Threads

Threads

Thread is an execution unit that consists of its own program counter, a stack, and a set of registers. Threads are also known as Lightweight processes. Threads are a popular way to improve the application through parallelism. The CPU switches rapidly back and forth among the threads giving the illusion that the threads are running in parallel. As each thread has its own independent resource for process execution, multiple processes can be executed parallelly by increasing the number of threads.

Types of Thread

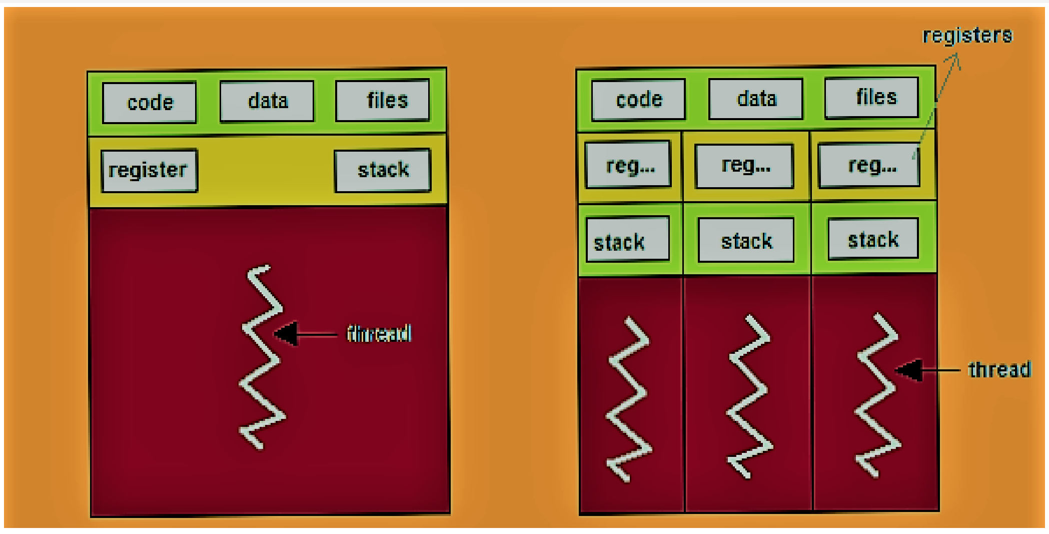
- **User threads** are above the kernel and without kernel support. These are the threads that application programmers use in their programs.
- **Kernel threads** are supported within the kernel of the OS itself. All modern OSs support kernel-level threads, allowing the kernel to perform multiple simultaneous tasks and/or to service multiple kernel system calls simultaneously.

Program vs Process vs Thread

Program	**Process**	**Thread**
An execution file stored in harddrive.	**An execution file stored in memory.**	**An execution path of part of the process.**
Program contains in the instruction.	**A process is a sequence of instruction.**	**Thread is a single sequence stream within a process.**
One Program contains many processes.	**One Process can contain several threads.**	**One thread can belong to exactly one process.**

Multithreading

Multithreading is a phenomenon of executing multiple threads at the same time. For example, in a browser, multiple tabs can be different threads. MS Word uses multiple threads: one thread to format the text, another thread to process inputs, etc.



Single threaded Process Multithreaded Process

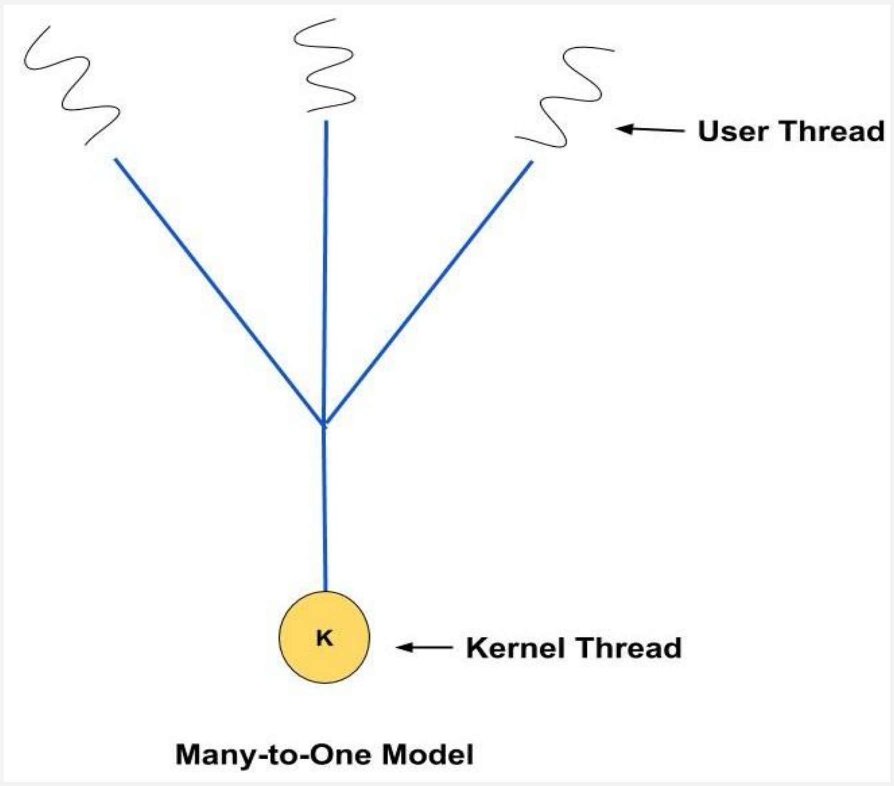
Multithreading Models

The user threads must be mapped to kernel threads, by one of the following strategies:

Many to One Model

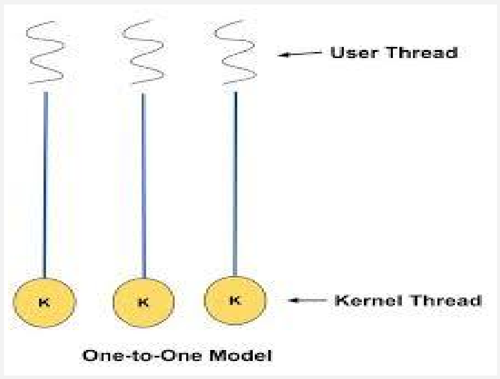
As the name suggests there is a many to one relationship between threads. Here, multiple user threads are associated or mapped with one kernel thread. The thread management is done on the user level so it is more efficient.

Example: Initial implementation of Java threads on the Solaris system



One to One Model

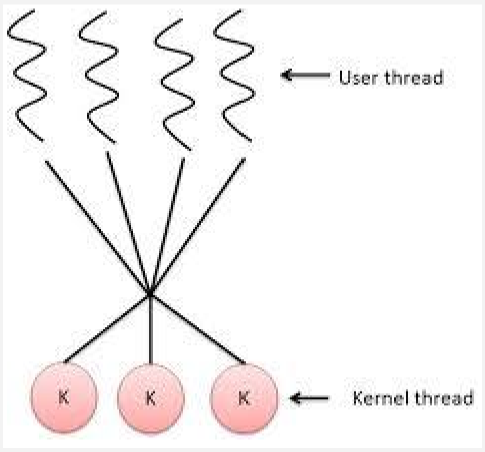
The one-to-one model creates a separate kernel thread to handle each and every user thread. Most implementations of this model place a limit on how many threads can be created. Linux and Windows from 95 to XP implement the one-to-one model for threads.



Example: Windows NT and the OS/2 threads package

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. Blocking the kernel system calls does not block the entire process.



Many to Many Model

Example: Implementation of Java on an MT operating system.