



Operating Systems

Internals and Design Principles

NINTH EDITION

William Stallings



Chapter 4 Threads

Learning Objectives

After studying this chapter, you should be able to:

- Understand the distinction between process and thread
- Describe the basic design issues for threads
- Explain the difference between user level threads and kernel level threads
- Describe the thread management facility in Windows 7, Solaris, and Linux

Processes and Threads

- The concept of a process embodies two characteristics as below:

Resource Ownership

Process includes a virtual address space to hold the process image and other resources (e.g., main memory, I/O devices, and files)

- The OS performs a protection function to prevent unwanted interference between processes with respect to resources

Scheduling/Execution

The execution of a process may be interleaved with other processes

- A process has an execution state (Running, Ready, etc.) and a dispatching priority
- It is the entity that is scheduled and dispatched by the OS

Processes and Threads

- The two characteristics are independent and could be created independently by the OS
- The unit of dispatching is referred to as a *thread* or *lightweight process*
- The unit of resource ownership is referred to as a *process* or *task*
- **Multithreading** - The ability of an OS to support multiple, concurrent paths of execution within a single process

Single Threaded Approaches

- A single thread of execution per process is referred to as a single-threaded approach
- The concept of a thread was not known
- MS-DOS supports only a single user process
- Old versions of UNIX supports multiple user processes

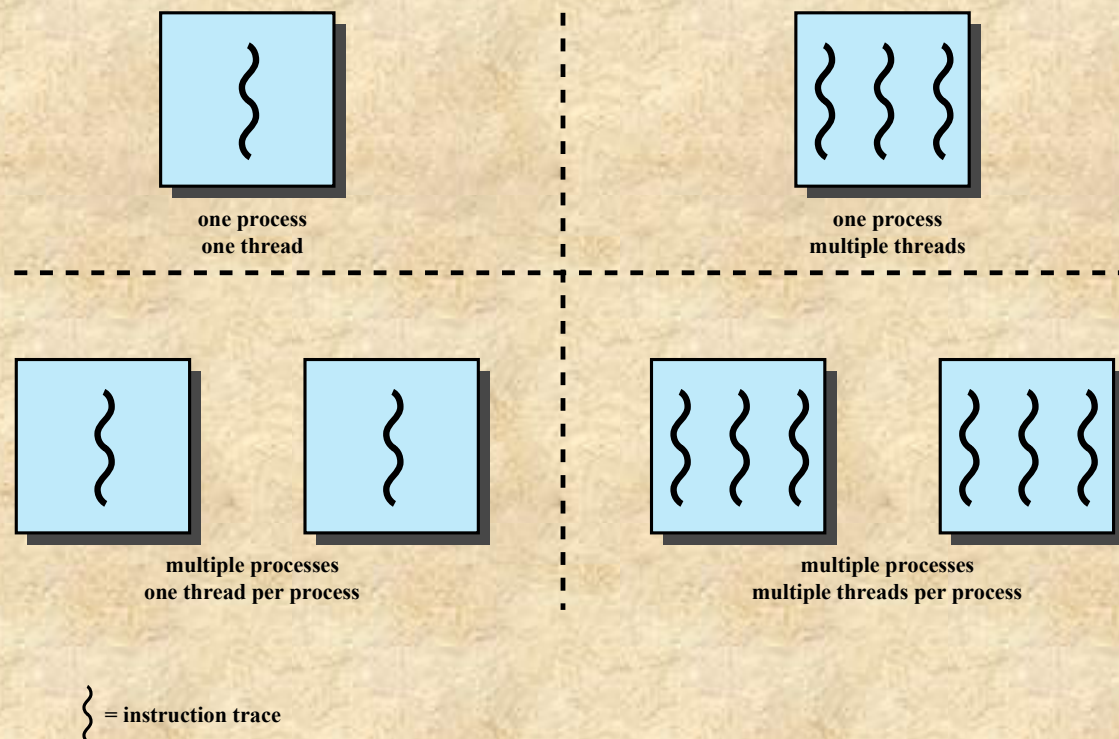


Figure 4.1 Threads and Processes

Multithreaded Approaches

- The right half of Figure 4.1 depicts multithreaded approaches
- A Java run-time environment is an example of a system of one process with multiple threads
- Windows, Solaris, and Linux allow multiple processes, each of which supports multiple threads

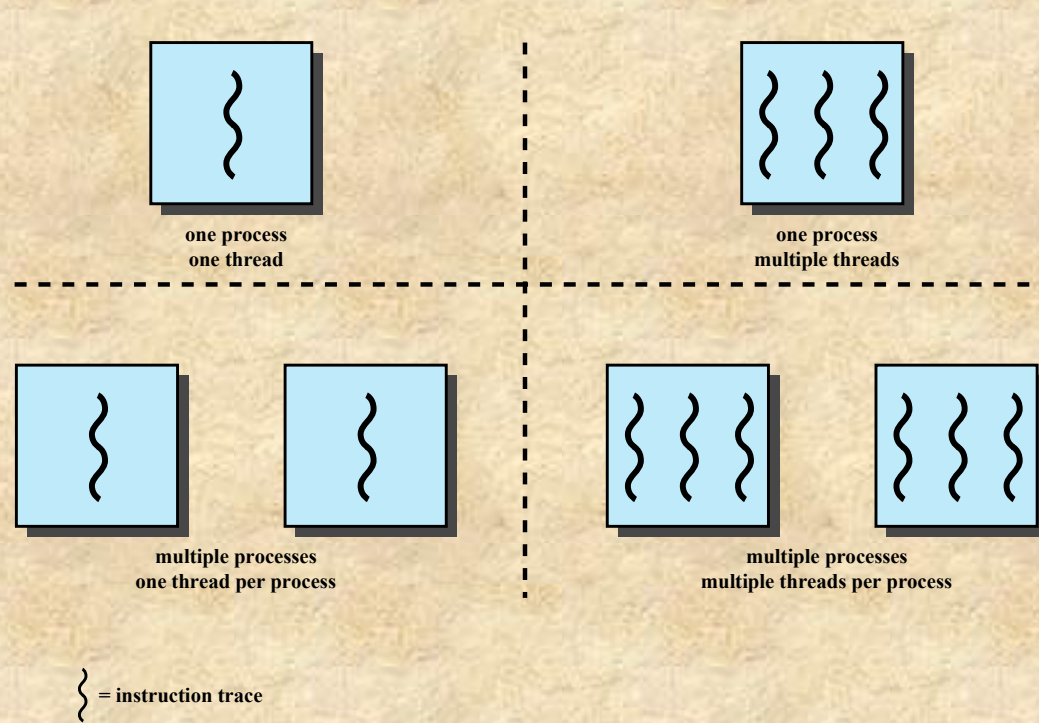


Figure 4.1 Threads and Processes

Process

- Defined as “the unit of resource allocation and a unit of protection” in a multithreaded environment
- Associated with processes:
 - A virtual address space that holds the process image
 - Protected access to:
 - Processors
 - Other processes (for interprocess communication)
 - Files
 - I/O resources (devices and channels)

One or More Threads in a Process

Each thread has:

- An execution state (Running, Ready, etc.)
- A saved thread context when not running
- An execution stack
- Some per-thread static storage for local variables
- Access to the memory and resources of its processes, shared with all other threads in that process

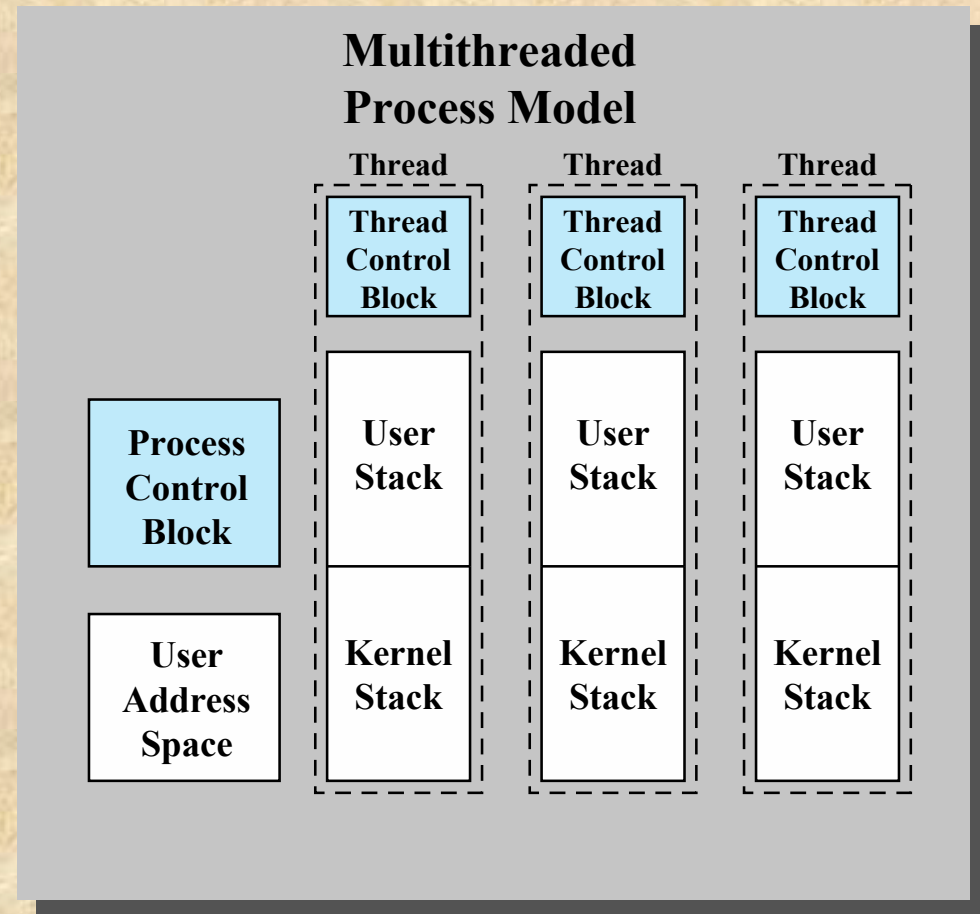
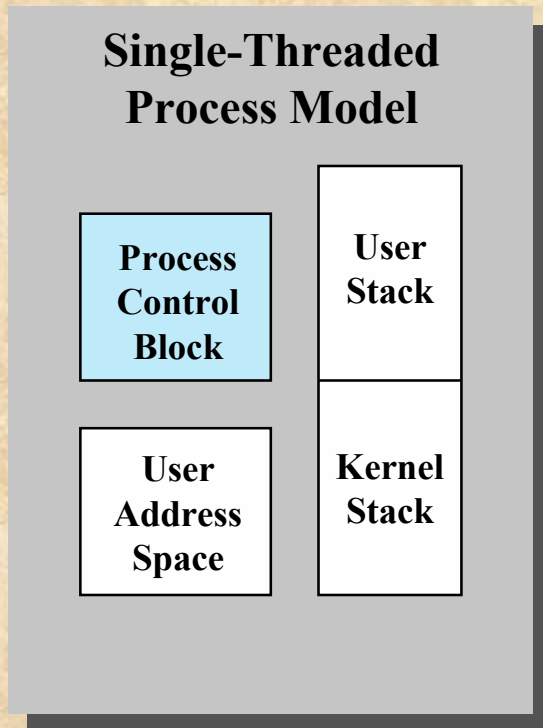
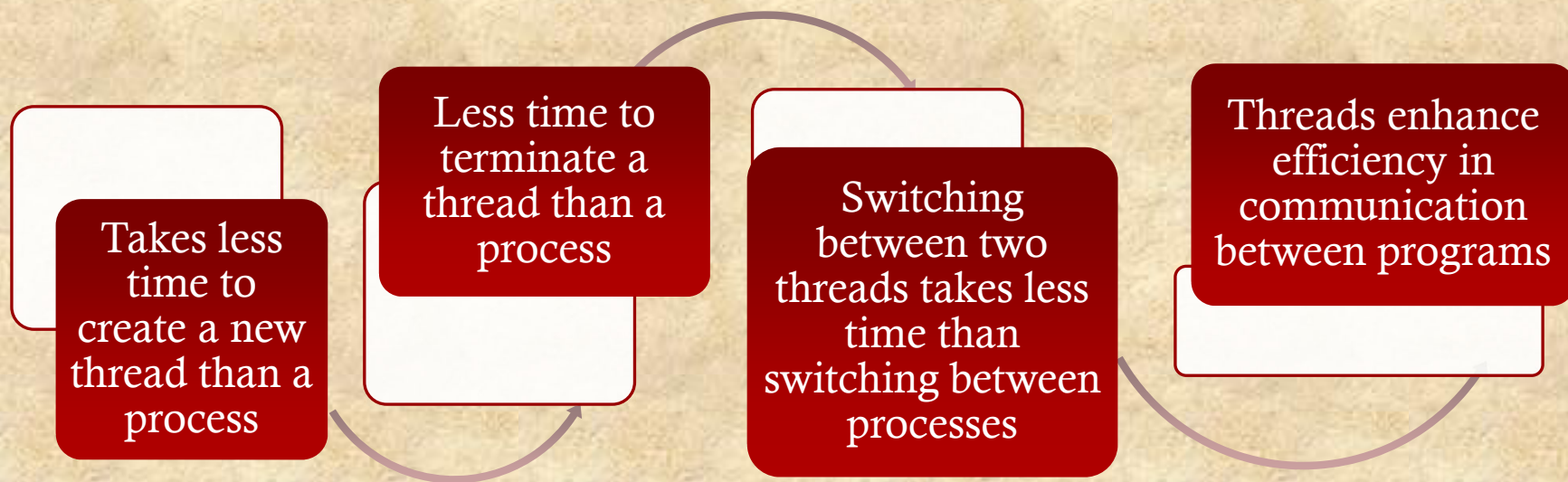


Figure 4.2 Single Threaded and Multithreaded Process Models

Key Benefits of Threads



Thread Use in a Single-User System

- Foreground and background work
- Asynchronous processing
- Speed of execution
- Modular program structure

Threads

- In an OS that supports threads, scheduling and dispatching is done on a thread basis
- Most of the state information dealing with execution is maintained in thread-level data structures
- Actions that affect all threads in a process
 - Suspending a process involves suspending all threads of the process, because all the threads share the same address space
 - Termination of a process terminates all threads within the process

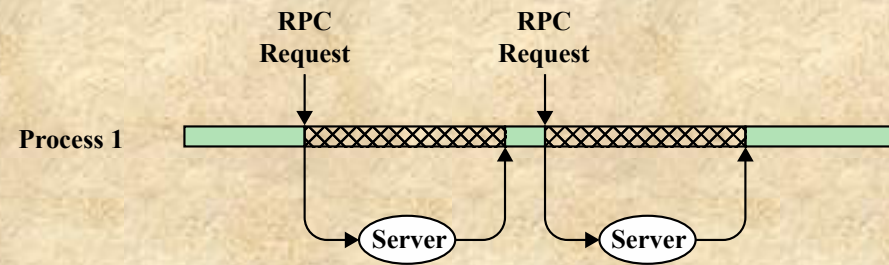
Thread Execution States

The key states for a thread are:

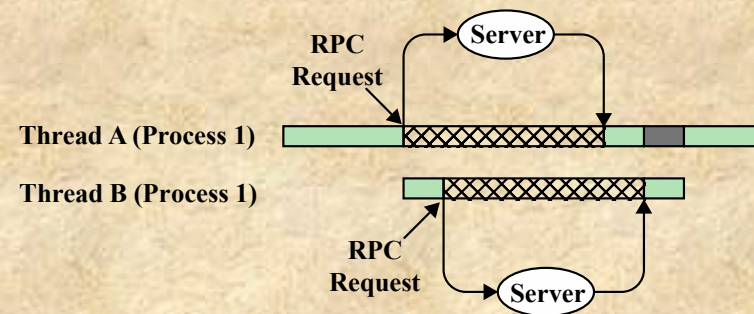
- Running
- Ready
- Blocked

Thread operations associated with a change in thread state are:

- Spawn
- Block
- Unblock
- Finish



(a) RPC Using Single Thread



(b) RPC Using One Thread per Server (on a uniprocessor)




-  Blocked, waiting for response to RPC
-  Blocked, waiting for processor, which is in use by Thread B
-  Running

Figure 4.3 Remote Procedure Call (RPC) Using Threads

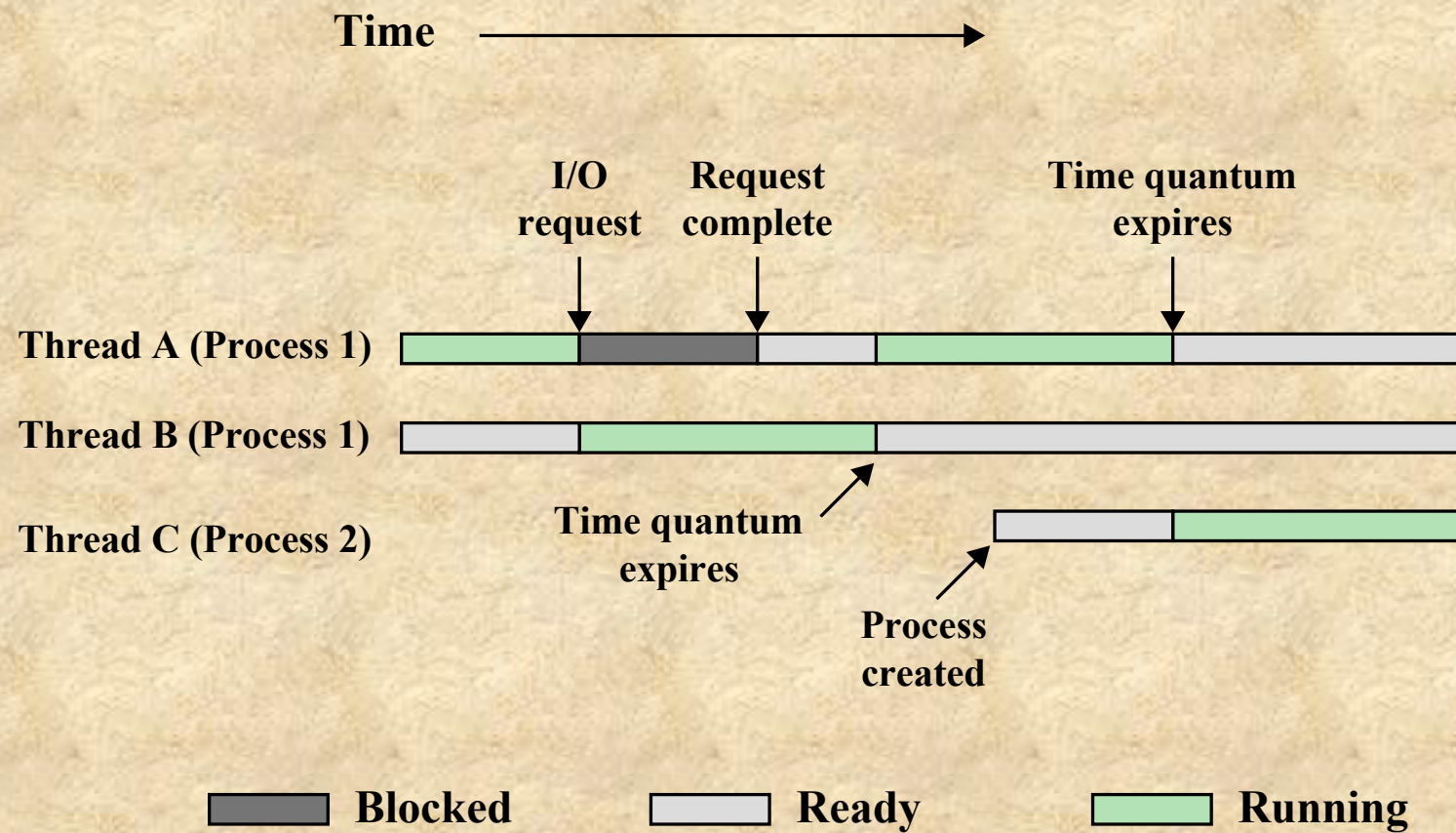


Figure 4.4 Multithreading Example on a Uniprocessor

Thread Synchronization

- It is necessary to synchronize the activities of the various threads
 - All threads of a process share the same address space and other resources
 - Any alteration of a resource by one thread affects the other threads in the same process