Zakho Technical Institute / IT

Operation System -

Theory

5. Thread in Operating System

Lecturer:

Sipan M. Hameed

www.sipan.dev

2023-2024

Contents

Overview

A thread is a basic unit of CPU utilization; it comprises a thread ID, a program counter (PC), 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. A traditional process has a single thread of control. If a process has multiple threads of control, it can perform more than one task at a time. Figure 4.1 illustrates the difference between a traditional single-threaded process and a multithreaded process.

Motivation

Most software applications that run on modern computers and mobile devices are multithreaded. An application typically is implemented as a separate process with several threads of control. Below we highlight a few examples of multithreaded applications:

- An application that creates photo thumbnails from a collection of images may use a separate thread to generate a thumbnail from each separate image.
- A web browser might have one thread display images or text while another thread retrieves data from the network.
- 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.

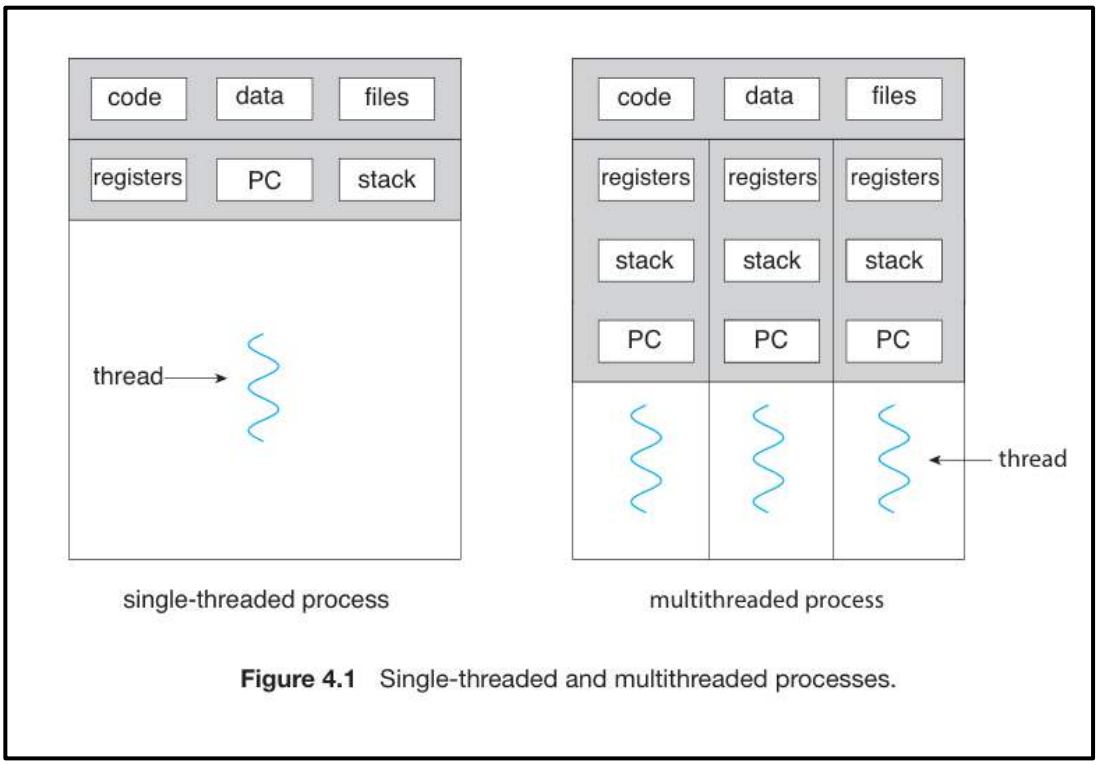

Figure 4.1 Single-threaded and multithreaded processes.

Benefits of multithreaded programming

The benefits of multithreaded programming can be broken down into four major categories:

1. Responsiveness.
2. Resource sharing.
3. Economy.
4. Scalability.

Multicore Programming

Earlier in the history of computer design, in response to the need for more computing performance, single-CPU systems evolved into multi-CPU systems. A later, yet similar, trend in system design is to place multiple computing cores on a single processing chip where each core appears as a separate CPU to the operating system.

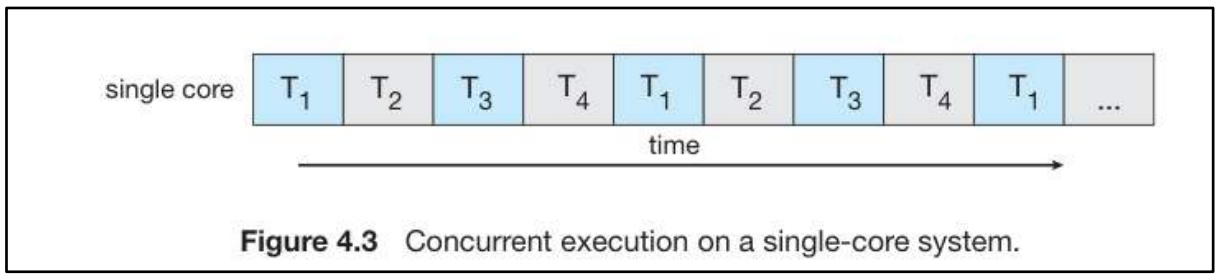

Figure 4.3 Concurrent execution on a single-core system.

We refer to such systems as multicore, and multithreaded programming provides a mechanism for more efficient use of these multiple computing cores and improved concurrency. Consider an application with four threads. On a system with a single computing core, concurrency merely means that the execution of the threads will be interleaved over time (Figure 4.3), because the processing core is capable of executing only one thread at a time. On a system with multiple cores, however, concurrency means that some threads can run in parallel, because the system can assign a separate thread to each core (Figure 4.4).

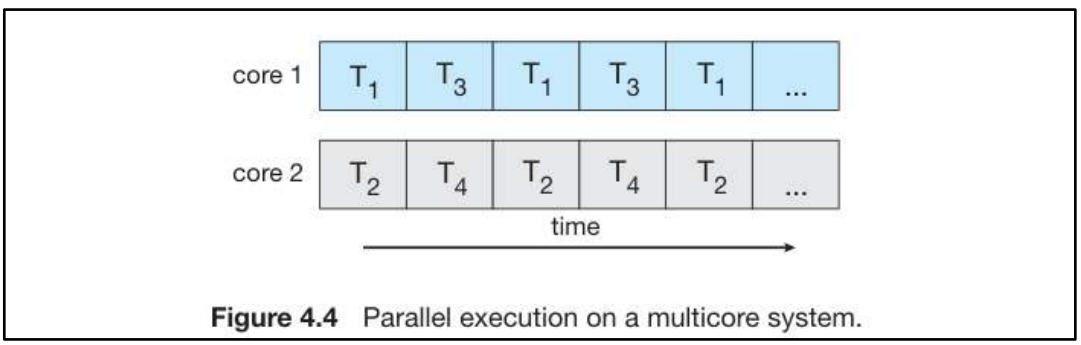

Figure 4.4 Parallel execution on a multicore system.

Programming Challenges in multicore systems

1. Identifying tasks.
2. Balance.
3. Data splitting.
4. Data dependency.
5. Testing and debugging.

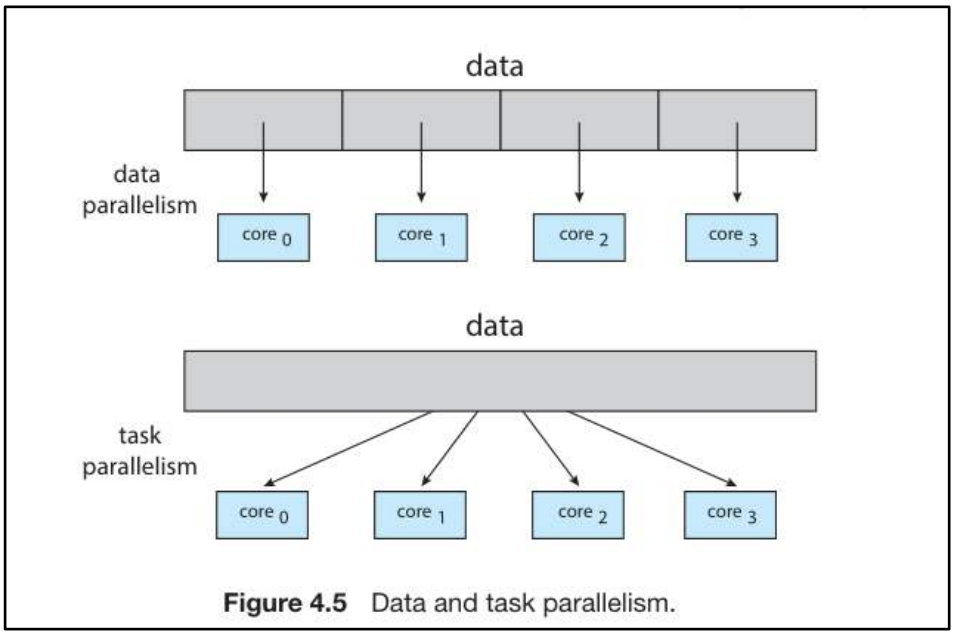

Figure 4.5 Data and task parallelism.

Types of Parallelism

1. Data parallelism
2. Task parallelism

Multithreading Models

Our discussion so far has treated threads in a generic sense. However, 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. Virtually all contemporary operating systems—including Windows, Linux, and macOS— support kernel threads.

Ultimately, a relationship must exist between user threads and kernel threads, as illustrated in Figure 4.6. In this section, we look at three common ways of establishing such a relationship: the many-to-one model, the one-to-
one model, and the many-to-many model.

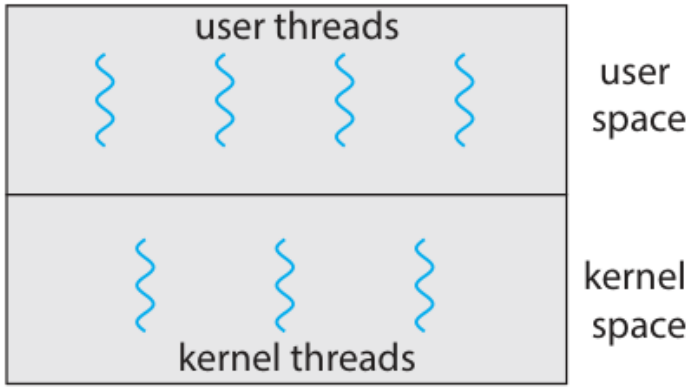

Figure 4.6 User and kernel threads.

1. *Many-to-One Model*

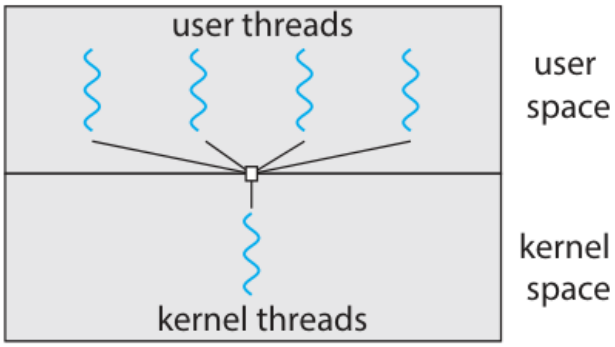

Figure 4.7 Many-to-one model.

2. *One-to-One Model*

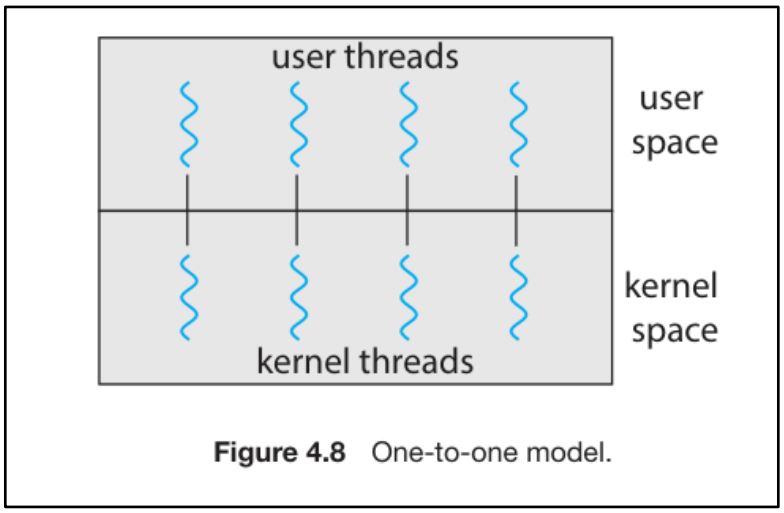

Figure 4.8 One-to-one model.

3. *Many-to-Many Model*

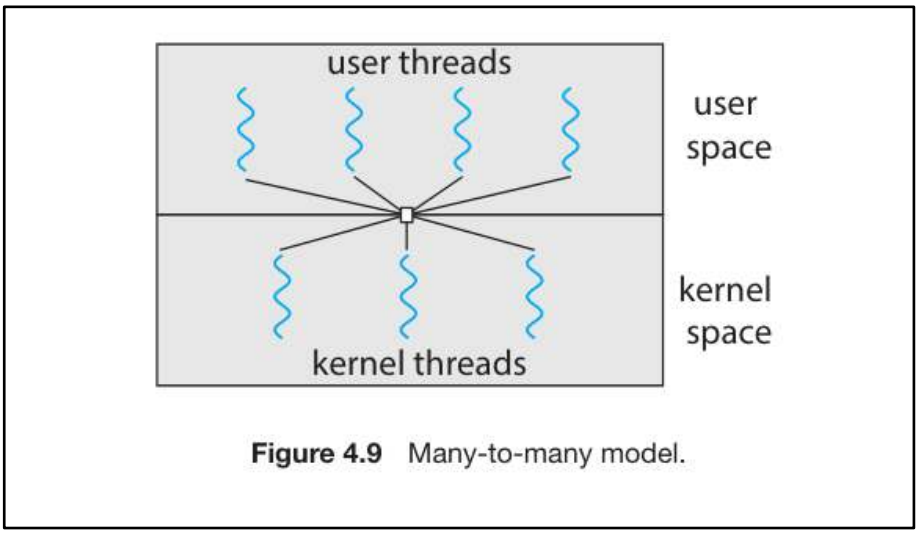

Figure 4.9 Many-to-many model.

Thread Libraries

1. Pthreads
2. Windows Threads
3. Java Threads