THREADS

Chapter outline

- Overview +
- Multithreading Models +
- Threading Issues +
- Pthreads +
- Examples of Threads +

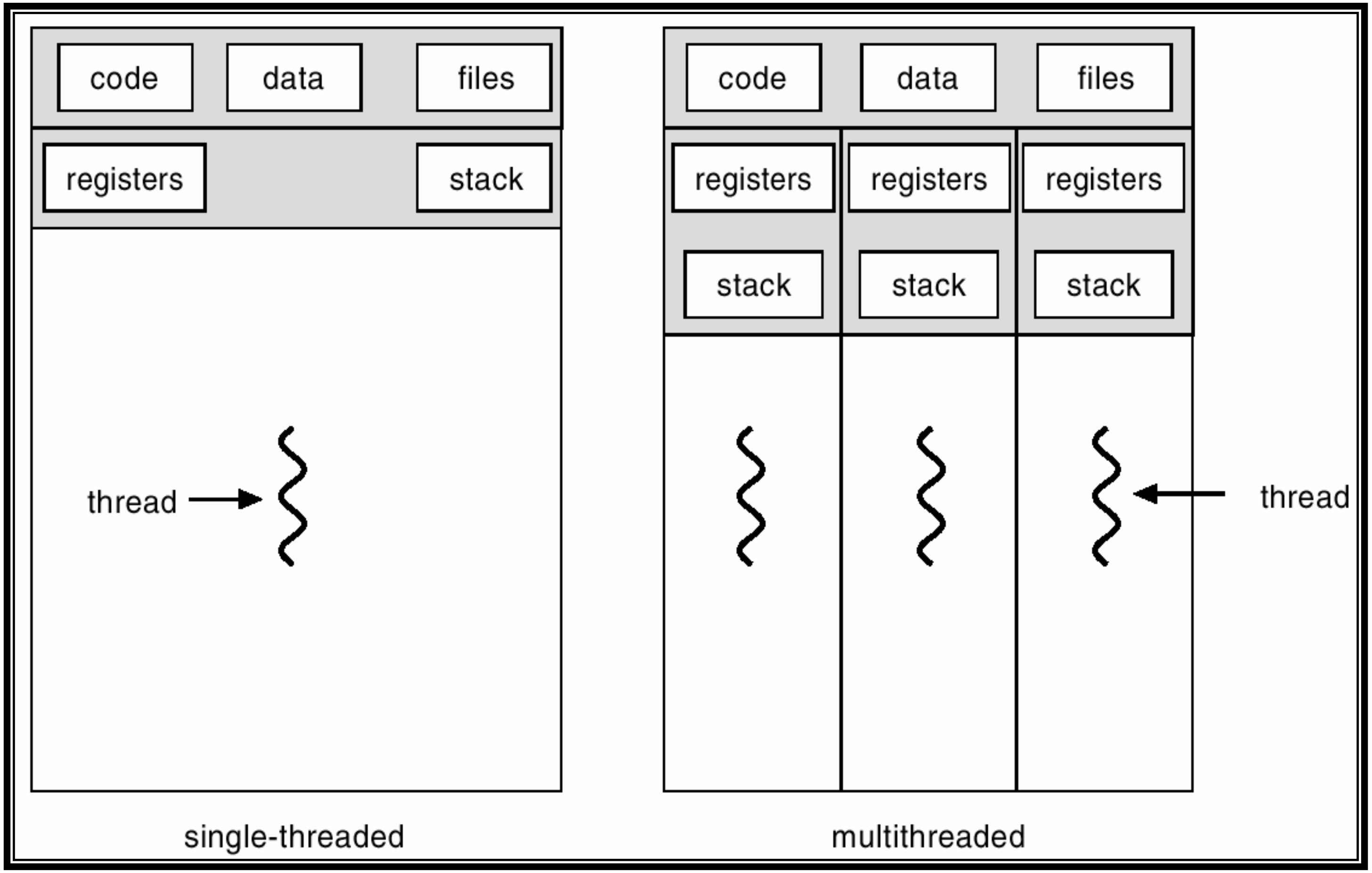
- Overview

- Thread concept +
- Benefits of threads +
- Thread states +
- Supporting threads +

-- Thread Concepts ...

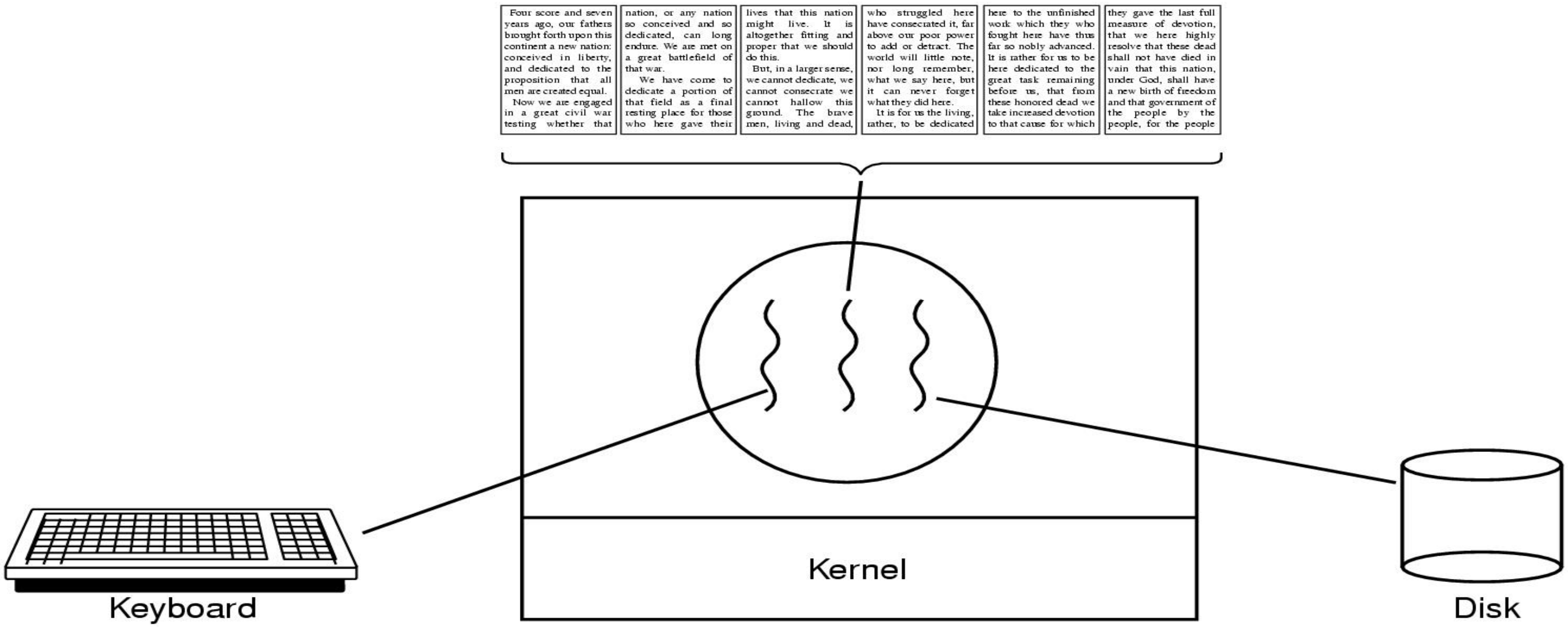
- A thread
  - Is simply an execution stream through a process.
  - Some times it is called lightweight process (LWP)
  - Has:
    - A program counter
    - A register set
    - A stack
    - State
  - It shares with other threads of the same process:
    - Data section
    - Code section
    - Global Variables
    - Accounting information
    - Other OS resources, such as open files and signals

... -- Thread Concepts ...



... -- Thread Concepts

- A modern application is implemented as a process with several threads. For example a word processor can have:
  - A thread to display graphics
  - A thread for performing spelling
  - A thread for reading user input



-- Benefits of Threads

- Responsiveness:
  - One thread could be blocked while another thread could be responding to user requests.
- Resource sharing:
  - Threads of the same process share resources.
- Economy:
  - Compared to processes, threads are:
    - Faster to create (around 30 times faster in Solaris 2)
    - Faster to context switch (around 5 times faster in Solaris 2)
    - Easier to manage
    - Consume less resources
    - Communicate with out involving the kernel (Those of the same process)
- Utilization of multiprocessor architecture

-- Thread States

- Threads have three states:
  - Running
  - Ready
  - Blocked

- They have no suspend state in user level threads (ULT) (ULTs will be explained later) because all threads within the same process share the same address space. Suspending (swapping) a single thread involves suspending all threads of the same process.

- Termination of a process, terminates all threads within the process.

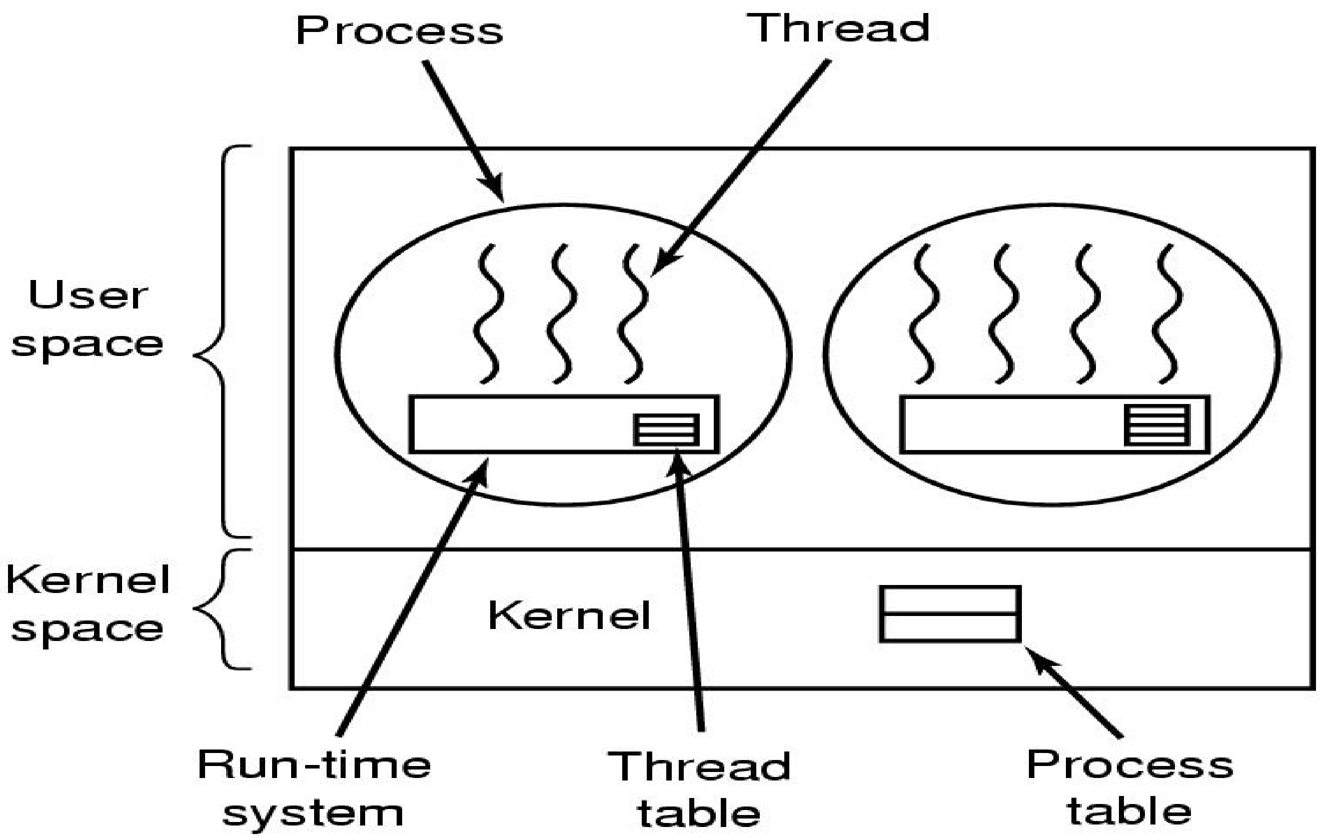
-- Supporting Threads

- Support for threads may be provided at either the user level or the kernel level. In this section we will discuss:
  - User Level Threads +
  - Kernel Level Threads +
  - User Vs. Kernel-Level threads +

--- User-level Threads (ULT)

- Are supported above the kernel.
- Are implemented by thread library.
- With no support from the kernel, the thread library provides:
  - Thread creation
  - Thread termination
  - Thread scheduling
  - Thread Management
- All thread management is done in the user space.
- Examples
  - POSIX Pthreads
  - Mach C-threads
  - Solaris threads

---- A user-level threads package



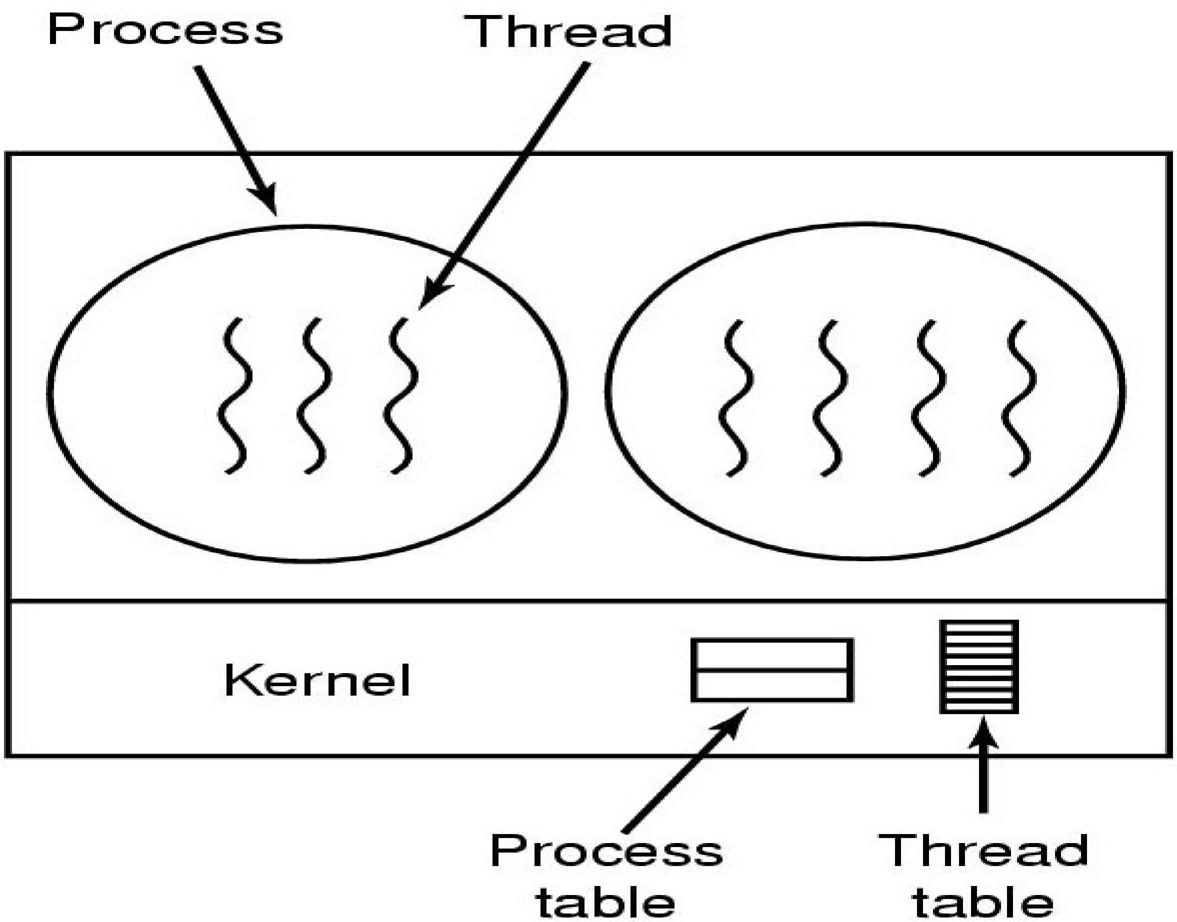
--- Kernel-Level Threads (KLT)

- Are supported by the OS.

- The kernel does thread:
  - Creation
  - termination
  - Scheduling
  - Management

- Examples:
  - Windows 95/98/NT/2000
  - Solaris
  - Tru64 UNIX
  - BeOS
  - Linux

----- A Threads Package managed by Kernel

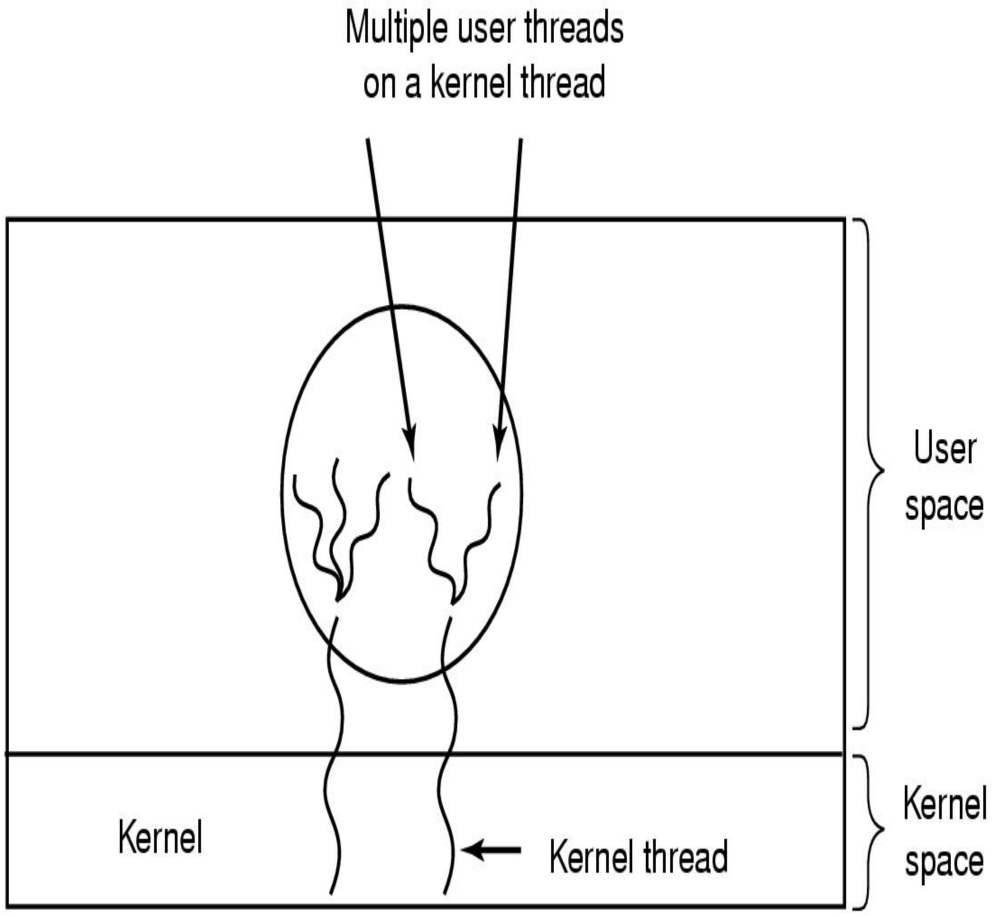


--- ULT Vs. KLT

- ULTs are:
  - Faster than KLTs.
  - More portable than KLT
  - Tunable by user
- In a single threaded kernel:
  - With ULT, if one thread is blocked, all the threads which belong to the same KLT get blocked. (Some system threading libraries translate blocking system calls into nonblocking system calls).
  - With KLT, if one thread is blocked, the other threads of the same process don't get blocked.

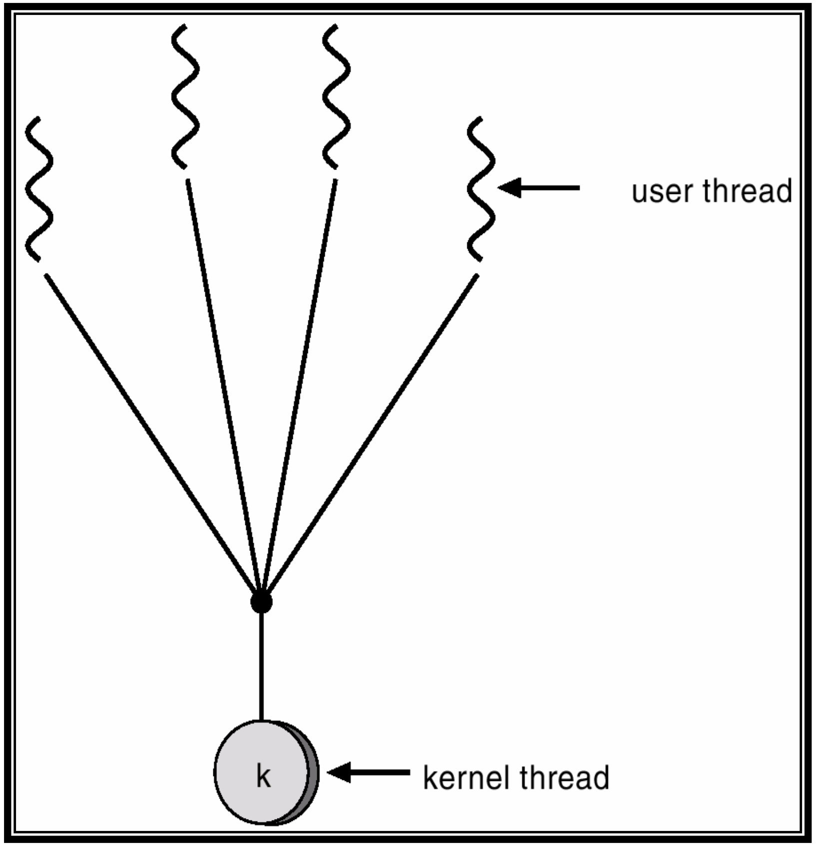
- Multithreading Models

- Many systems provide support for both user & kernel threads, resulting in different multi-threaded models. The four common types of threading implementation are:
  - Many-to-One Model+
  - One-to-One Model +
  - Many-to-Many Model +
  - Two-level Model +



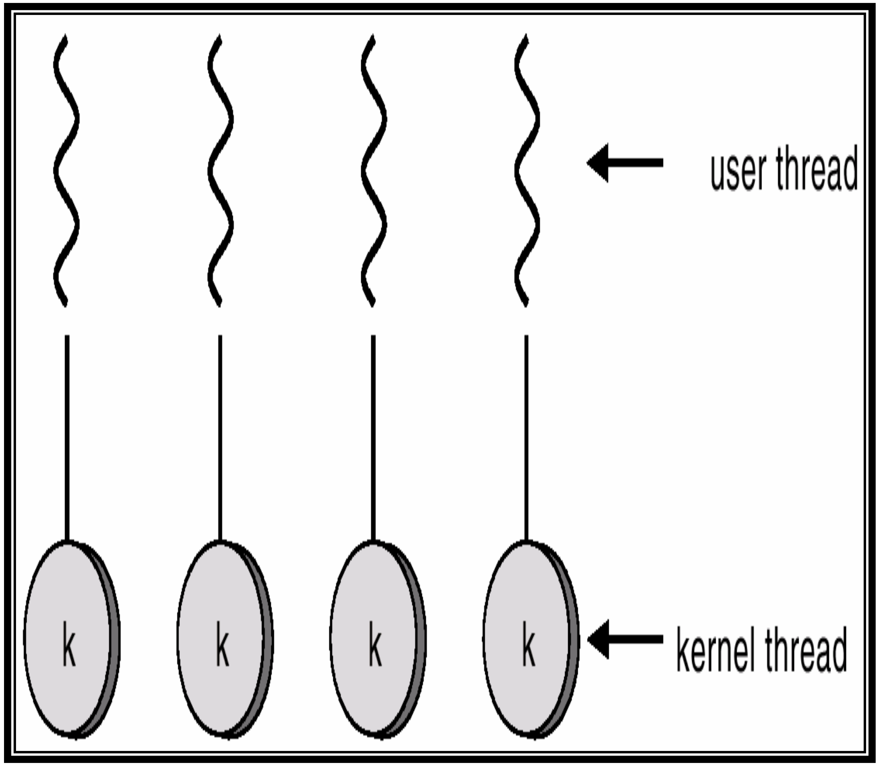
-- Many-to-One Model

- Many user-level threads mapped to single kernel thread.
- Used on systems that do not support kernel threads.
- Efficient: Management is done in the user space.
- Can be blocked.
- No parallelism.
- Example
  - Green threads of Solaris 2



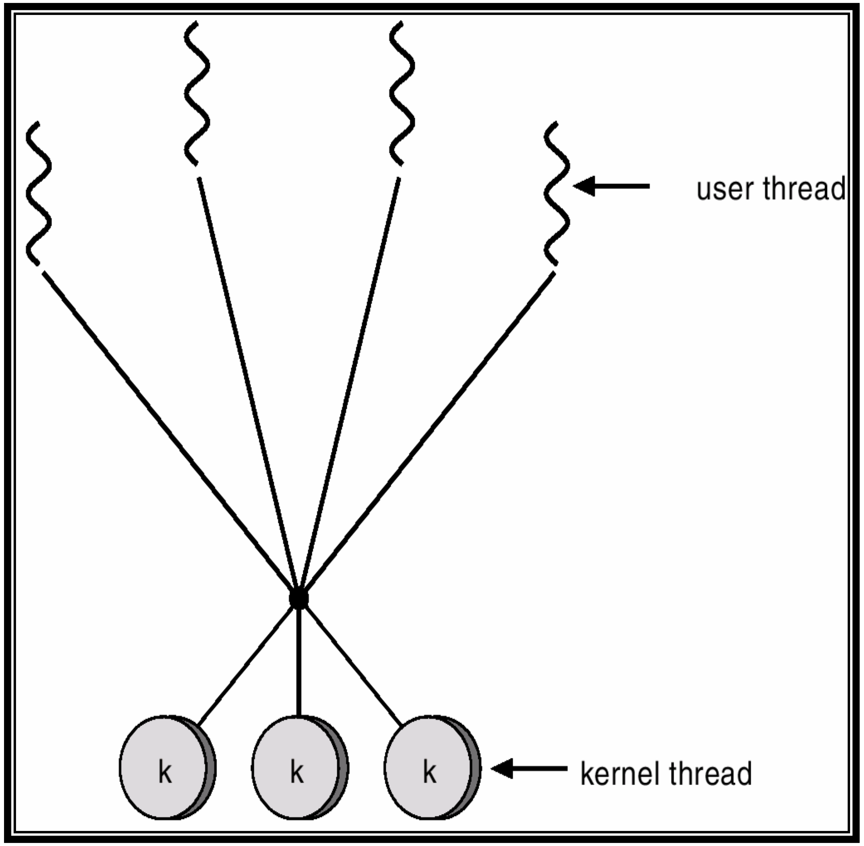
-- One-to-One Model

- Each ULT maps to KLT.
- More concurrency than Many-to-one.
- Threads can run in parallel.
- When one thread gets blocked, CPU is assigned another thread.
- Drawback:
  - Frequent creation of KLTs. (overhead)
  - May need to Limit the number of KLTs, hence ULTs.
- Examples:
  - Windows 95/98/NT/2000/XP
  - OS/2



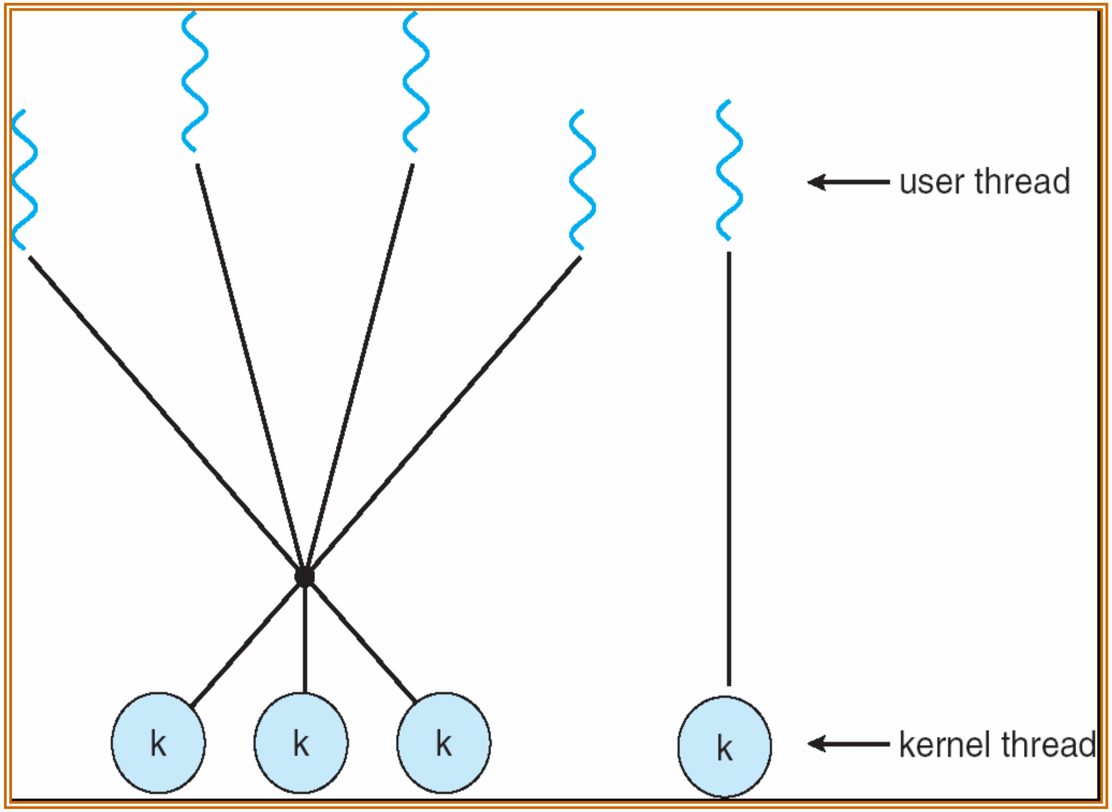
-- Many-to-Many Model

- Multiplexes ULTs to less than or equal number of KLTs.
- Allows the operating system to create a sufficient number of kernel threads.
- Less concurrency than One-to-One but better than Many-to-one.
- More ULTs can be created in this model than in the One-to-One.
- If a thread is blocked, CPU is assigned another thread.
- Examples:
  - Solaris 2
  - 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



- Threading Issues

- In this section we will discuss some issues to consider with multithreaded programs.
  - Thread creation +
  - Thread cancellation +
  - Signal handling +
  - Thread-specific Data +
  - Thread pools +

-- Thread Creation

- The semantic of fork and exec is different for threads and processes.
  - If a thread in a program calls fork:
    - Does it duplicate all the threads of the process or
    - Or it only duplicates the thread that invoked the fork.
  - Some UNIX systems have chosen to have 2 versions of fork.
    - One that duplicates all threads (when exec follows fork).
    - Another one that duplicates the thread that invoked the fork (when no exec).

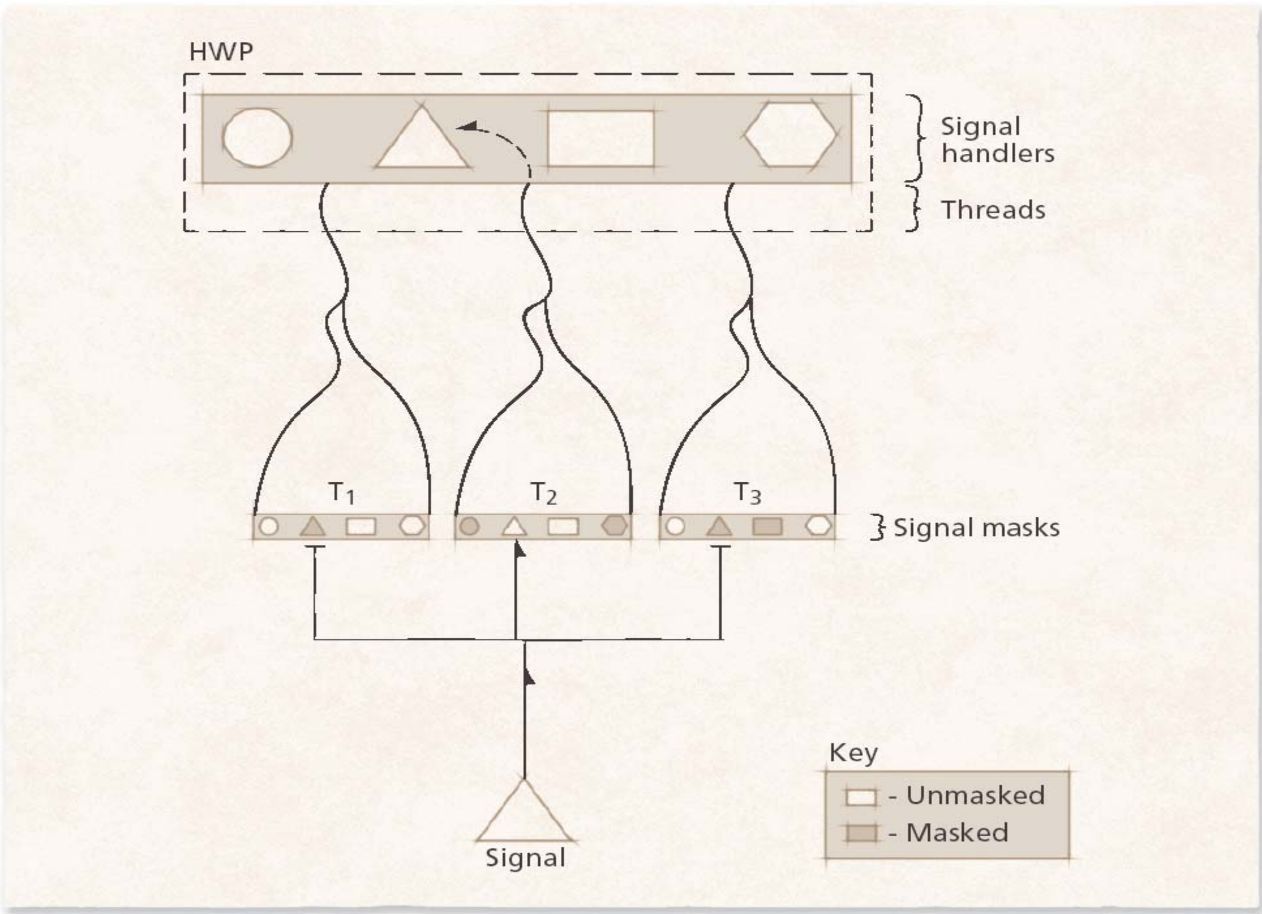
-- Threading Cancellation

- Is a task of terminating a thread before it has competed.
  - Example:
    - Threads concurrently searching a DB. If one thread returns the result others might be cancelled.
    - When a user presses the STOP button on a web browser.
- Cancellation of a thread can occur in two ways:
  - **Asynchronous**: One thread immediately terminated by another. (used by most OS)
  - **Deferred**: A thread checks to terminate itself.

-- Signal Handling

- A signal is used to notify a process that a particular event has occurred.
- A signal can be:
  - Synchronous
  - Asynchronous
- Signals follow the same pattern
  - A signal is generated by the occurrence of a particular event
  - A generated signal is delivered to a process
  - One delivered, the signal must be handled by either
    - A default signal handler
    - A user-defined signal handler
- A process may have many threads so where then should a signal be delivered. In general, the following options exist:
  - Deliver the signal to the thread to which the signal applies
  - Deliver the signal to every thread in the process (like <control>< c>)
  - Deliver the signal to certain threads in the process
  - Assign a specific thread to receive all signals of the process (Solaris 2)

--- Thread Signal Delivery



-- Thread-Specific Data

- Threads belonging to a process share the data of the process.
- In some circumstances each thread might need its own copy of data which is called thread specific-data.
  - <u>Example:</u> a unique identifier of a transaction.

-- Thread Pools

- A number of KLTs are created at process startup and are placed into a pool.
- A thread waits in the pool till it is needed.
- If a thread completes a service, it is not destroyed. It is put back into the pool.
- If a pool contains no available threads, the server waits until one becomes free.
- The benefits of thread pools are:
  - Faster, because of not creating and destroying threads.
  - Number of threads can be controlled.

- 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.

- Summary

- Thread: lightweight process (LWP).
- Benefits of threads:
  - Responsiveness
  - resource sharing
  - economy
  - Utilization of multiprocessor architecture
- Thread states:
  - running
  - ready
  - Blocked
- ULT, KLT.
- Multi-threading Models:
  - Many-to-one
  - One-to-one
  - Many-to-Many
  - Two-level
- Thread creation, thread cancellation, Signal handling, thread pools

End