Name Aqib Hafeez

Registration no #5359

Assignment 3

Q: what is Multithreading Model? And model of Multithreading

Many to one multithreading model

One to one multithreading model

Many to Many multithreading models

Multithreading Model:

There are two types of threads to be managed in a modern system: User threads and kernel threads. User threads are supported above the kernel, without kernel support. These are the threads that application programmers would put into their programs. Kernel threads are supported within the kernel of the OS itself. All modern OSes support kernel level threads, allowing the kernel to perform multiple simultaneous tasks and/or to service multiple kernel system calls simultaneously.

Many-To-One Model

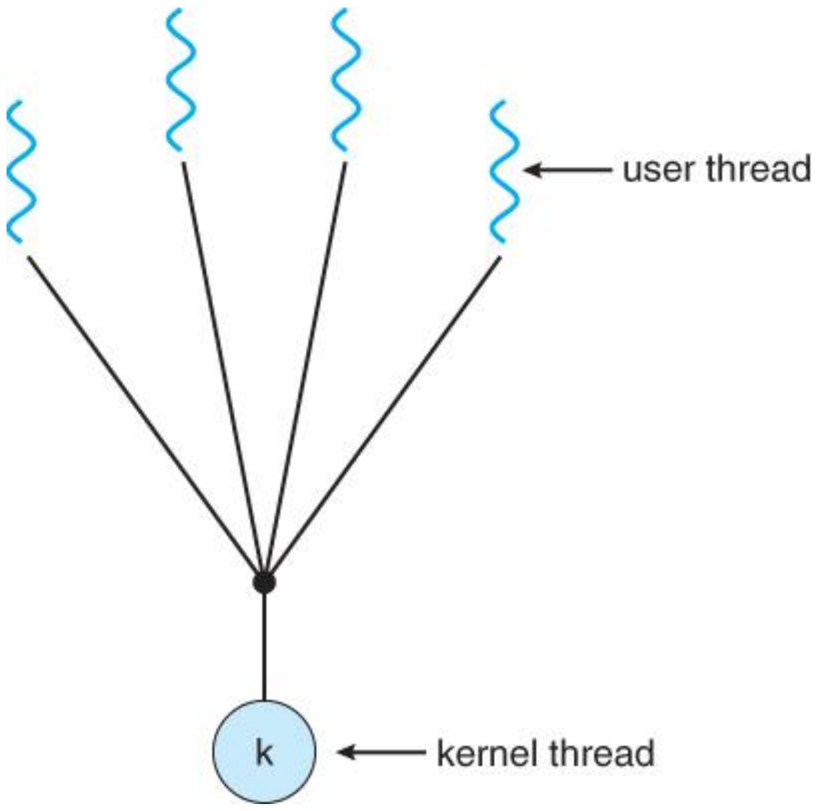
In the many-to-one model, many user-level threads are all mapped onto a single kernel thread.

Thread management is handled by the thread library in user space, which is very efficient.

However, if a blocking system call is made, then the entire process blocks, even if the other user threads would otherwise be able to continue.

Because a single kernel thread can operate only on a single CPU, the many-to-one model does not allow individual processes to be split across multiple CPUs.

Green threads for Solaris and GNU Portable Threads implement the many-to-one model in the past, but few systems continue to do so today.



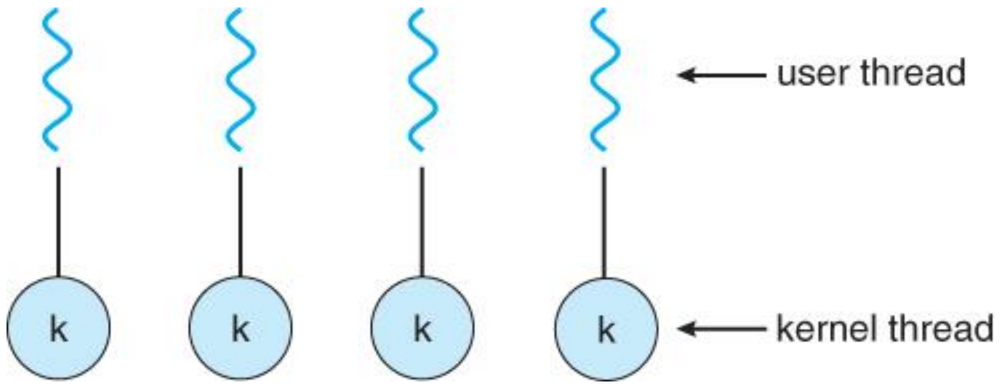
One-To-One Model

The one-to-one model creates a separate kernel thread to handle each user thread.

One-to-one model overcomes the problems listed above involving blocking system calls and the splitting of processes across multiple CPUs.

However the overhead of managing the one-to-one model is more significant, involving more overhead and slowing down the system.

Most implementations of this model place a limit on how many threads can be created.



Many-To-Many Model

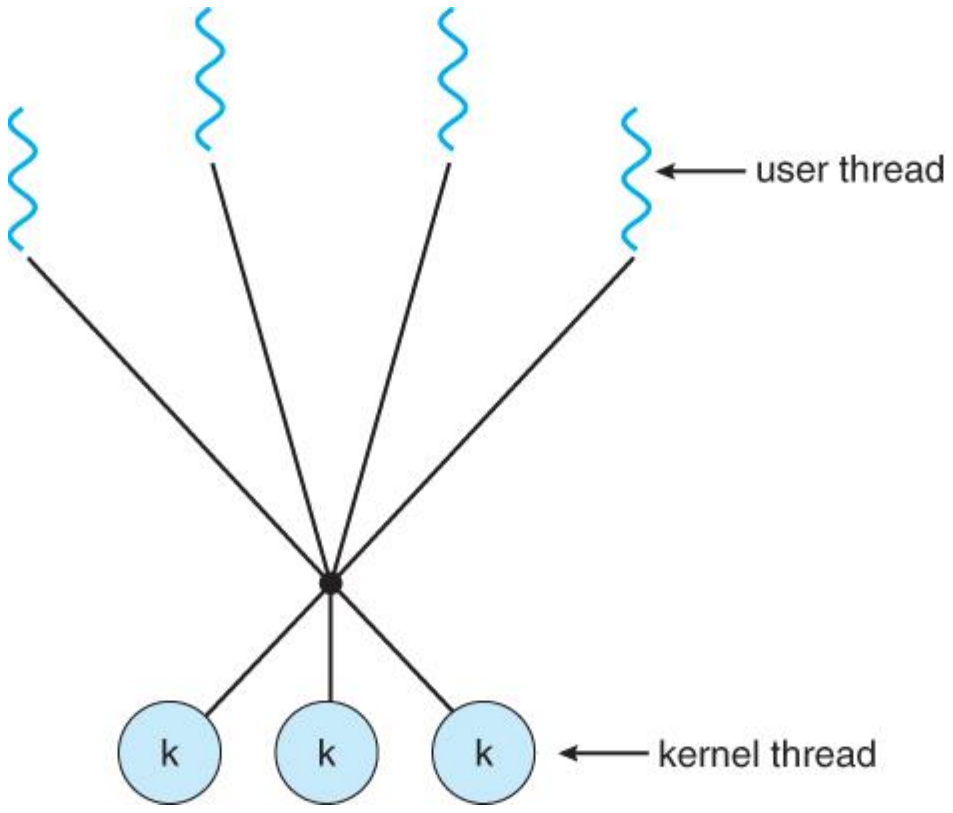
The many-to-many model multiplexes any number of user threads onto an equal or smaller number of kernel threads, combining the best features of the one-to-one and many-to-one models.

Users have no restrictions on the number of threads created.

Blocking kernel system calls do not block the entire process.

Processes can be split across multiple processors.

Individual processes may be allocated variable numbers of kernel threads, depending on the number of CPUs present and other factors.


user thread
k
k
k
kernel thread