UNIT-2

CHAPTER-2

MULTI THREADED PROGRAMMING

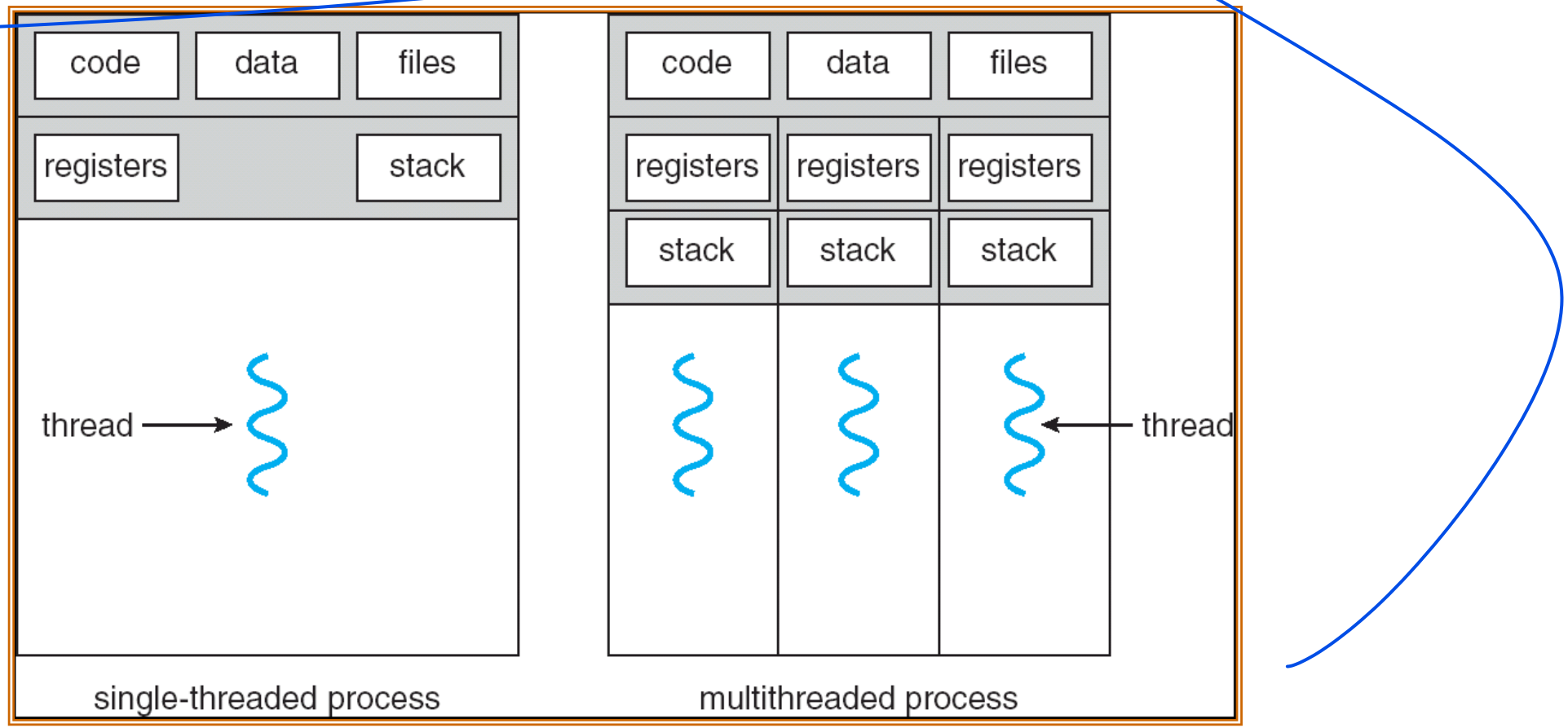
Threads

- Overview
- Multithreading Models

Thread

- A thread is a basic unit of CPU utilization; it comprises a thread ID, a program counter, a register set, and a stack.
- Thread shares with other threads belonging to the same process its code section, data section, and other operating-system resources, such as open files and signals.
- If a process has multiple threads of control, it can perform more than one task at a time.

Difference between a traditional single-threaded process and a multithreaded process.



Motivation

- An application typically is implemented as a separate process with several threads of control.

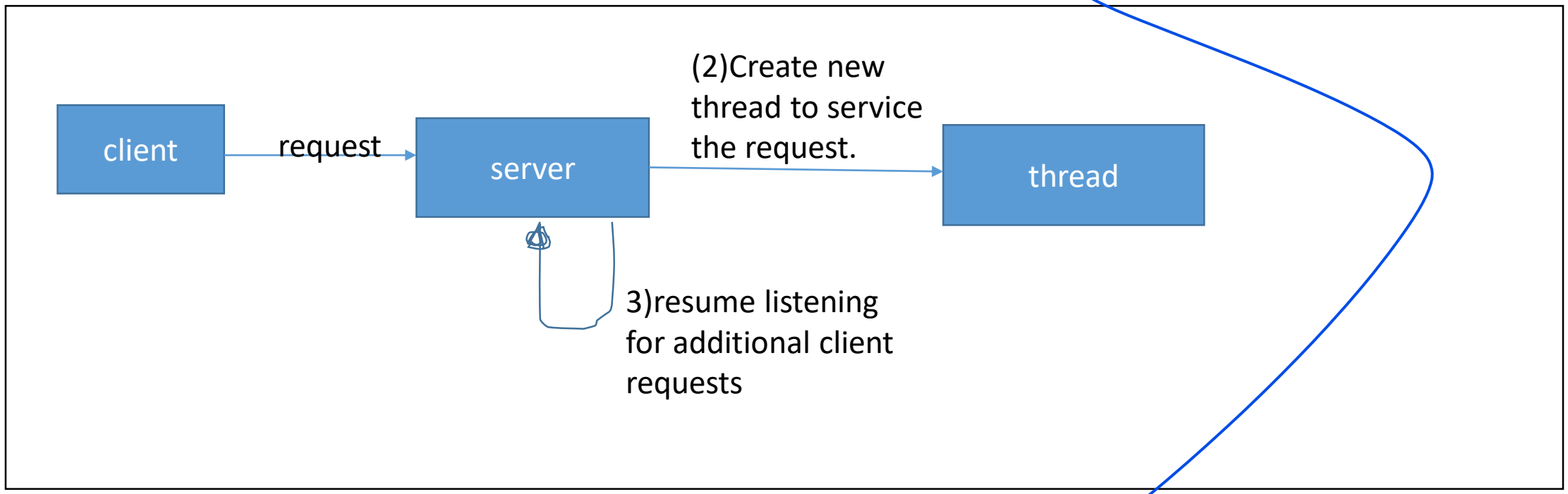
Example-1 :A web browser might have one thread display images or text while another thread retrieves data from the network.

Example-2:A word processor may have a thread for displaying graphics, another thread for responding to keystrokes from the user, and a third thread for performing spelling and grammar checking in the background.

Example-3: A busy web server may have several (perhaps thousands) of clients concurrently accessing it. If the web server ran as a traditional single-threaded process, it would be able to service only one client at a time. The amount of time that a client might have to wait for its request to be serviced could be enormous.

- One solution is to have the server run as a single process that accepts requests. When the server receives a request, it creates a separate process to service that request. In fact, this process-creation method was in common use before threads became popular. Process creation is time consuming and resource intensive.
- It is generally more efficient to use one process that contains multiple threads. This approach would multithread the web-server process. The server would create a separate thread that would listen for client requests; when a request was made, rather than creating another process, the server would create another thread to service the request.

Multithreaded server architecture



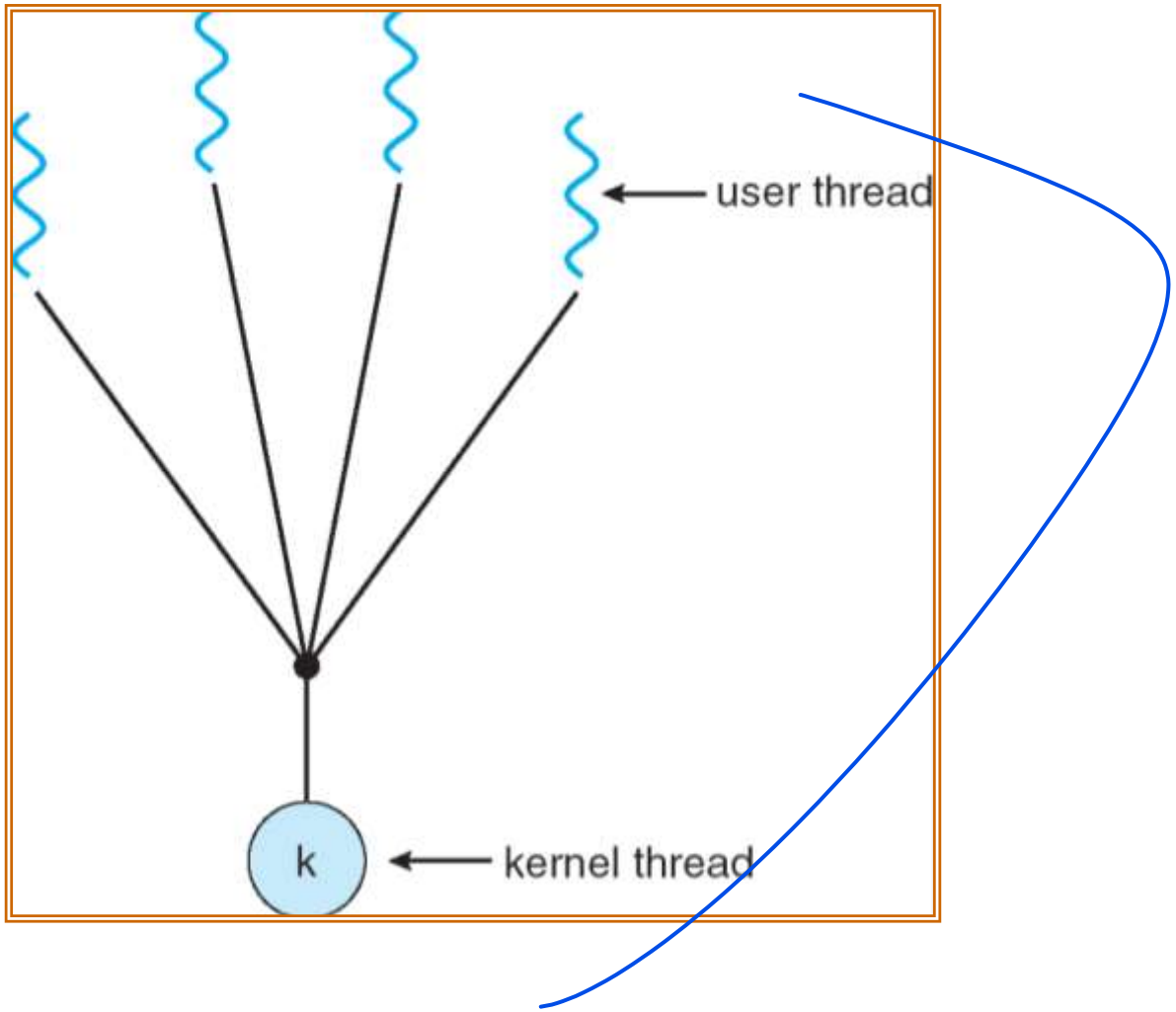
Benefits

- Responsiveness : Multithreading an interactive application may allow a program to continue running even if part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user.
- Resource Sharing: By default, threads share the memory and the resources of the process to which they belong. The benefit of sharing code and data is that it allows an application to have several different threads of activity within the same address space.
- Economy: Threads share resources of the process to which they belong, it is more economical to create and context-switch threads.
- Utilization of MP Architectures: The benefits of multithreading can be greatly increased in a multiprocessor architecture, where threads may be running in parallel on different processors.

Multithreading Models

- Support for threads may be provided either at the user level, for user threads, or by the kernel, for kernel threads.
- User threads are supported above the kernel and are managed without kernel support, whereas kernel threads are supported and managed directly by the operating system.
- Three common ways of establishing relationship between user threads and kernel threads:
- Many-to-One model
- One-to-One model
- Many-to-Many model

Many-to-One Model



Many-to-One Model

- The many-to-one model maps many user-level threads to one kernel thread.
- Solaris operating system uses this model.

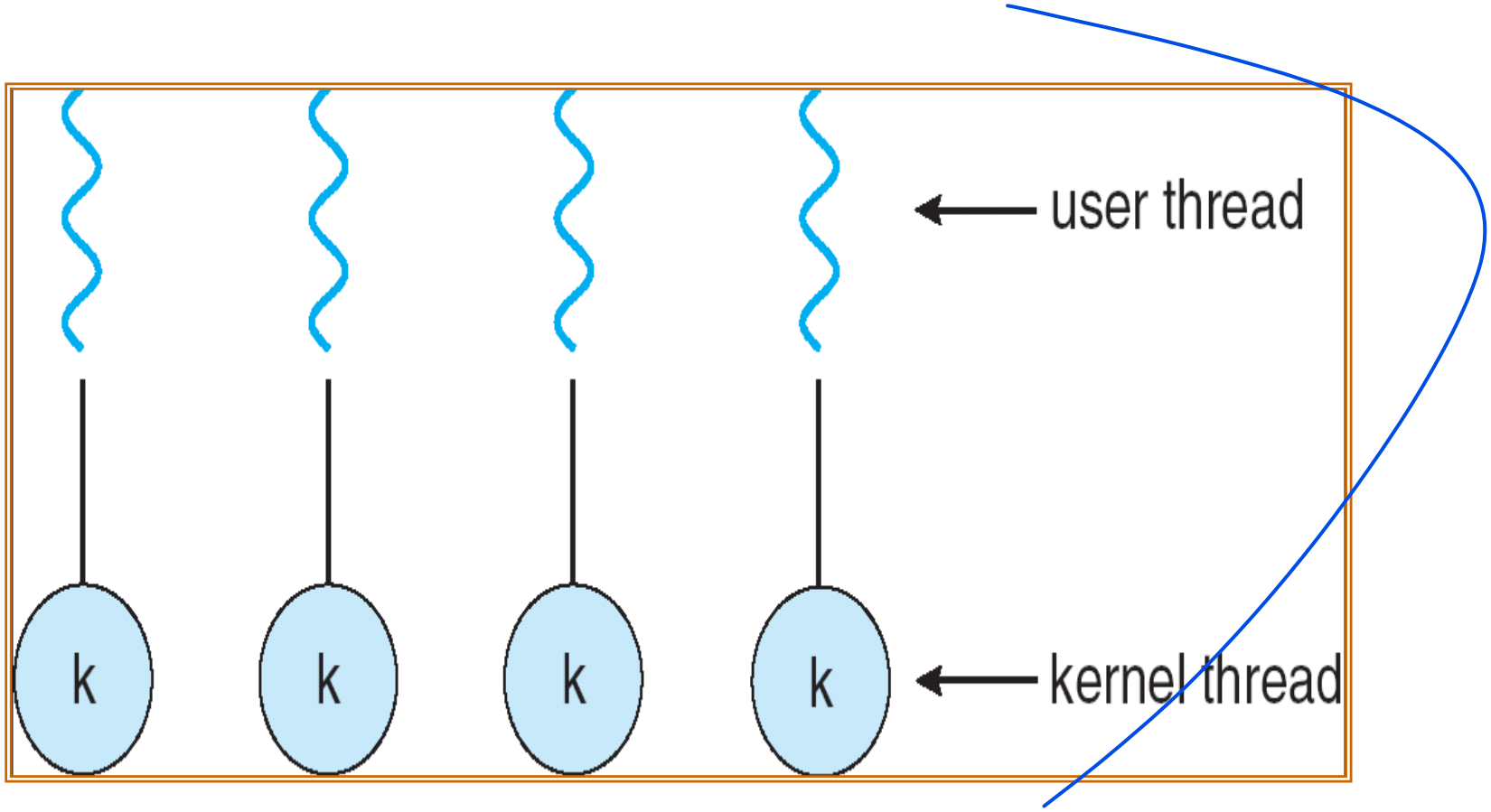
Advantages:

- Thread management is done by the thread library in user space, so it is efficient.

Disadvantage:

- The entire process will block if a thread makes a blocking system call.
- only one thread can access the kernel at a time, multiple threads are unable to run in parallel on multiprocessors.

One-to-one Model



One-to-one Model

The one-to-one model maps each user thread to a kernel thread.

- Linux, along with the family of Windows operating systems-including Windows 95, 98, NT, 2000, and XP implement the one-to-one model.

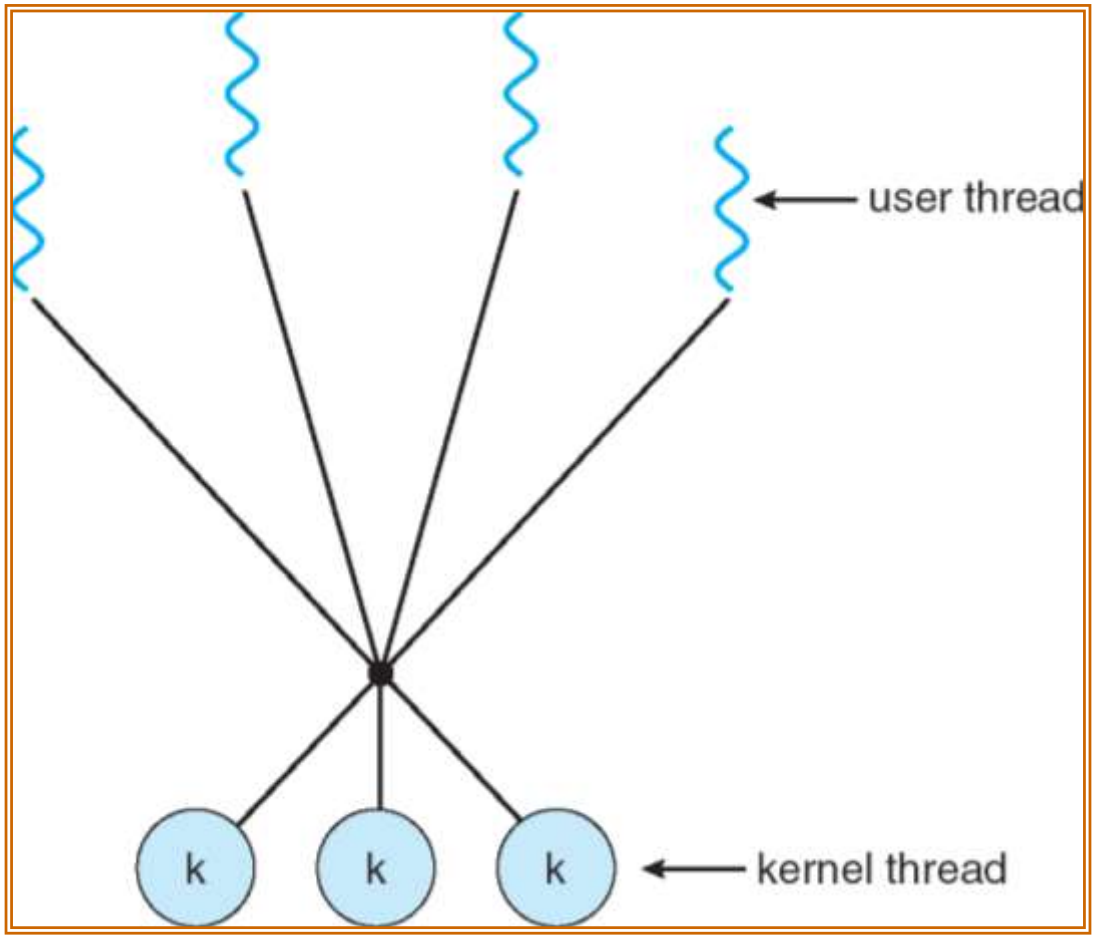
Advantages:

- It provides more concurrency than the many-to-one model by allowing another thread to run when a thread makes a blocking system call.
- It also allows multiple threads to run in parallel on multiprocessors.

Disadvantages:

- Creating a user thread requires creating the corresponding kernel thread. Because the overhead of creating kernel threads can burden the performance of an application, most implementations of this model restrict the number of threads supported by the system.

Many-to-Many Model



Many-to-Many Model

- The many-to-many model multiplexes many user-level threads to a smaller or equal number of kernel threads.
- The number of kernel threads may be specific to either a particular application or a particular machine.
- many-to-one model allows the developer to create as many user threads as he/she wishes, true concurrency is not gained because the kernel can schedule only one thread at a time.
- Developers can create as many user threads as necessary, and the corresponding kernel threads can run in parallel on a multiprocessor.
- When a thread performs a blocking system call, the kernel can schedule another thread for execution.
- Supported by operating systems such as IRIX, HP-UX, and Tru64 UNIX.