SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department Of Artificial Intelligence and Data Science

Course Code & Name –23ITB203 & Operating Systems

II Year / IV Semester

Unit 2 - THREADS

THREADS

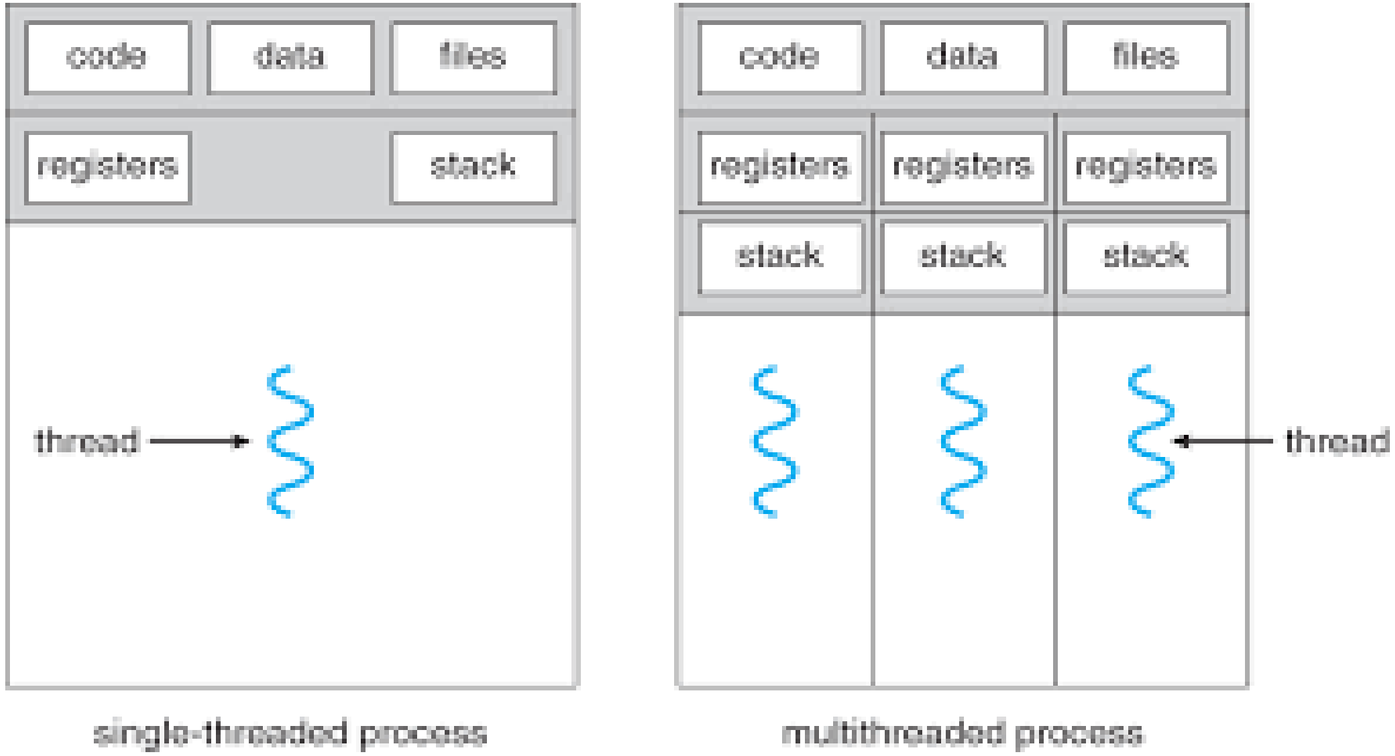
A thread is a basic unit of CPU utilization.

It comprises

- A thread ID
- A program counter
- A register and
- A stack

It 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.

A traditional / heavy weight process has a single thread of control.
If a process has multiple threads of control, it can perform more than one task at a time.



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 code sharing is that it allows an application to have several different threads of activity all within the same address space.

*Economy:

Allocating memory and resources for process creation is costly, because threads share resources of the process to which they belong,it is more economical to create and context switch threads

* Utilization of multiprocessor architecture:

The benefits of multithreading can be greatly increased in a multiprocessor architecture, where threads may be running in parallel on different processors. A single threaded process can only run on one CPU no matter how many are available. Multithreading on a multi CPU machine increases concurrency.

Types of threads

In the operating system, there are two types of threads.
User level thread: Supported above the kernel and are managed without kernel support

Kernel-level thread: Supported and managed directly by the OS.
Ultimately there must exist a relationship between user threads and kernel threads.

There are three common ways of establishing this relationship

1. Many to One
2. One to One
3. Many to many

MULTITHREADED MODELS

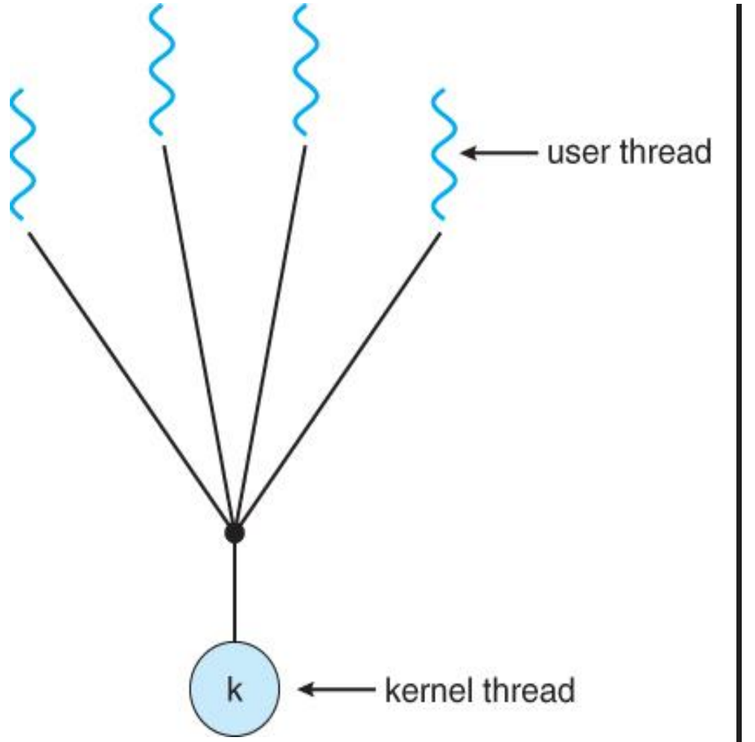
Many to One model :

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.



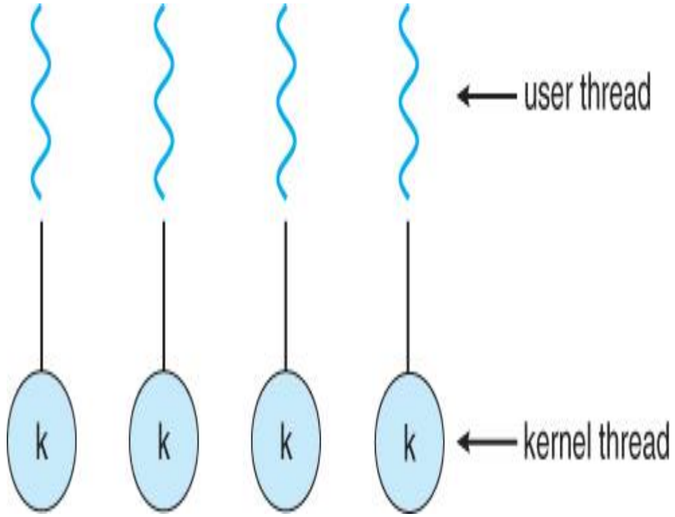
One to One model

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.

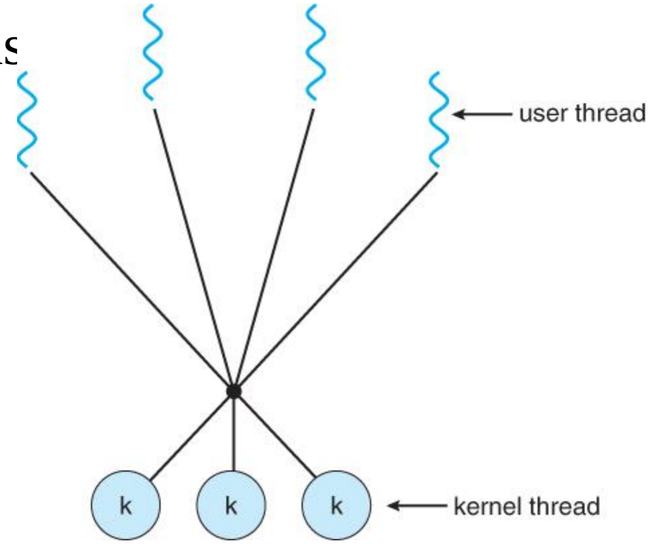
However the overhead of managing the one-to-one model is more significant, involving more overhead and slowing down the system.

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.