Operating Systems

Chapter 4: Threads

Dr. Ahmed Hagag

Scientific Computing Department,
Faculty of Computers and Artificial Intelligence
Benha University

2019

Chapter 4: Threads

- Introduction
- Multicore Programming
- Multithreading Models
- Benefits of Threads
- Thread Libraries

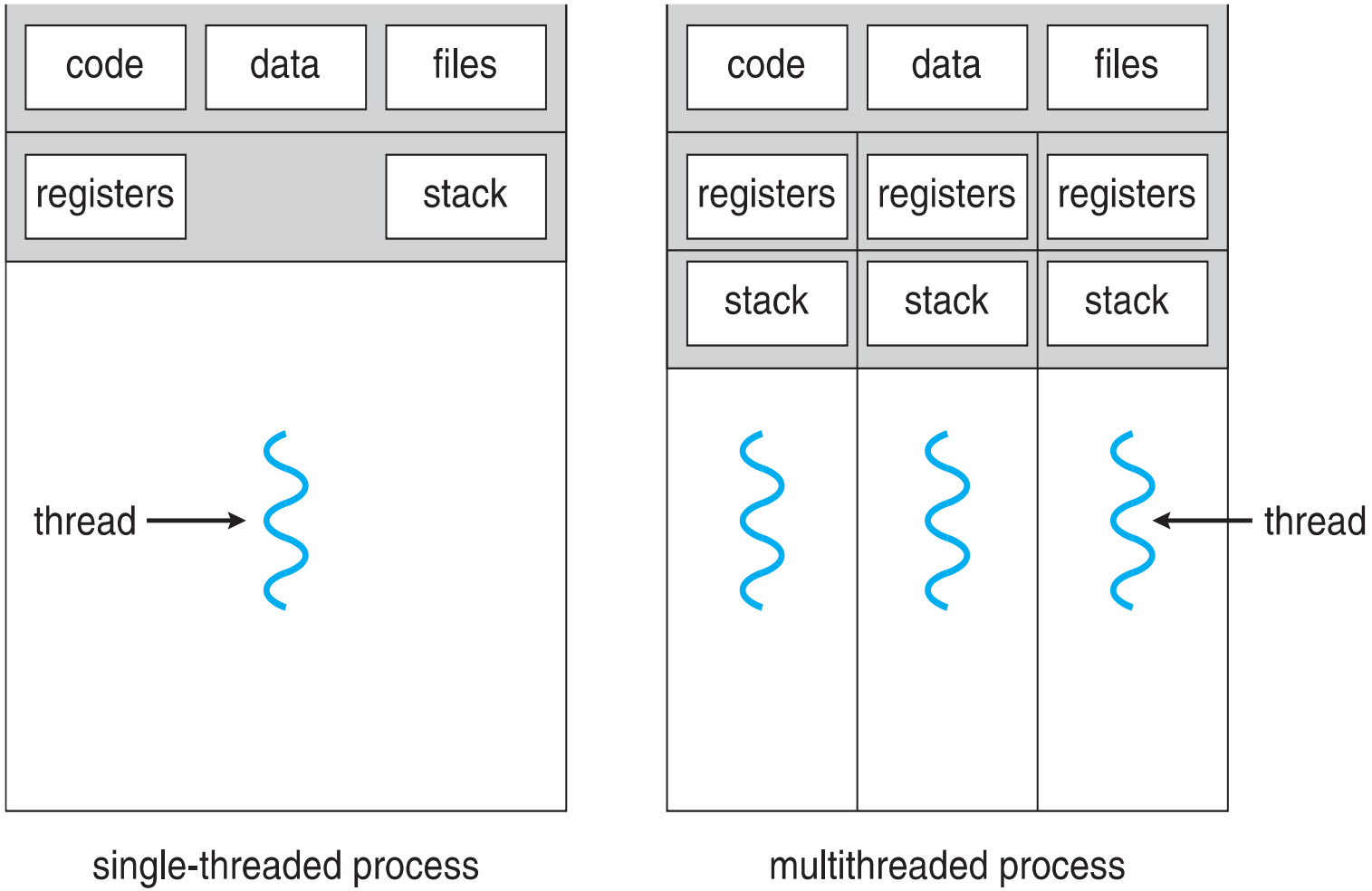
Introduction (1/6)

- A **thread** is a basic unit of CPU utilization; it comprises a thread ID, a program counter, a register set, and a stack.
- It shares with other threads belonging to the same process its **code section**, **data section**, and other operating-system **resources**, such as open files and signals.

Introduction (2/6)

- Let's say, for example, a program is not capable of drawing pictures while reading keystrokes. The program must give its full attention to the keyboard input lacking the ability to handle more than one event at a time.
- The ideal solution to this problem is the seamless execution of two or more sections of a program at the same time. Threads allows us to do this.

Introduction (3/6)



single-threaded process

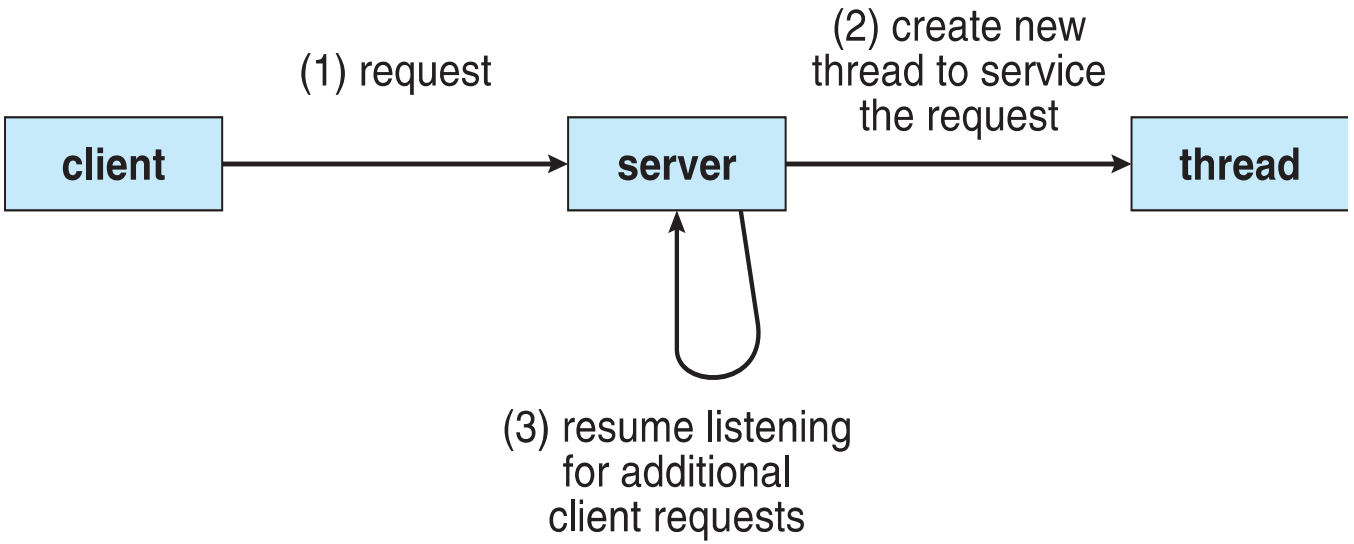
multithreaded process

Introduction (4/6)

- Most software applications that run on modern computers are **multithreaded**.
- An application typically is implemented as a separate process with several threads of control.
  - A web browser might have one thread display images or text while another thread retrieves data from the network, for example.
  - A word processor may have a thread for displaying graphics, another thread for responding to keystrokes from the user, and a third thread for performing spelling and grammar checking in the background.

Introduction (5/6)

- Multithreaded server architecture

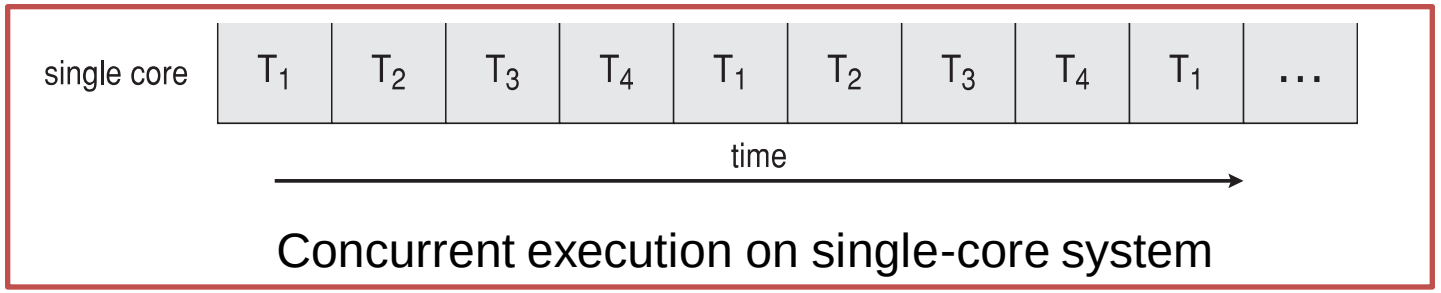


Introduction (6/6)

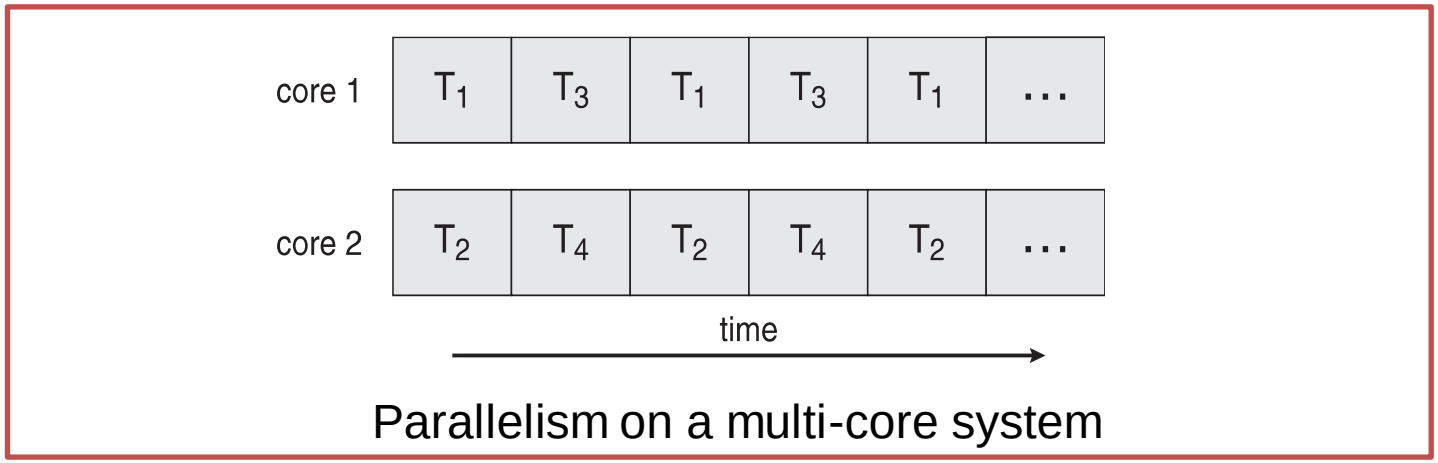
- Finally, most operating-system kernels are now multithreaded. Several threads operate in the kernel, and each thread performs a specific task, such as managing devices, managing memory, or interrupt handling.

Multicore Programming

- **Multithreaded** programming provides a mechanism for more efficient use of these **multicore** or **multiprocessor** systems.



Concurrent execution on single-core system



Parallelism on a multi-core system

Multithreading Models (1/6)

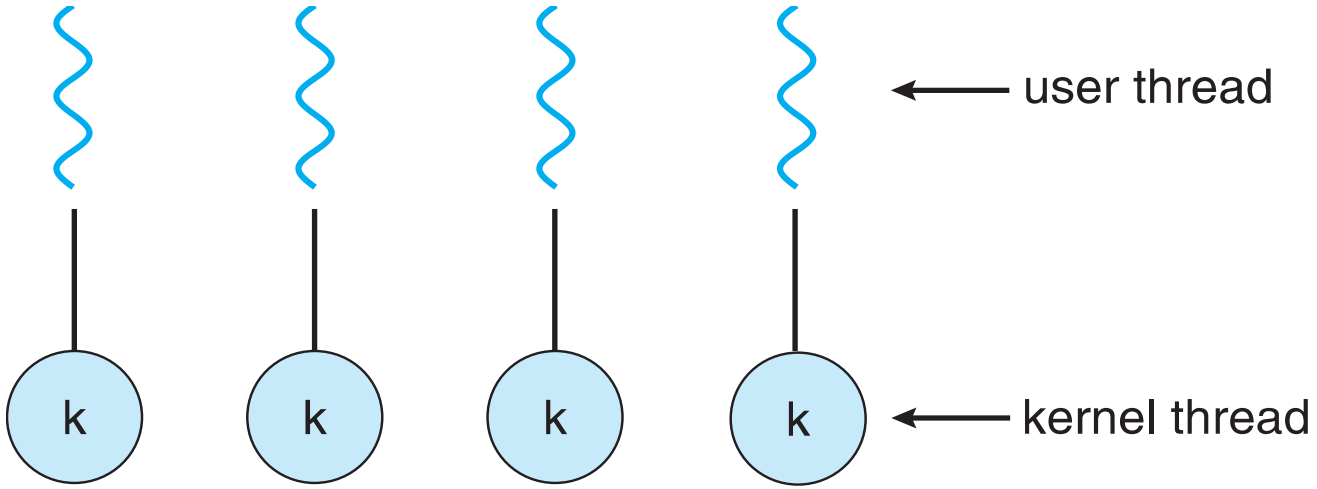
- Support for threads may be provided either at the user level, for **user threads**, or by the kernel, for **kernel threads**.
- User threads are supported above the kernel and are managed without kernel support, whereas kernel threads are supported and managed directly by the operating system.

Multithreading Models (2/6)

- Ultimately, a relationship must exist between user threads and kernel threads.
- We look at three common models:
  - the one-to-one model,
  - the many-to-one model, and
  - the many-to-many model.

Multithreading Models (3/6)

One-to-One model (1/3)



Multithreading Models (3/6)

One-to-One model (2/3)

- The one-to-one model maps each user thread to a kernel thread. It also allows multiple threads to run in parallel on multiprocessors.

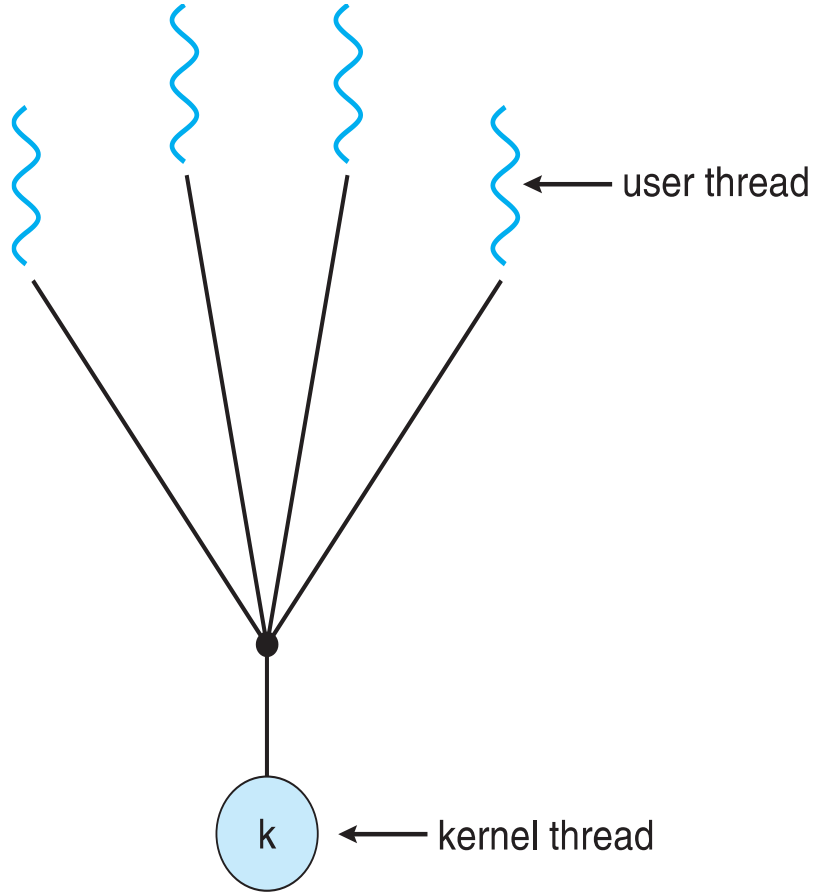
Multithreading Models (3/6)

One-to-One model (3/3)

- The only drawback to this model is that creating a user thread requires creating the corresponding kernel thread. Because the overhead of creating kernel threads can burden the performance of an application, most implementations of this model **restrict** the number of threads supported **by the system**. Linux, along with the family of Windows operating systems, implement the one-to-one model.

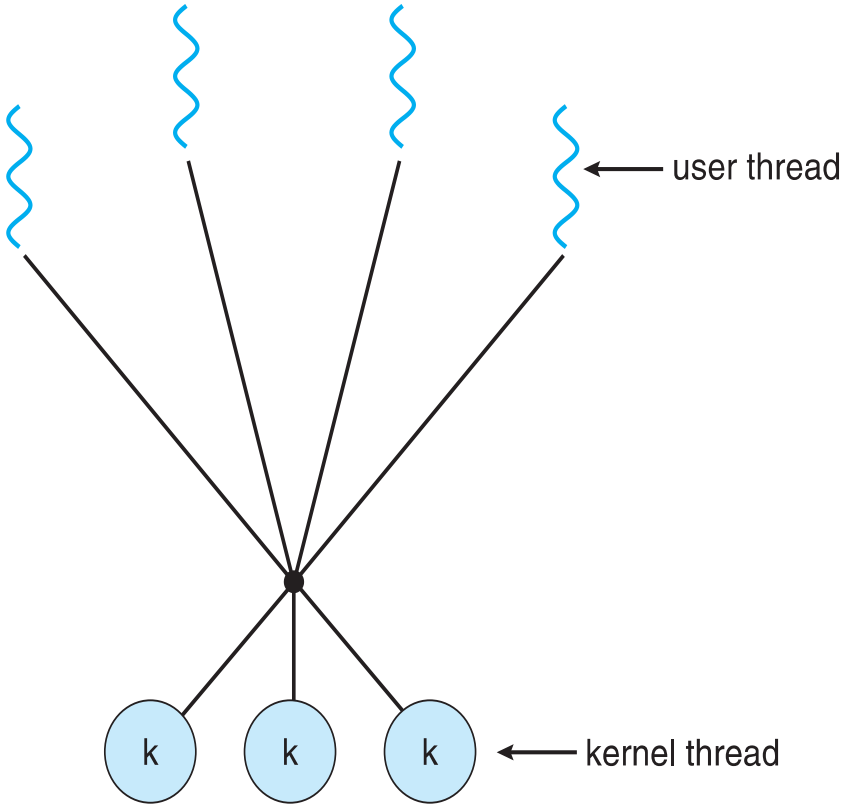
Multithreading Models (4/6)

Many-to-One model



Multithreading Models (5/6)

Many-to-Many model (1/2)



Multithreading Models (5/6)

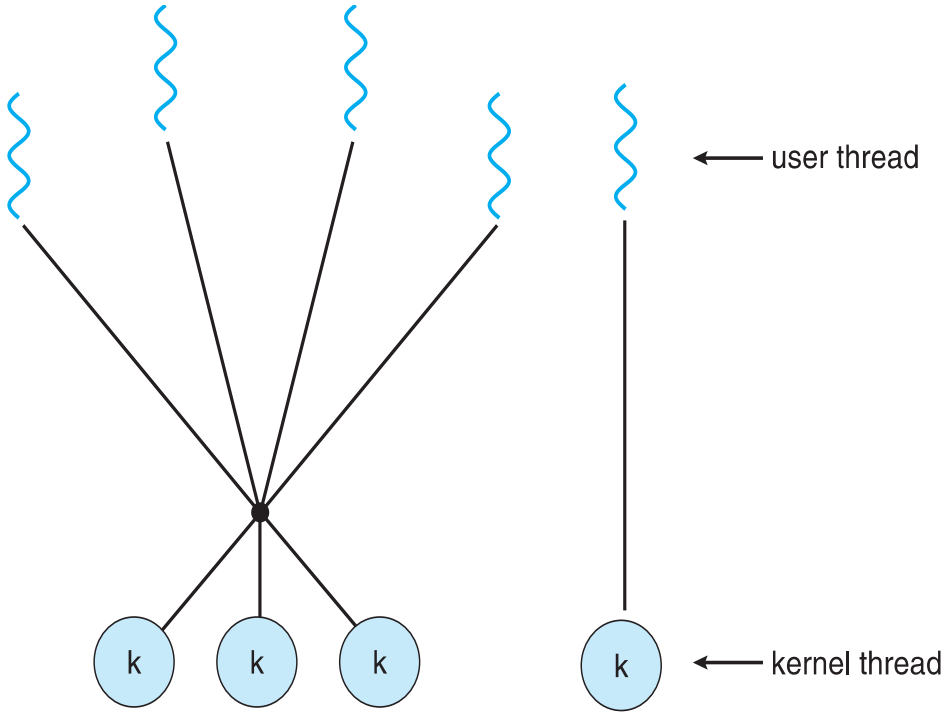
Many-to-Many model (2/2)

- Allows many user level threads to be mapped to many kernel threads.
- Allows the operating system to create a sufficient number of kernel threads.
- Allows the developer to create as many user threads as she wishes, it does not result in true concurrency, because the kernel can schedule only one thread at a time.

Multithreading Models (6/6)

Two-level model

- Similar to Many-to-Many, except that it allows a user thread to be **bound** to kernel thread.



Benefits of Threads

- **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.

Thread Libraries

- **Thread library** provides programmer with API for creating and managing threads.
- Three main thread libraries are in use today:
  - POSIX Pthreads,
  - Windows, and
  - Java.

Thread Libraries

- **Thread library** provides programmer with API for creating and managing threads.
- Three main thread libraries are in use today:
  - POSIX Pthreads,
  - Windows, and
  - Java.

Thank You

Dr. Ahmed Hagag

ahagag@fci.bu.edu.eg

https://www.youtube.com/channel/UCzYgAyyZTLfnLFjQexOKxbQ

https://www.facebook.com/ahmed.hagag.71/