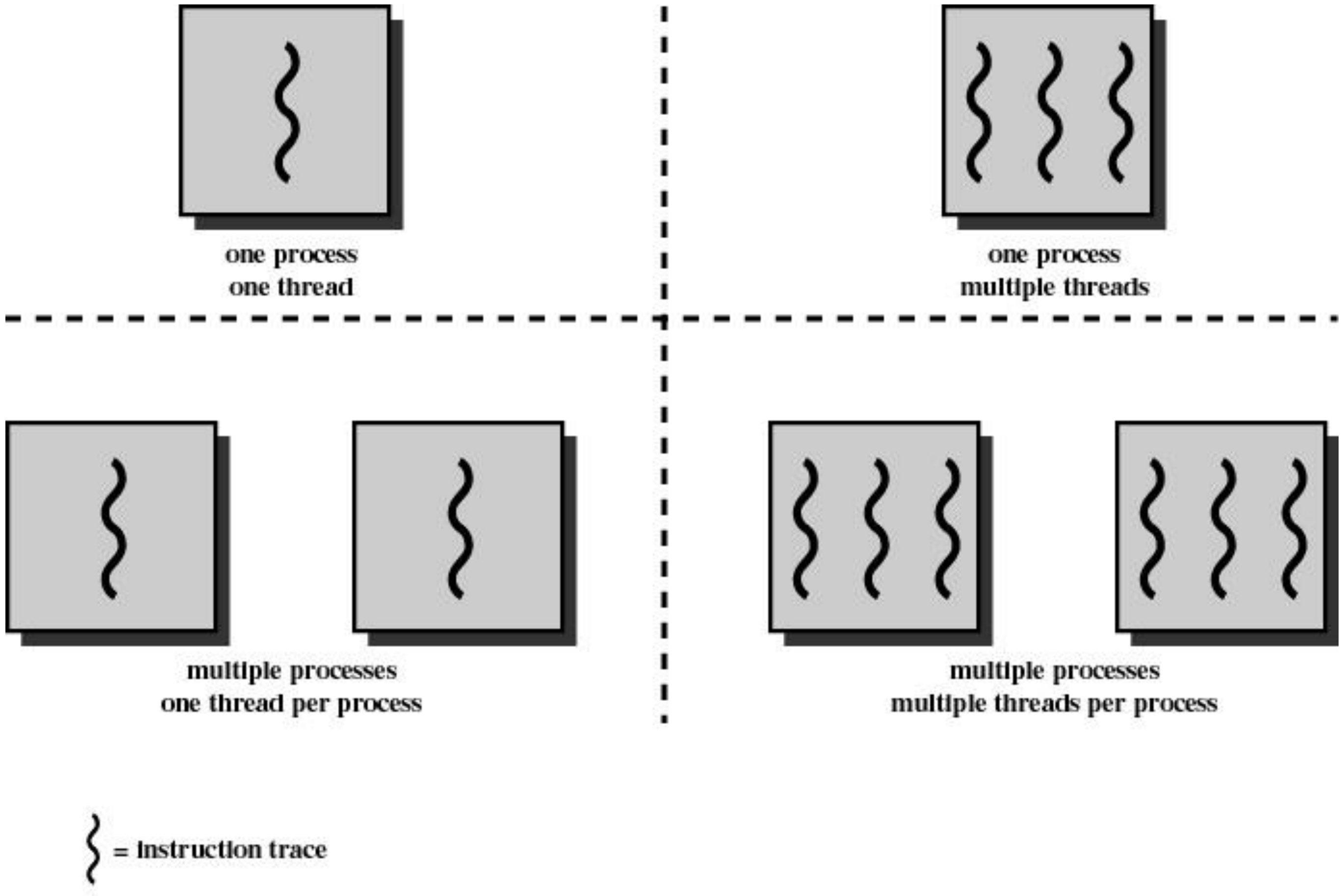
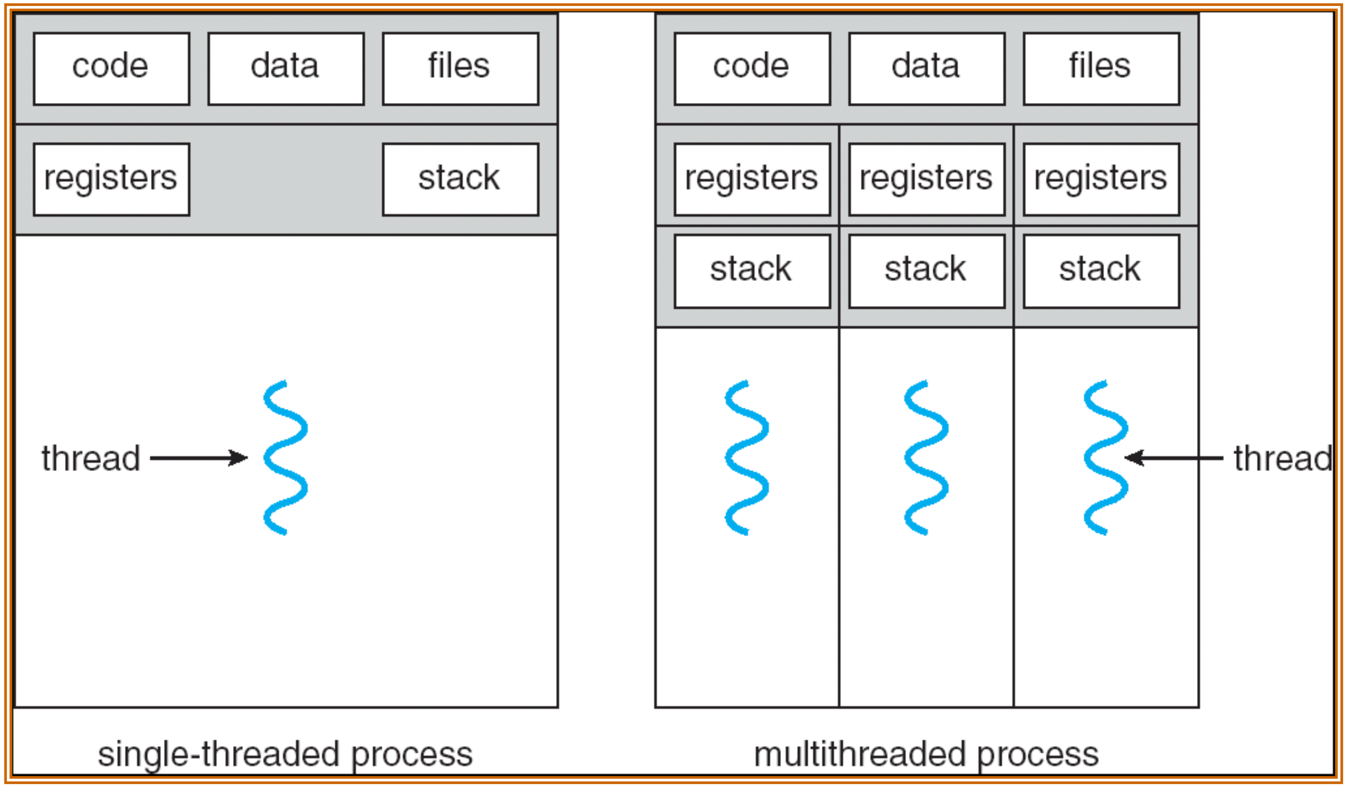
Threads

Chapter 4

Single and Multithreaded Processes



Single and Multithreaded Processes



Benefits

- Resource Sharing
- Since threads within the same process share memory and files, they can communicate with each other without invoking the kernel
- Utilization of MP Architectures
- 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

User and Kernel Threads

- In general, threads could be realized in one of two ways:
  - User-Level Threads: (Pthreads library)
    - All thread management is done by the application
    - The kernel is not aware of the existence of threads
  - Kernel-Level Threads: (W2K, Linux, and OS/2)
    - Kernel maintains context information for the process and the threads
    - Scheduling is done on a thread basis

Combined Approaches

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

Kernel Threads

Examples

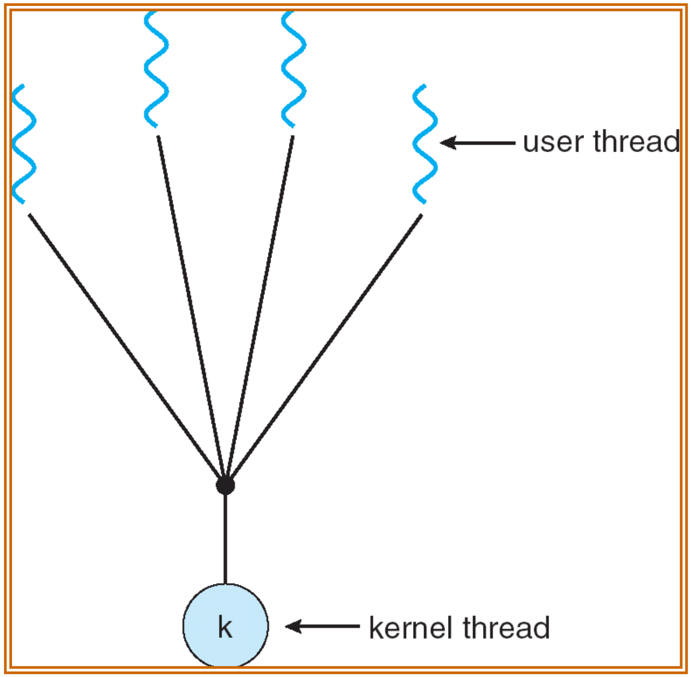
- Windows XP/2000
- Solaris
- Linux
- Tru64 UNIX
- Mac OS X

Multithreading Models

- Mapping user threads to kernel threads:
  - Many-to-One
  - One-to-One
  - Many-to-Many

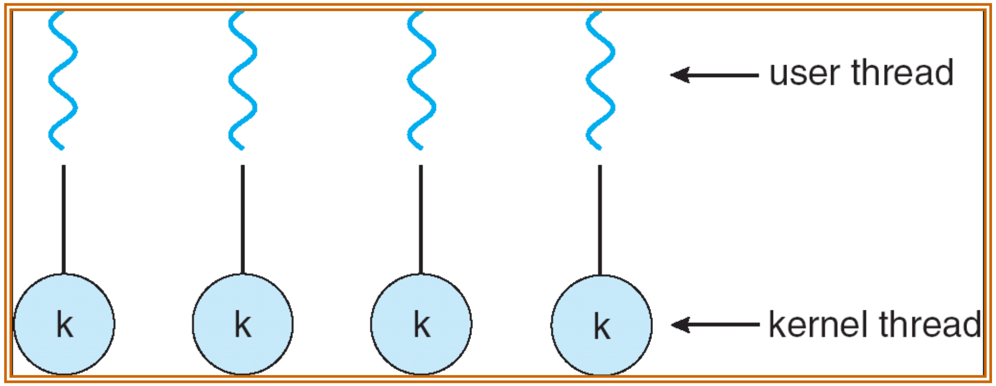
Many-to-One

- Many user-level threads mapped to single kernel thread
- Examples:
  - – Solaris Green Threads
  - – GNU Portable Threads



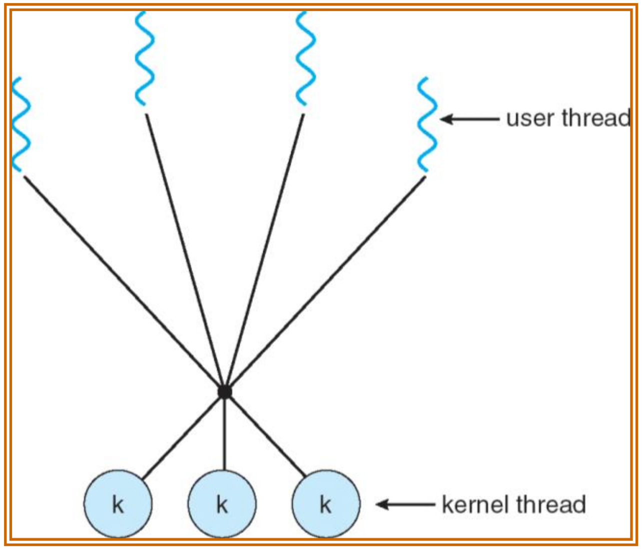
One-to-One

- Each user-level thread maps to kernel thread
- Examples
  - Windows NT/XP/2000
  - Linux
  - Solaris 9 and later



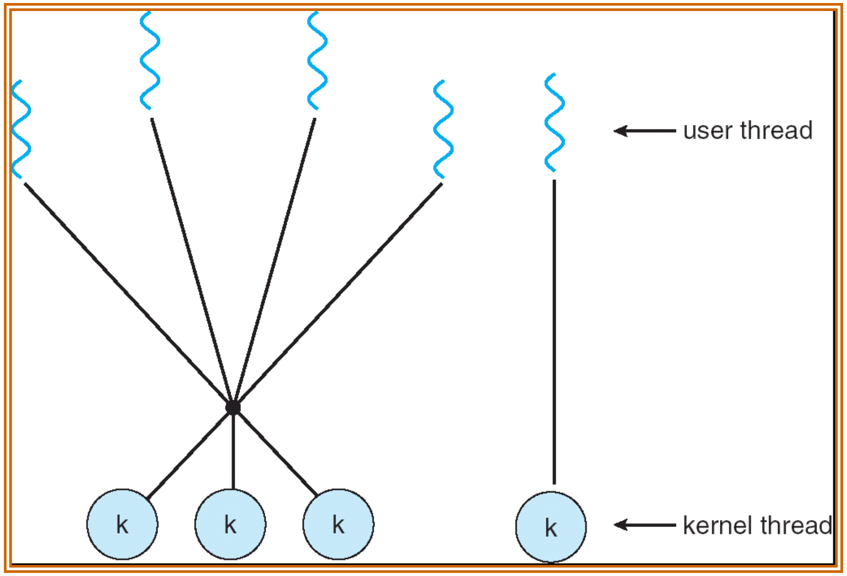
Many-to-Many Model

- Allows many user level threads to be mapped to many kernel threads
- Allows the operating system to create a sufficient number of kernel threads
- Solaris prior to version 9
- Windows NT/2000 with the *ThreadFiber* package



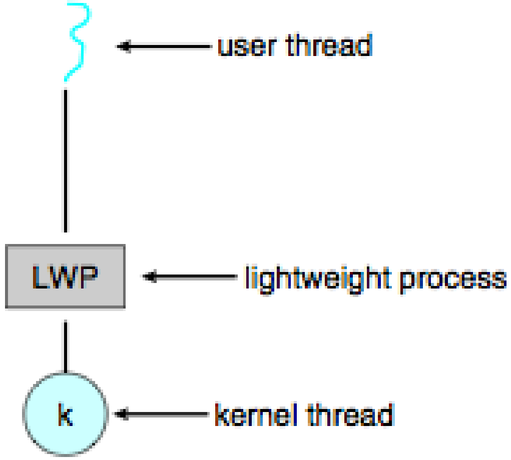
Two-level Model

- Similar to M:M, except that it allows a user thread to be bound to kernel thread
- Examples
  - – IRIX
  - – HP-UX
  - – Tru64 UNIX
  - – Solaris 8 and earlier



Scheduler Activations

- Both M:M and Two-level models require communication to maintain the appropriate number of kernel threads allocated to the application
- Scheduler activations provide **upcalls** - a communication mechanism from the kernel to the thread library
- This communication allows an application to maintain the correct number of kernel threads



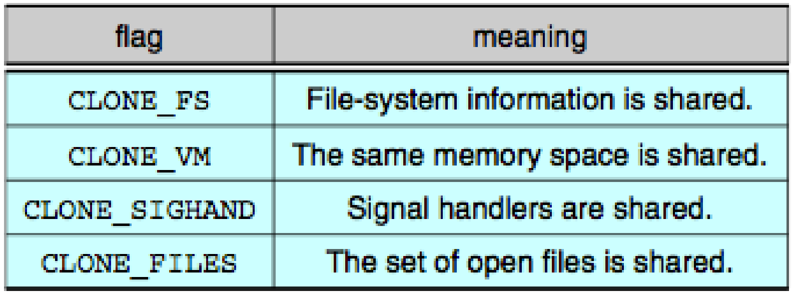
© 2015 Prof. Amr El-Kadi

Pthreads

- A POSIX standard (IEEE 1003.1c) API for thread creation and synchronization
- API specifies behavior of the thread library, implementation is up to development of the library
- Common in UNIX operating systems (Solaris, Linux, Mac OS X)

Linux Threads

- Linux refers to them as *tasks* rather than *threads*
- Thread creation is done through **clone()** system call
- **clone()** allows a child task to share the address space of the parent task (process)



flag	meaning
CLONE_FS	File-system information is shared.
CLONE_VM	The same memory space is shared.
CLONE_SIGHAND	Signal handlers are shared.
CLONE_FILES	The set of open files is shared.