
- **Scalability –** process can take advantage of multiprocessor architectures

User Threads and Kernel Threads

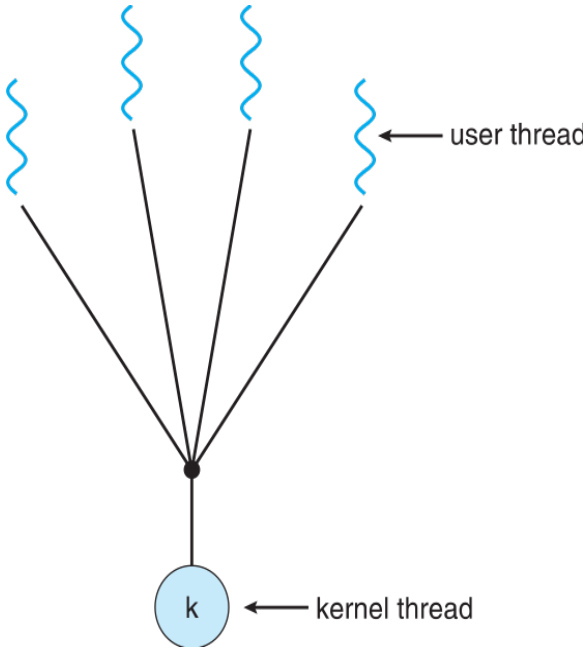
- **User threads** - management done by user-level threads library
- Three primary thread libraries:
  - POSIX **Pthreads**
  - Windows threads
  - Java threads
- **Kernel threads** - Supported by the Kernel
- Examples – virtually all general purpose operating systems, including:
  - Windows
  - Solaris
  - Linux
  - Tru64 UNIX
  - Mac OS X

Multithreading Models

- The relationship between the User Thread and the Kernel Threads are established mainly by three models,
  - Many-to-One
  - One-to-One
  - Many-to-Many

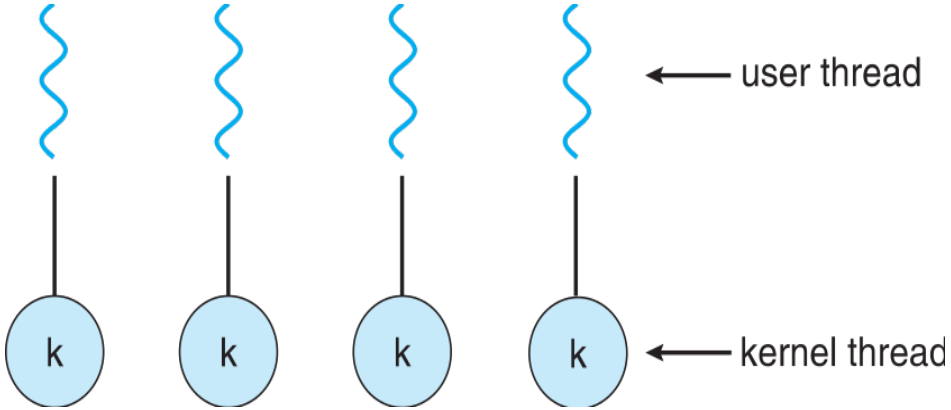
Many-to-One

- Many user-level threads mapped to single kernel thread
- One thread blocking causes all to block
- Multiple threads may not run in parallel on muticore system because only one may be in kernel at a time



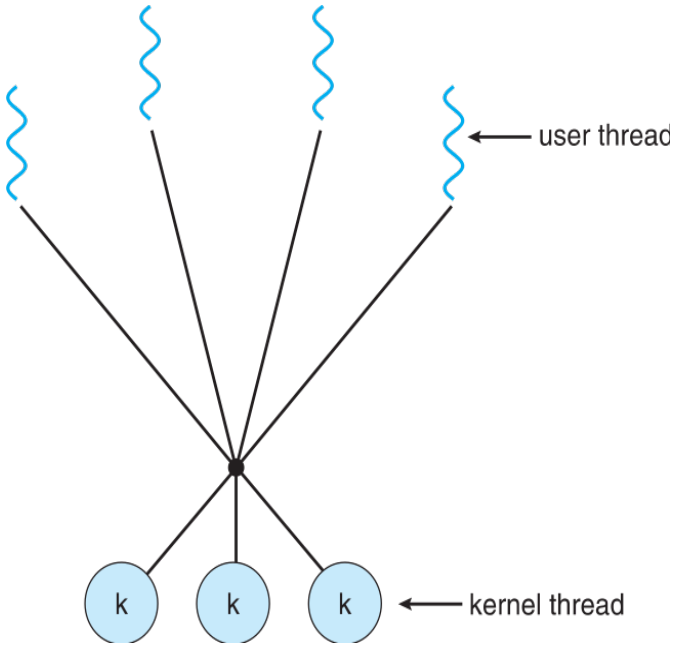
One-to-One

- Each user-level thread maps to kernel thread
- Creating a user-level thread creates a kernel thread
- More concurrency than many-to-one
- Number of threads per process sometimes restricted due to overhead



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



Threading Issues

- Semantics of **fork()** and **exec()** system calls
- Thread cancellation of target thread
  - Asynchronous or deferred

Semantics of fork() and exec()

- What does the fork() system calls do?
  - Duplicate a process or creating a child process with different process ID
- What does exec() system calls do?
  - Replacing the contents of a process with the contents of another process but with same process ID
- Issue — The behaviour of fork() and exec() is different in a multithreaded prog.
- Does **`fork()`** duplicate only the calling thread or all threads?
  - Some UNIXes have two versions of fork
- **`exec()`** usually works as normal – replace the running process including all threads

Semantics of fork() and exec()

- Does `fork()` duplicate only the calling thread or all threads?
  - Some UNIXes have two versions of fork

- **Two types:**
- **fork() – duplicate all threads**
- **fork() – duplicate only the thread that invoked the system call**

- **Which fork() to use and when?**
- **If exec() calls immediately after fork() – in this case duplicate the thread that invoked the call.**
- **If no exec() call after fork() – duplicating all threads will be the one.**

Thread Cancellation

- Terminating a thread before it has finished
- The thread to be cancelled is called **target thread**
- Two general approaches:
  - **Asynchronous cancellation** terminates the target thread immediately
  - **Deferred cancellation** allows the target thread to periodically check if it should be cancelled
- How hard it is to cancel and how it is done?

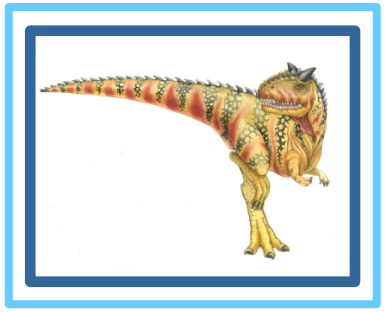
Thread Cancellation

- How hard it is to cancel and how it is done?
- In situations like:
  - Resources are allocated to the target thread
  - When target thread is sharing data with other threads

Chapter 4: Threads

- Overview
- Multithreading Models
- Threading Issues

End of Chapter 4



Quiz

- What are the multithreading models discussed in today's class?
- What is the system call used o duplicate a process with the same process ID?
- exec() system call replaces the contents of a process and creates another one with the same ID — T or F?
- The thread to be cancelled is known as …….