Chapter 4
Threads
Ninth Edition
By William Stallings
Processes and Threads

Resource Ownership

Process includes a virtual address space to hold the process image
- The OS performs a protection function to prevent unwanted interference between processes with respect to resources

Scheduling/Execution

Follows an execution path that may be interleaved with other processes
- A process has an execution state (Running, Ready, etc.) and a dispatching priority, and is the entity that is scheduled and dispatched by the OS
Processes and Threads

- 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, in which the concept of a thread is not recognized, is referred to as a single-threaded approach.

- MS-DOS is an example.

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.

Figure 4.1 Threads and Processes
Process

- Defined in a multithreaded environment as “the unit of resource allocation and a unit of protection”

- 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

- Takes less time to create a new thread than a process
- Less time to terminate a thread than a process
- Switching between two threads takes less time than switching between processes
- Threads enhance efficiency in communication between programs
Thread Use in a Single-User System

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

- Suspending a process involves suspending all threads of the process.
- 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
Figure 4.3 Remote Procedure Call (RPC) Using Threads

(a) RPC Using Single Thread

(b) RPC Using One Thread per Server (on a uniprocessor)
Figure 4.4  Multithreading Example on a Uniprocessor
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