4)

Multithreading allows the execution of multiple parts of a program at the same time. These parts are known as threads and are lightweight processes available within the process. Therefore, multithreading leads to maximum utilization of the CPU by multitasking.

The main models for multithreading are one to one model, many to one model and many to many model. Details about these are given as follows

`Many to many model.`

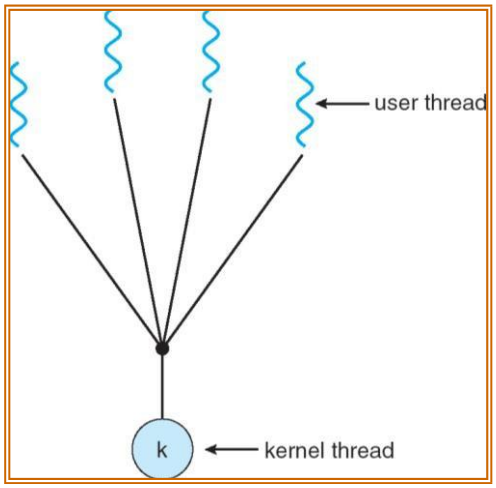
`Many to one model.`

`one to one model.`

.

Many-to-One Model

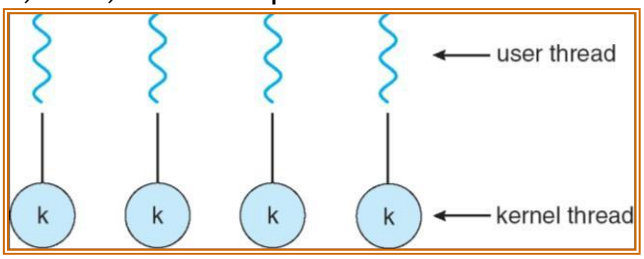
- The many-to-one model maps many user-level threads to one kernel thread.
- Thread management is done by the thread library in user space, so it is efficient; but the entire process will block if a thread makes a blocking system call.
- Also, because only one thread can access the kernel at a time, multiple threads are unable to run in parallel on multiprocessors.
- **Green threads**—a thread library available for Solaris—uses this model, as does GNU **Portable Threads.**



One-to-One Model

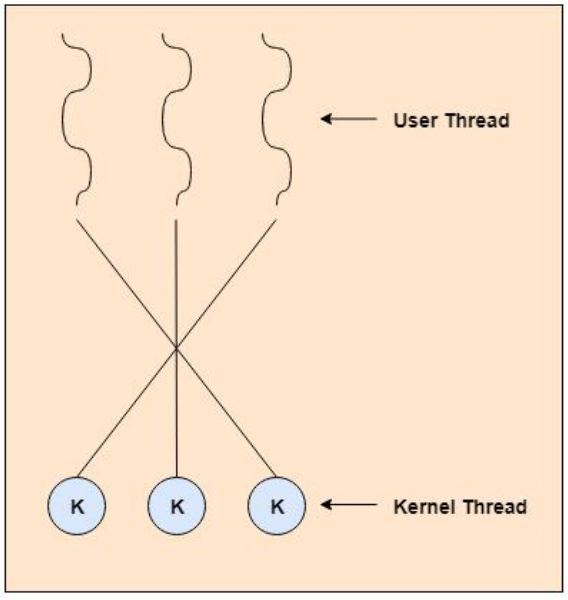
- The one-to-one model maps each user thread to a kernel thread.
- It provides more concurrency than the many-to-one model by allowing another thread to run when a thread makes a blocking system call; it also allows multiple threads to run in parallel on multiprocessors.
- The only drawback to this model is that 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.

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



Many-to-Many Model

- The many-to-many model multiplexes many user-level threads to a smaller or equal number of kernel threads.
- Many-to-one model allows the developer to create as many user threads as she wishes; true concurrency is not gained because the kernel can schedule only one thread at a time.
- The one-to-one model allows for greater concurrency, but the developer has to be careful not to create too many threads within an application.
- The many-to-many model suffers from neither of these shortcomings: Developers can create as many user threads as necessary, and the corresponding kernel threads can run in parallel on a multiprocessor.
- Also, when a thread performs a blocking system call, the kernel can schedule another thread for execution.



Many to Many Model