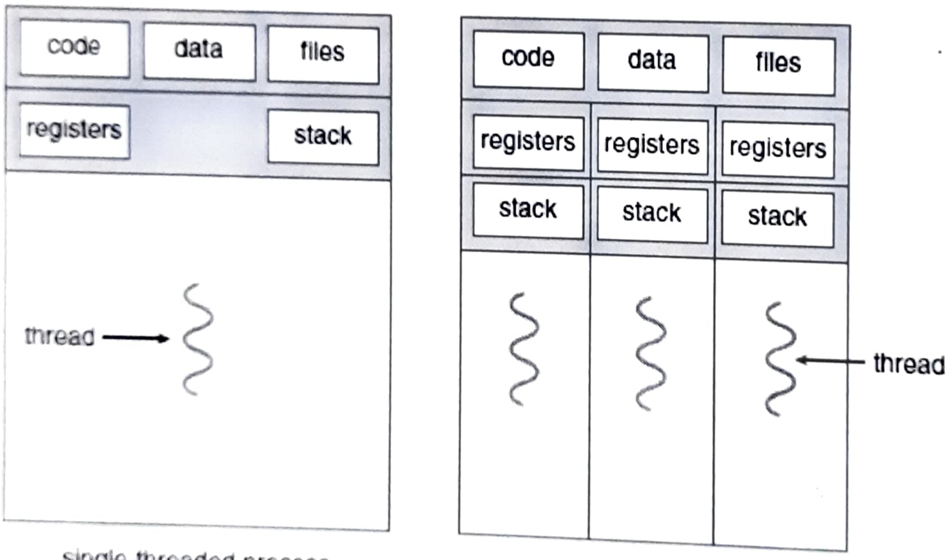
Threads

A thread is a basic unit of CPU utilization. It comprises of:

- a thread ID
- a program counter
- a register set
- and a stack

Threads share code, data and files.



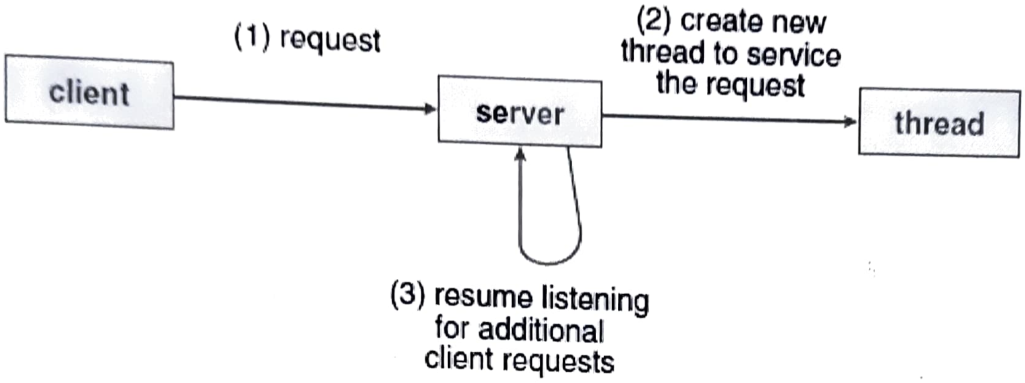
single-threaded process

multithreaded process

Most software applications that run on modern computers are multithreaded.

Process creation is time consuming and resource intensive. It is efficient to use one process with multiple threads.

Example:


client
(1) request
server
(2) create new thread to service the request
thread
(3) resume listening for additional client requests

Threads also play vital role in IPC. Most OS kernels are also multi-threaded.

Benefits: (of multi-threaded programming)

1. Responsiveness.
   - Continue running even if a part is blocked or time consuming

2. Resource Sharing.
   - ~~Seve~~ ~~th~~ -several threads of activity in the same address space.

3. Economy.
   - Economical to create & context switch.

4. Scalability.
   - Threads in parallel on different cores.

Amdahl's law

Identifies potential performance gains from adding additional computing cores to an application that has both Serial & parallel Components.

If $\underline{S}$ is the portion of application that must be performed Serially on a System with $\underline{N}$ processing cores, then,

$$\text{Speedup} \leq \frac{1}{S + \frac{(1-S)}{N}}$$

Eg:- If there is an application on a system which has 75% parallel & 25% Serial, If we run this with two processing cores, we can get a speed up of 1.6 times.

If we add 2 more, the Speedup is 2.28 times.

Programming Challenges for Threads in multicore Systems:-

- Identifying tasks
- Balance
- Data Splitting
- Data dependency
- Testing and debugging

User Threads are threads managed by user-level thread libraries & Kernel threads are supported by the kernel

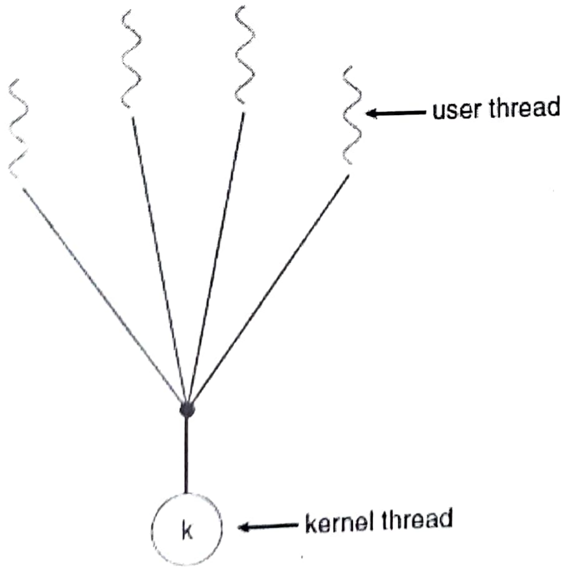
Three primary thread Libraries:

- POSIX Pthreads
- Windows threads
- Java threads

Multi-threading Models

A relationship must exist between user threads & kernel threads which are managed by OS.

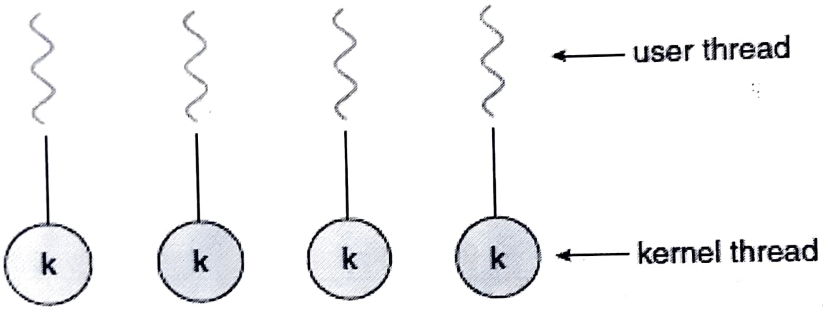
Many-to-One Model



- Maps many user level threads to one kernel thread.

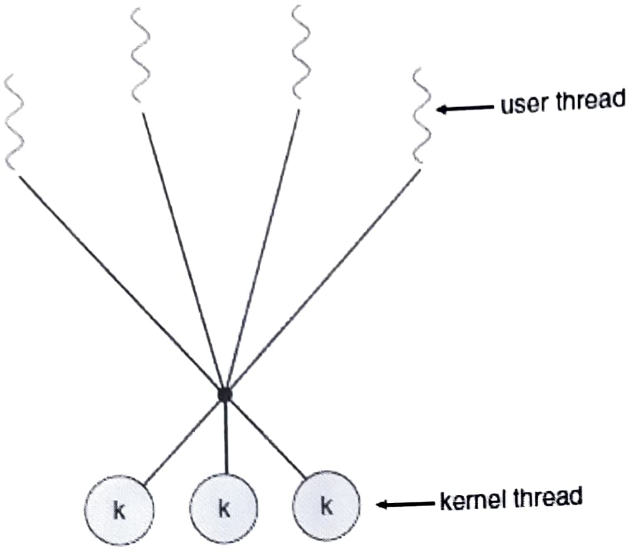
- Thread management is done by thread library in user space, so its efficient.
- Entire process will block if one thread makes a blocking system call
- Multiple threads cannot run in parallel on a multicore system as only one thread can access kernel at a time
- Example: Green threads (solaris systems)

One-to-One model



- Maps each user thread to a kernel thread
- A blocking system call blocks only one called thread
- Every user thread needs to create a kernel thread : overhead and reduces the performance
- Number of threads per process sometimes is restricted due to the overhead
- Examples: Windows, linux

Many-to-Many Model

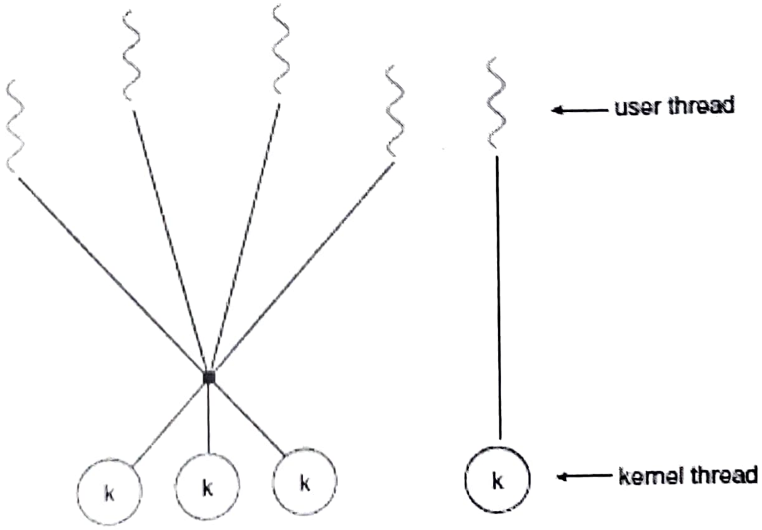


- Multiplexes many user level threads to a smaller or equal number of kernel threads
- Developers can create as many user threads as necessary, and the corresponding kernel threads can run in parallel on a multiprocessor.
- when a thread performs a blocking system call, the kernel can schedule another thread for execution
- Eg. windows with Thread Fiber package

Variation :

Two-level Model →

Eg:- IRIX, HP-UX, Tru64 Unix, Solaris 8 & older



Notes:

- A thread library provides the programmer with an API for creating and managing threads.
  - ↳ The first approach is to provide a library entirely in user space with no kernel Support
  - ↳ The second approach is to implement a kernel-level library supported directly by the OS.

- Three main thread libraries in use are
  - POSIX Pthreads
  - Windows
  - Java

- Implicit threading : To support the design of multithreaded applications, transfer the creation and management of threading from application developers to compilers and run-time libraries.

- Thread Pool : Create a number of threads at process startup and place them into a pool, where they sit and wait for work. Threads are picked for work & returned back after completion.

Thread Cancellation involves terminating a thread before it has completed.

Cancellation can be:

Asynchronous: One thread immediately terminates the target thread.

Deferred: The target thread periodically checks whether it should terminate, allowing it an opportunity to terminate itself in an orderly fashion.

GATE Questions

Q1: Which one of the following is FALSE?
a. User level threads are not scheduled by the kernel.
b. When a user level thread is blocked, all other threads of its process are blocked.
c. Context switching between user level threads is faster than context switching between kernel level threads.
d. Kernel level threads cannot share the code segment

Q2: Which of the following is/are shared by all the threads in a process?
I. Program Counter
II. Stack
III. Address space
IV. Registers

a. I and II only
b. III only
c. IV only
d. III and IV only

Q1: d

Q2: b