**Threads**

- A *thread* (or *lightweight process*) is a basic unit of CPU utilization; it consists of:
  - Stack space
  - PCB in the kernel

- A thread shares with its peer threads its:
  - Text/Code section
  - Data section
  collectively known as a *task*.

- A traditional or *heavyweight* process is equal to a task with one thread

**Threads (Cont.)**

- In a multiple threaded task, while one server thread is blocked and waiting, a second thread in the same task can run.
- Applications that require sharing a common buffer (i.e., producer-consumer) benefit from thread utilization.
- Threads provide a mechanism that allows sequential processes to make blocking system calls while also achieving parallelism.

- Types of threads:
  - Kernel-supported threads.
  - User-level threads; supported above the kernel, via a set of library calls at the user level.
  - Anderson's threads: Like Kernel-supported threads, but sends events that influence user-level scheduling to the task, rather than interpret these events on its own.

**Single and Multithreaded Processes**

- Many-to-One
  - Many user-level threads are mapped to a single kernel thread.
  - Used on systems that do not support kernel threads.
  - Example: Project Andrew of CMU.

- One-to-One
  - Each user-level thread is mapped to a kernel thread.
  - Examples: Windows95, Windows98 and OS/2.

- Many-to-Many Model
  - Allows many user level threads to be mapped to many kernel threads.
  - Allows the operating system to create a sufficient number of kernel threads.
  - Hybrid approach, implements both user-level and kernel-supported threads.
Multithreading Models (Cont.)

One-to-one

Many-to-one

Many-to-many

Threads Support in Solaris 2

- Solaris 2 is a version of UNIX with a support of threads at the kernel and user levels.
- LWP – intermediate level between user-level threads and kernel-level threads.

- Resource needs of thread types:
  - Thread of the kernel: Small data structure and a stack; thread switching does not require changing memory access information – relatively fast.
  - LWP: PCB with register data, accounting and memory information; switching between LWPs is relatively slow.
  - User-level thread: Only needs stack and program counter; no kernel involvement means fast switching. Kernel only sees the LWPs.

Solaris 2 Threads