第12讲 线程

Thread（线程）

- An execution state (running, ready, etc.)
- Saved thread context when not running.
- Has an execution stack.
- Some per-thread static storage for local variables.
- Access to the memory and resources of its process.
  - all threads of a process share this.

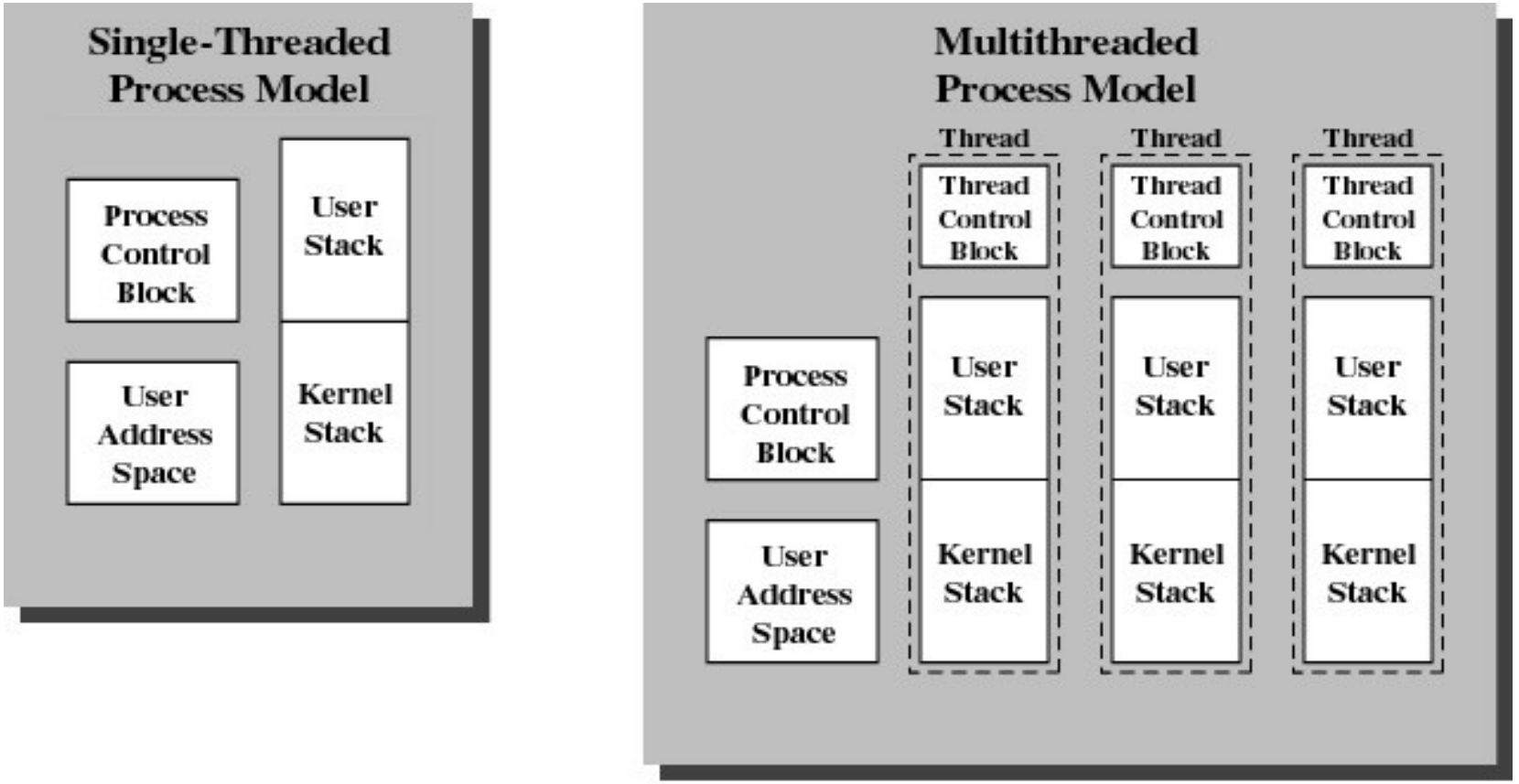

Figure 4.2 Single Threaded and Multithreaded Process Models

Benefits of Threads

- Takes less time to create a new thread than a process.
- Less time to terminate a thread than a process.
- Less time to switch between two threads within the same process.
- Since threads within the same process share memory and files, they can communicate with each other without invoking the kernel.

Threads

- Suspending a process involves suspending all threads of the process
  - since all threads share the same address space.
- Termination of a process, terminates all threads within the process.

Multithreading

- Operating system supports multiple threads of execution within a single process.
- MS-DOS supports a single thread.
- UNIX supports multiple user processes but only supports one thread per process.
- Windows 2000, Solaris, Linux, Mach, and OS/2 support multiple threads.

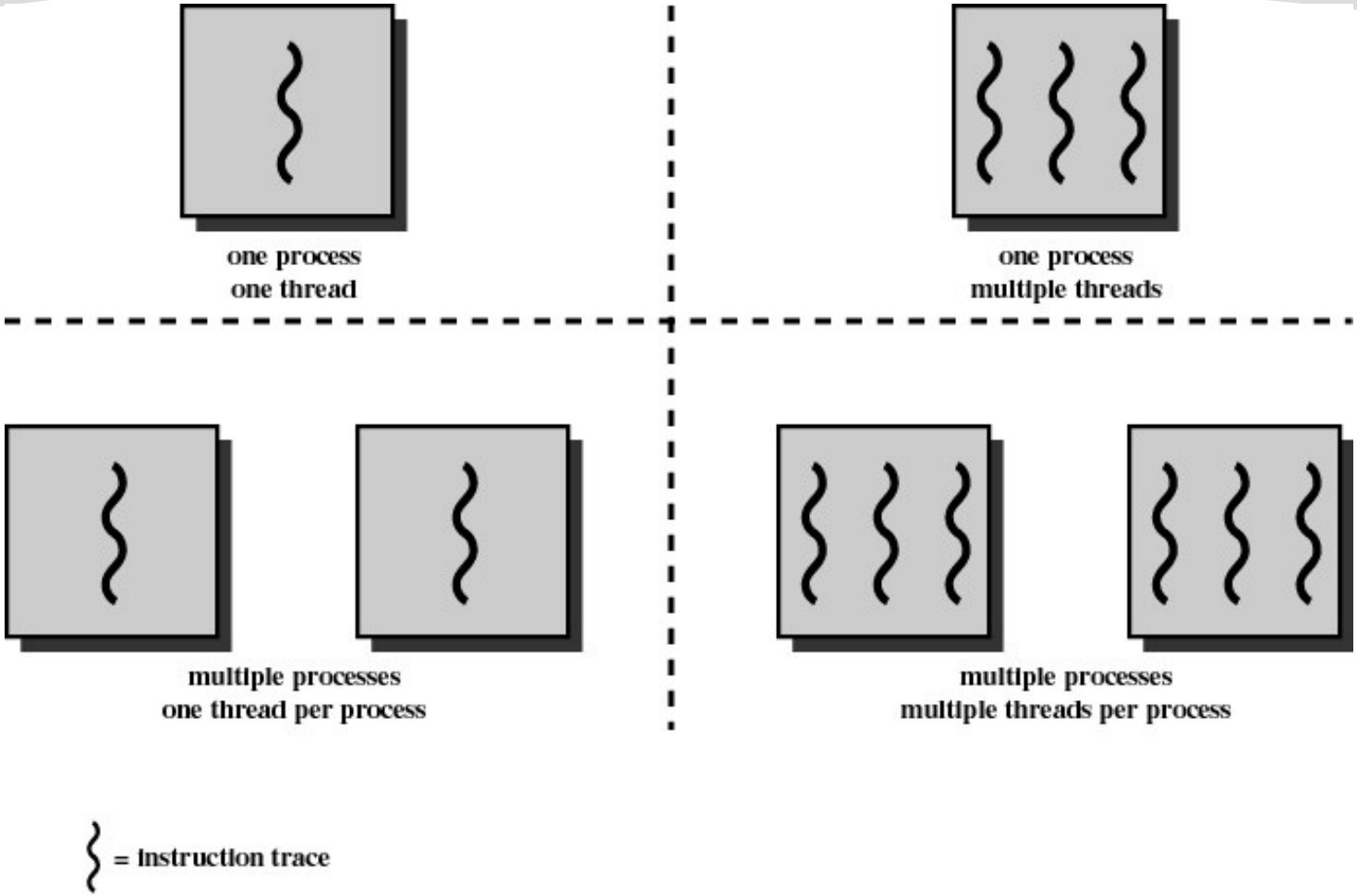

Figure 4.1 Threads and Processes [ANDE97]

Thread States

- Key states for a thread: Running, Ready, Blocked.
- Operations associated with a change in thread state.
  - Spawn(派生),Spawn another thread
  - Block
  - Unblock
  - Finish

User-Level Threads

- All thread management is done by the application.
- The kernel is not aware of the existence of threads.
- 描述此类线程的数据结构以及控制此类线程的原语在核外子系统中实现。

Kernel-Level Threads

- W2K, Linux, and OS/2 are examples of this approach.
- Kernel maintains context information for the process and the threads.
- Scheduling is done on a thread basis.
- 描述此类线程的数据结构以及控制此类线程的原语在核心子系统中实现。

Combined Approaches

- Example is Solaris.
- Thread creation done in the user space.
- Bulk （大批） of scheduling and synchronization of threads done in the user space.

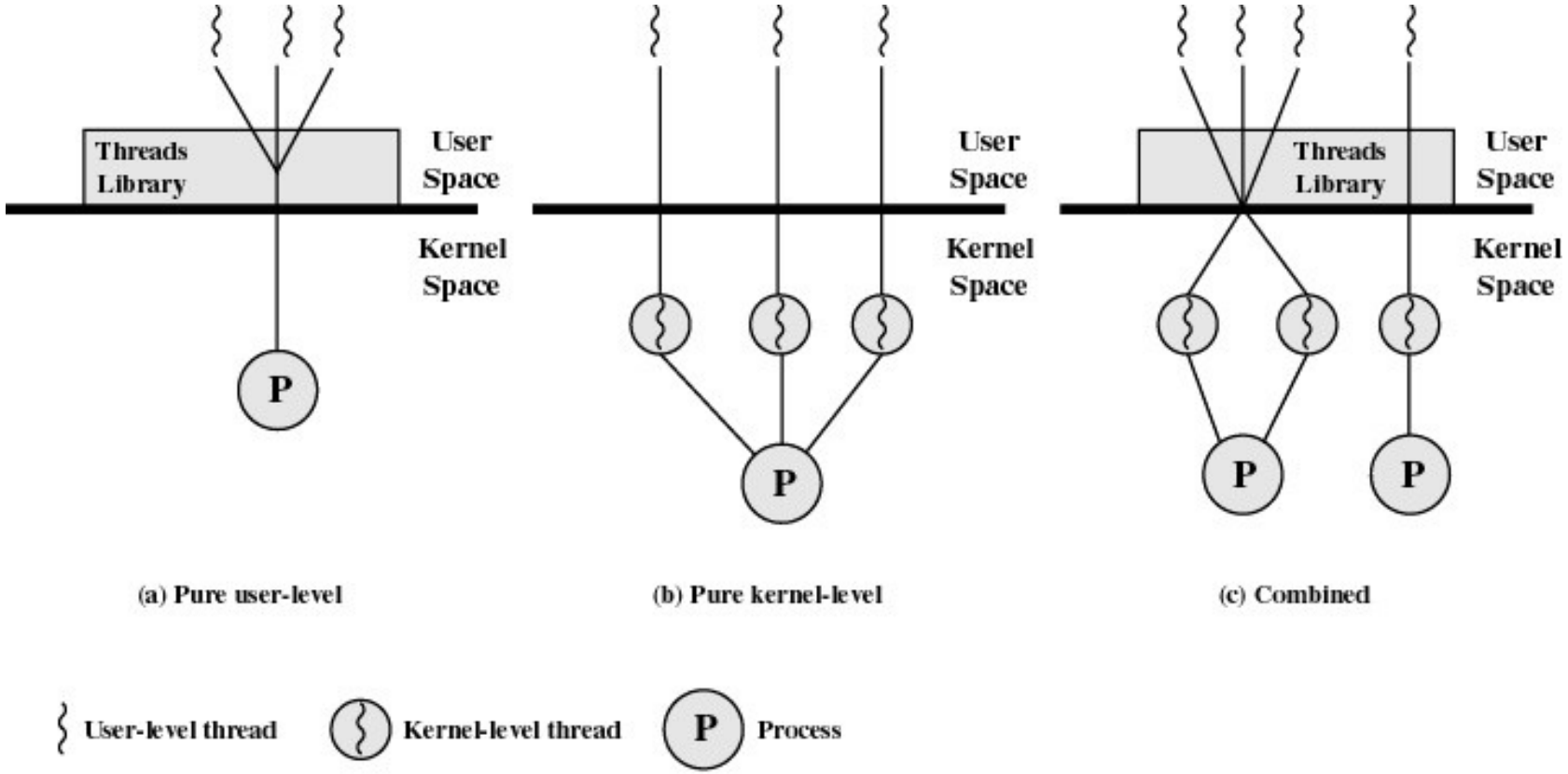

Figure 4.6 User-Level and Kernel-Level Threads

Relationship Between Threads and Processes

Threads:Process	Description	Example Systems
1:1	Each thread of execution is a unique process with its own address space and resources.	Traditional UNIX implementations
M:1	A process defines an address space and dynamic resource ownership. Multiple threads may be created and executed within that process.	Windows NT, Linux, Solaris, OS/2, OS/390, MACH

Relationship Between Threads and Processes

Threads:Process	Description	Example Systems
1:M	A thread may migrate from one process environment to another. This allows a thread to be easily moved among distinct systems.	Ra (Clouds), Emerald
M:M	Combines attributes of M:1 and 1:M cases	TRIX

系统实例分析

- UNIX 的进程管理
- Linux 的进程管理