

Bule Hora University

School of informatics

Department of Computer Science

Operating System Lecture

By: Firaol B.

12/18/2025

Threads (cont'd)

2

Outlines:-

- What is Threads?
- what is Multithreading?
- why do we need threads?

Threads (cont'd)

3

WHAT IS THREADS?

- A **thread** is a flow of execution through the process code
- A thread is also called a **lightweight process** which is the unit of dispatching.
- Threads represent a software approach to improving performance of operating system by reducing the overhead
- Each thread belongs to exactly one process and no thread can exist outside a process.

12/18/2025

Threads (cont'd)

4

- Each thread represents a separate flow of control.
- Threads have been successfully used in implementing network servers and web server.

Threads have their own:

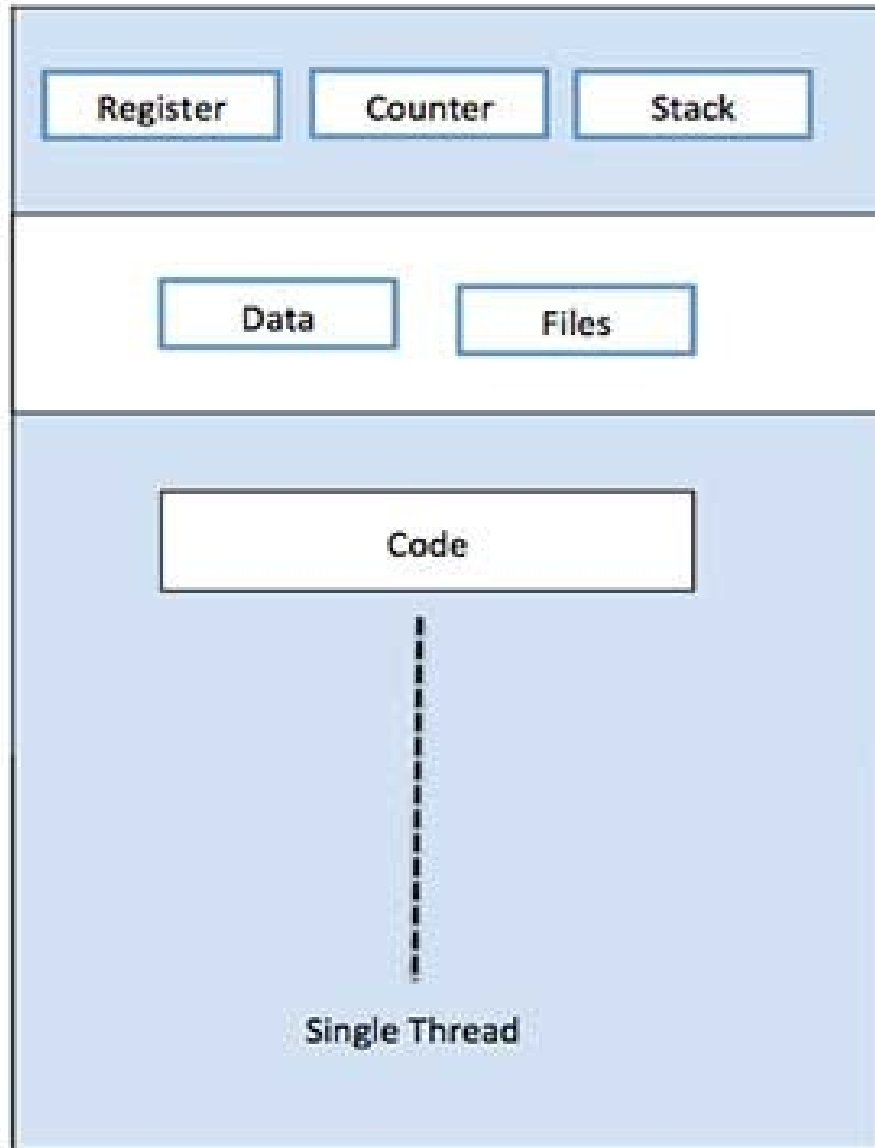
- **Program counter** (keeps track of which instruction to execute next),
- **System registers** (holds its current working variables), and
- A **stack** which contains the execution history.

12/18/2025

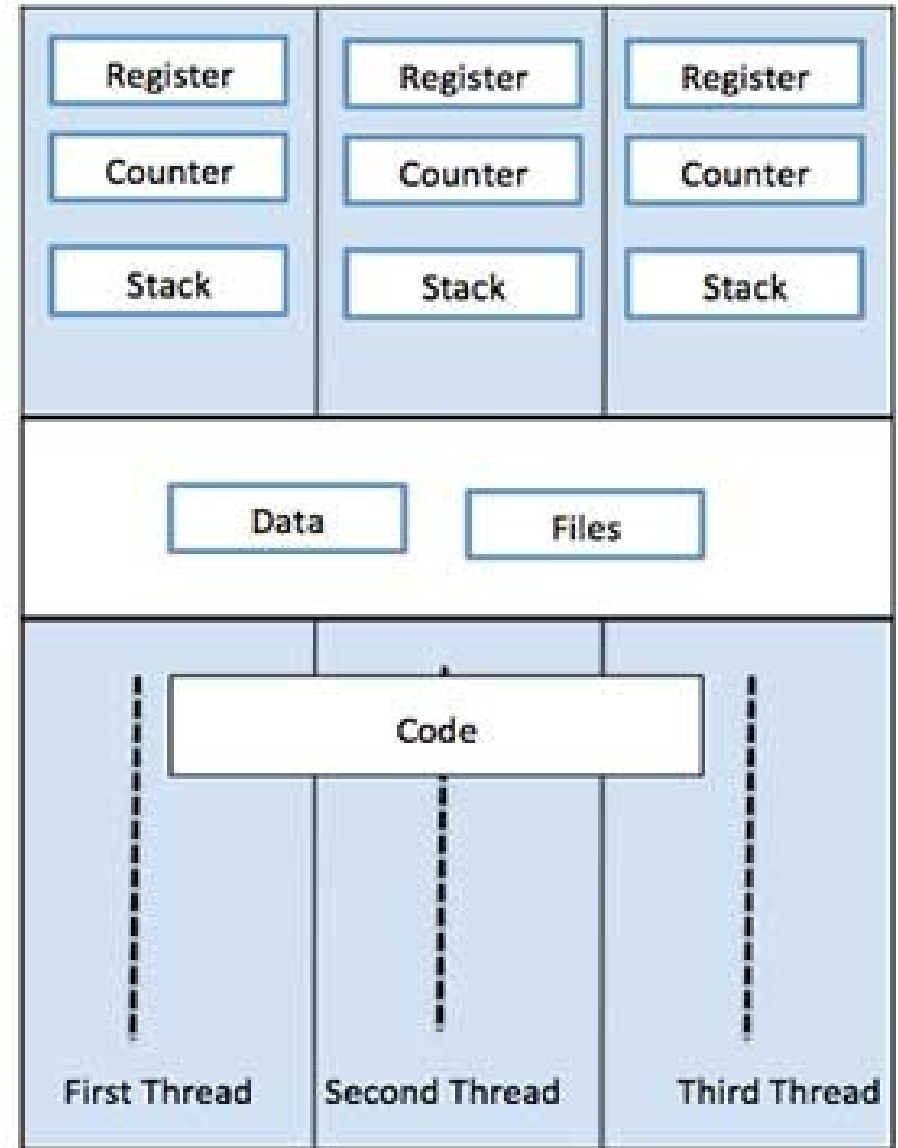
Threads (cont'd)

5

- A thread shares with its peer threads few information like **code segment**, **data segment** and **open files**.
(When one thread alters a code segment memory item, all other threads see that.)



Single Process P with single thread



Single Process P with three threads

Threads (cont'd)

7

Difference
between
Process
and
Thread

Process	Thread
Process is heavy weight or resource intensive.	Thread is light weight, taking lesser resources than a process.
Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

12/18/2023

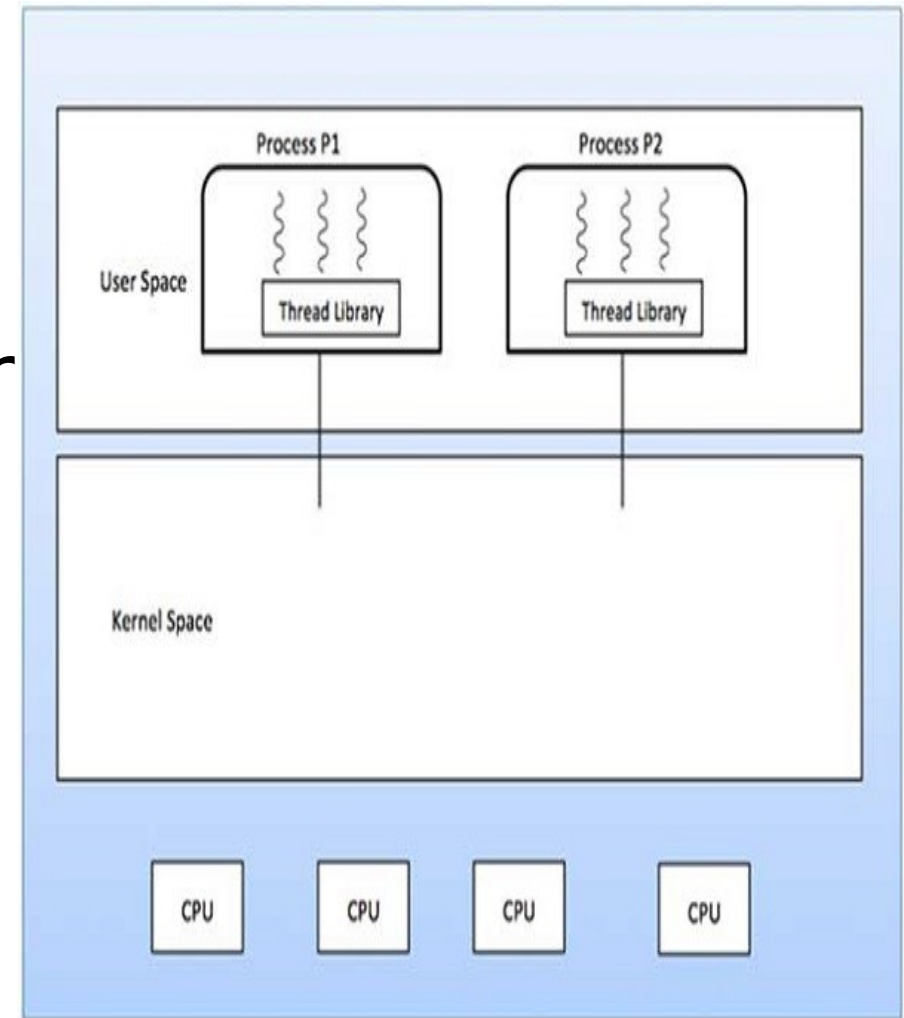
Threads (cont'd)

8

Types of Thread

Threads are implemented in following two ways –

- **User Level Threads** – User managed threads.
- **Kernel Level Threads** – Operating System managed threads acting on kernel, an operating system core.



12/10/2023

Threads (cont'd)

- **User Level Threads**

- In this case, the thread management kernel is not aware of the existence of threads.

The thread library contains code for:

- Creating and destroying threads,
 - Passing message and data between threads,
 - Scheduling thread execution
 - Saving and restoring thread contexts.
- (The application starts with a single thread.)

Threads (cont'd)

10

Advantages

- Thread switching does not require Kernel mode privileges.
- User level thread can run on any operating system.
- Scheduling can be application specific in the user level thread.
- User level threads are fast to create and manage.

Disadvantages

- In a typical operating system, most system calls

12/18/2025

Threads (cont'd)

11

- **Kernel Level Threads**

- In this case, thread management is done by the Kernel.
- There is no thread management code in the application area.
- Kernel threads are supported directly by the operating system.
- Any application can be programmed to be multithreaded.
- All of the threads within an application are supported within a single process.

12/18/2025

Threads (cont'd)

12

- The Kernel maintains context information for the process as a whole and for individuals threads within the process.
- The Kernel performs thread creation, scheduling and management in Kernel space.
- Kernel threads are generally slower to create and manage than the user threads.

Threads (cont'd)

Advantages

- Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
- If one thread in a process is blocked, the Kernel can schedule another thread of the same process.
- Kernel routines themselves can be multithreaded.

Disadvantages

- Kernel threads are generally slower to create and manage than the user threads.

Threads (cont'd)

14

Difference between User-Level & Kernel-Level

S.N.	User-Level Threads	Kernel-Level Thread
1	User-level threads are faster to create and manage.	Kernel-level threads are slower to create and manage.
2	Implementation is by a thread library at the user level.	Operating system supports creation of Kernel threads.
3	User-level thread is generic and can run on any operating system.	Kernel-level thread is specific to the operating system.
4	Multi-threaded applications cannot take advantage of multiprocessing.	Kernel routines themselves can be multithreaded.

12/18/2025

Threads (cont'd)

15

Multithreading Models

-- **Multithreading** is a type of execution model that allows multiple threads to exist within the context of a process such that they execute independently but share their **process resources**

-- Some operating system provide a combined user level thread and Kernel level thread facility (ex. Solaris).

-- In a combined system, multiple threads within the same application can run in parallel.

Multithreading models are three types

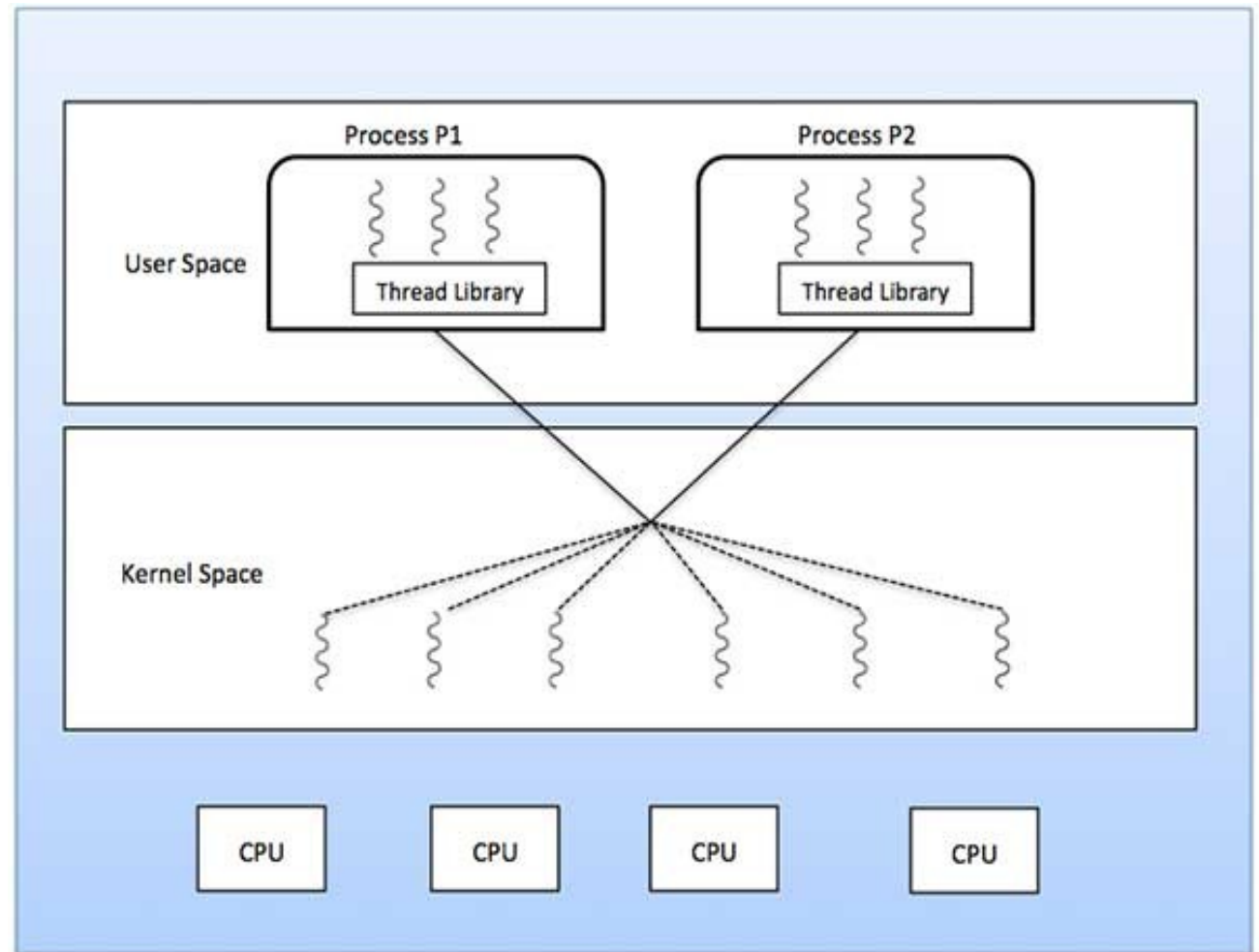
- **Many to many relationship.**
- **Many to one relationship.**
- **One to one relationship**

12/18/2025

Threads (cont'd)

- **Many to Many Model**

The many-to-many model multiplexes any number of user threads onto an equal or smaller number of kernel threads.



12/18/2025

Threads (cont'd)

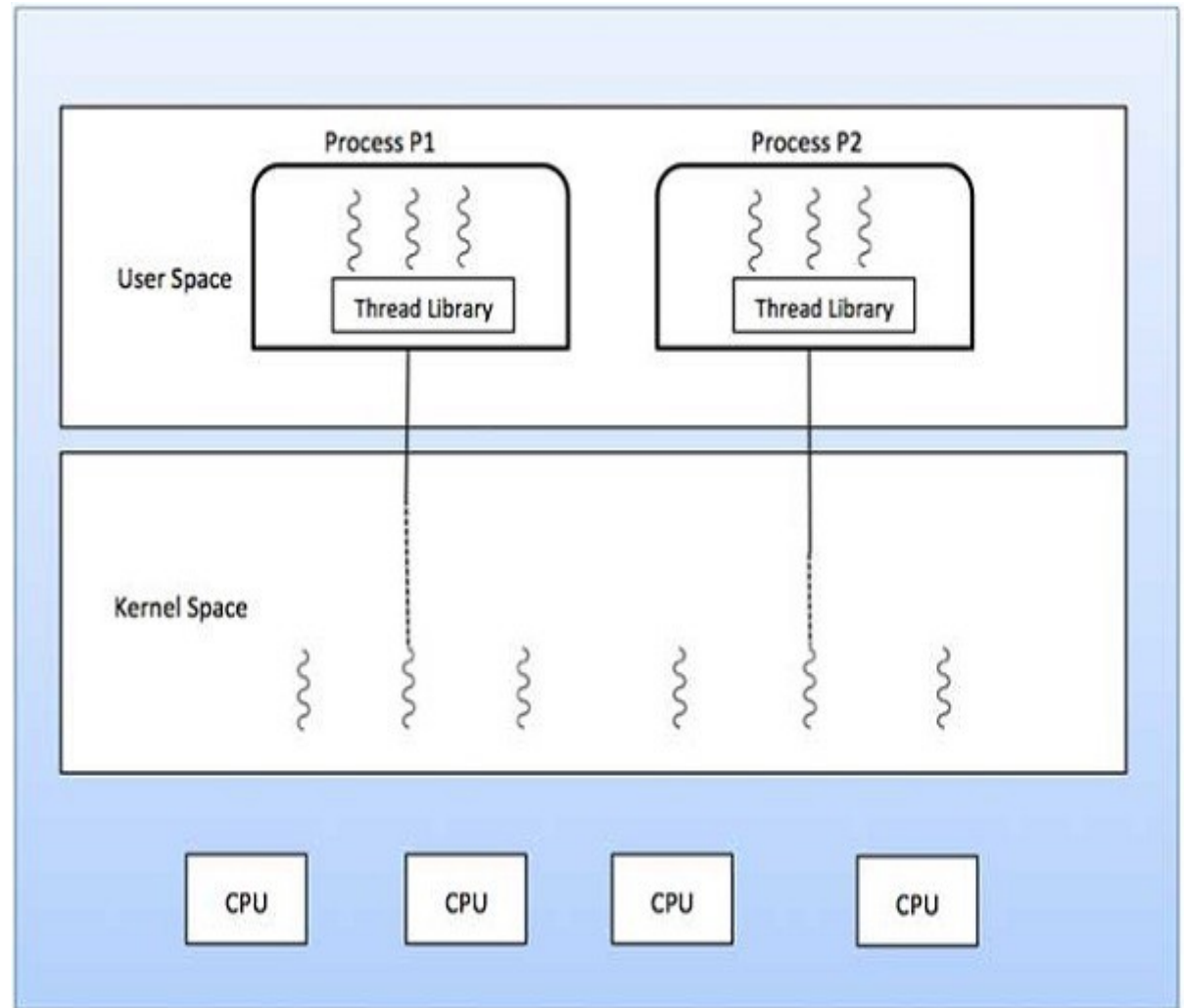
17

- **Many to One Model**

- Maps many user level threads to one Kernel-level thread.

- Thread management is done in user space by the thread library.

- Only one thread can access the Kernel at a time, so multiple threads are unable to run in parallel on multiprocessors

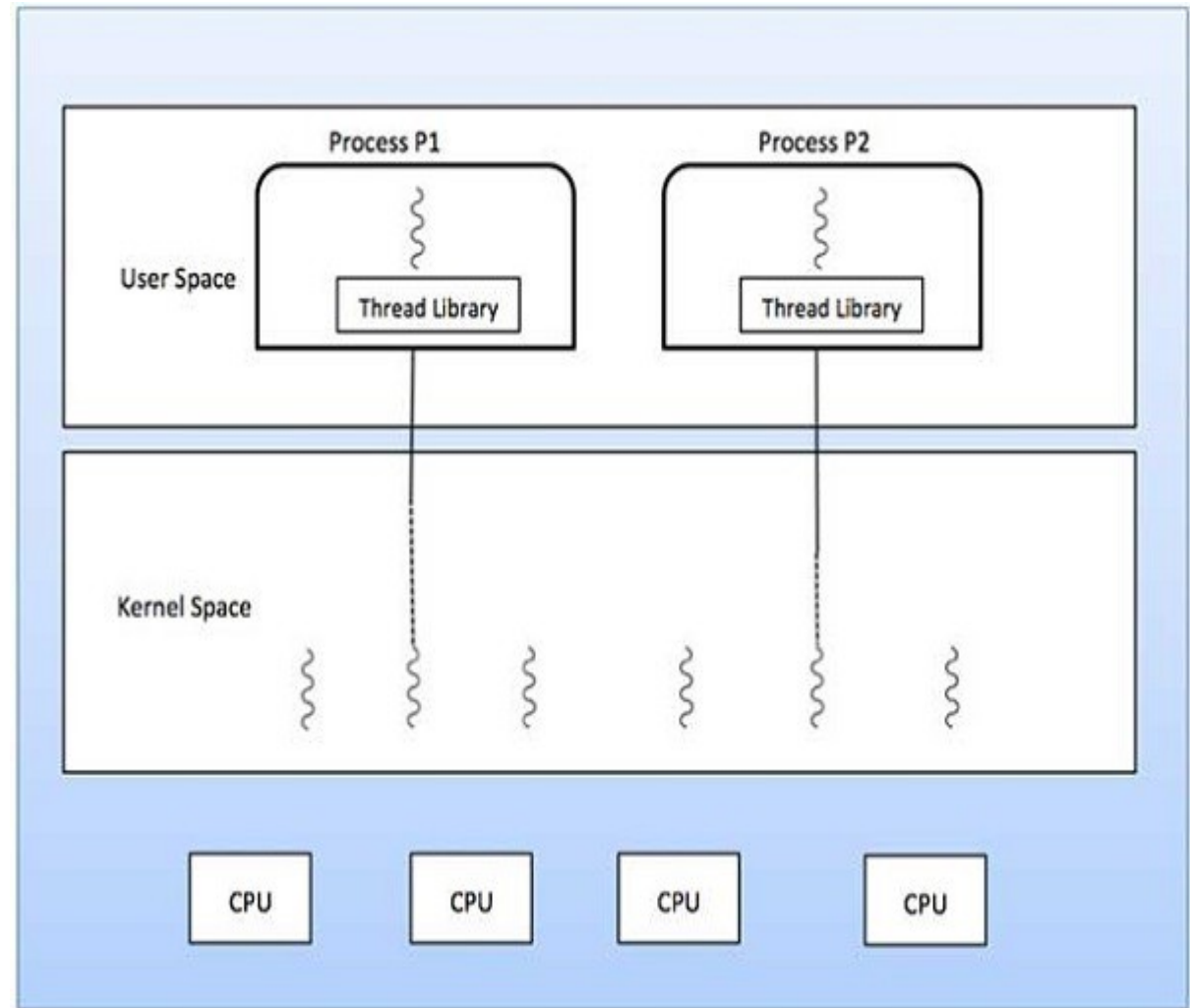


Threads (cont'd)

18

- **One to One Model**

- There is one-to-one relationship of user-level thread to the kernel-level thread.
- This model provides more concurrency than the many-to-one model.
- It also allows another thread to run when a thread makes a blocking system call.
- It supports multiple threads to execute in parallel on microprocessors.



Threads (cont'd)

19

WHY WE NEED THREADS?

- It provide a way to improve application performance through parallelism.
- The threads works independently and provides the maximum utilization of the CPU, thus enhancing the CPU performance.
- It minimize the context switching time.
- Use of threads provides concurrency (doing multiple things at same time) within a process

12/18/2025

Threads (cont'd)

20

- Efficient communication.
- It is more economical to create and context switch threads.
- Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.
- A process with multiple threads make a great server.
- B/c of they share common data, they do not need to use interprocess

12/18/2015

....any questions?