Threads

Untitled 1

Default Style | Liberation Serif | 12

Page 1 of 1 | 0 words, 0 characters | Default Style | English (USA) | 100%

A word processor may have to perform the following tasks *”at the same time”*:

- Displaying graphics (GUI).
- Responding to key strokes.
- Spelling and grammar checking in the background.

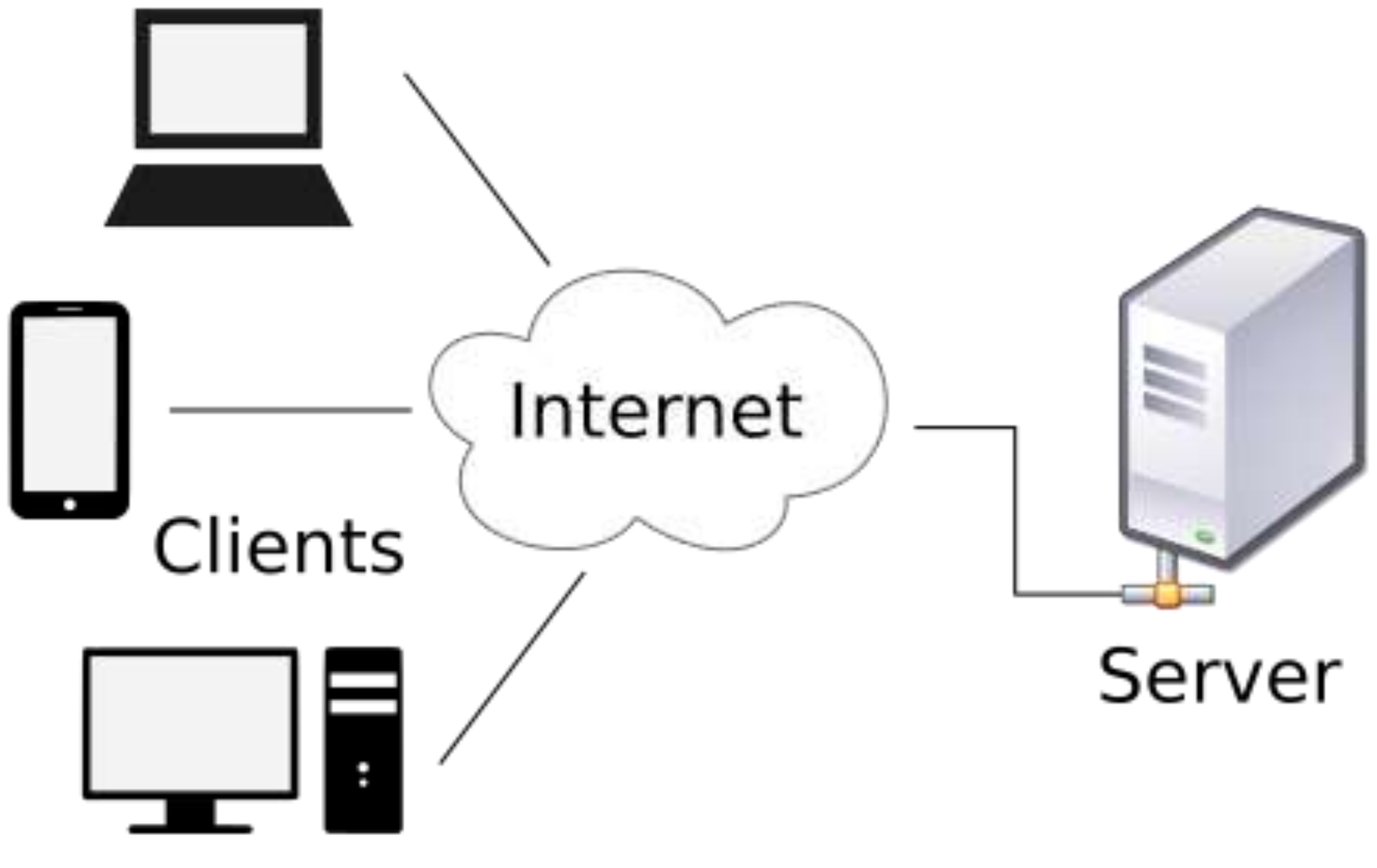
How can this be accomplished?

- Multiple processes?

Remark 1

Cannot easily use multiple processes since processes don’t share data.

Client — Server model



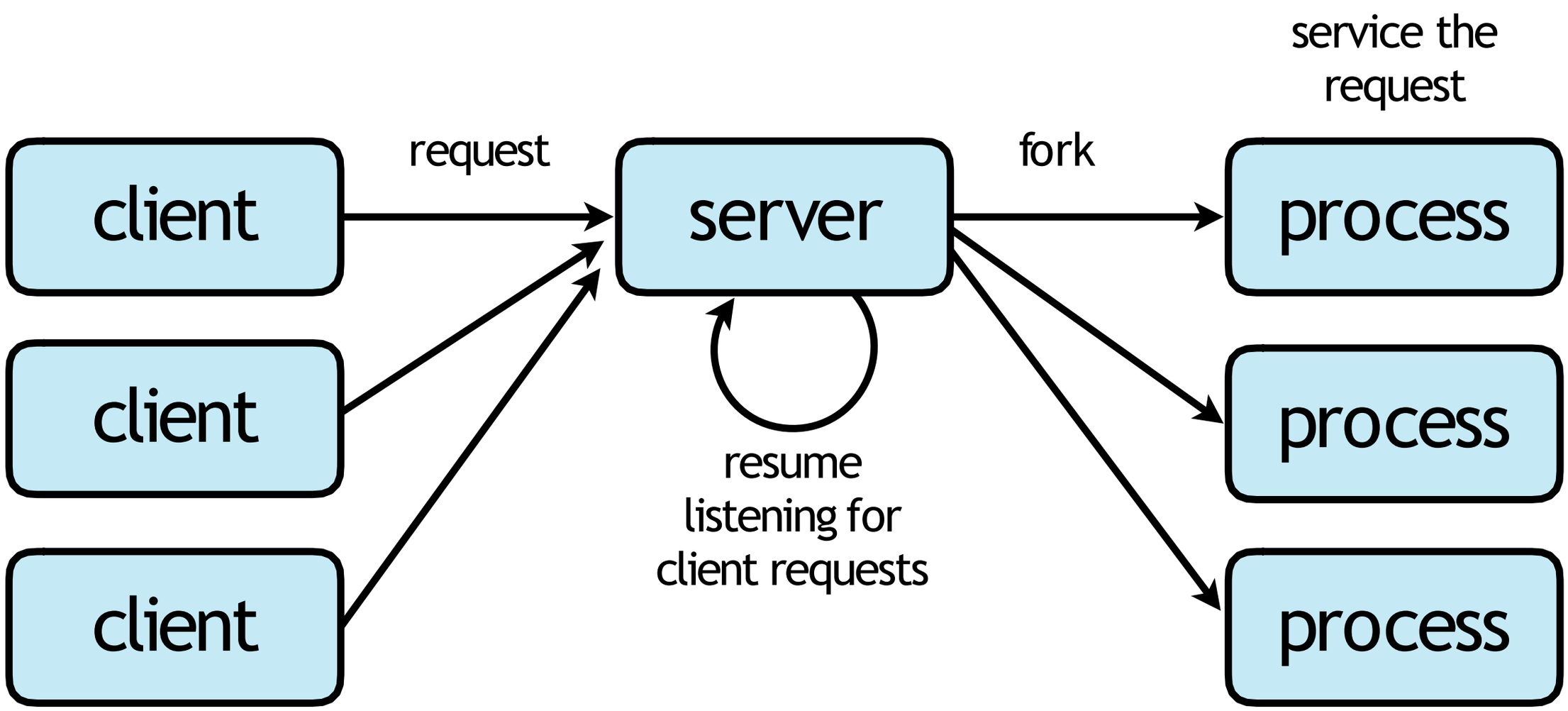
★ Often clients and servers communicate over a computer network on separate hardware …

★ … but both client and server may reside in the same system.

★ A **server** host runs one or more server programs which **share** their **resources with clients**.

★ **A client does not share any of its resources**, but requests a server's content or service function.

★ Clients therefore initiate communication sessions with servers which await incoming requests.



Remark 2

Creating a new process is time consuming and resource intensive.

The process

Stack

Heap

Static data

Text

main()

foo()

bar()

PC

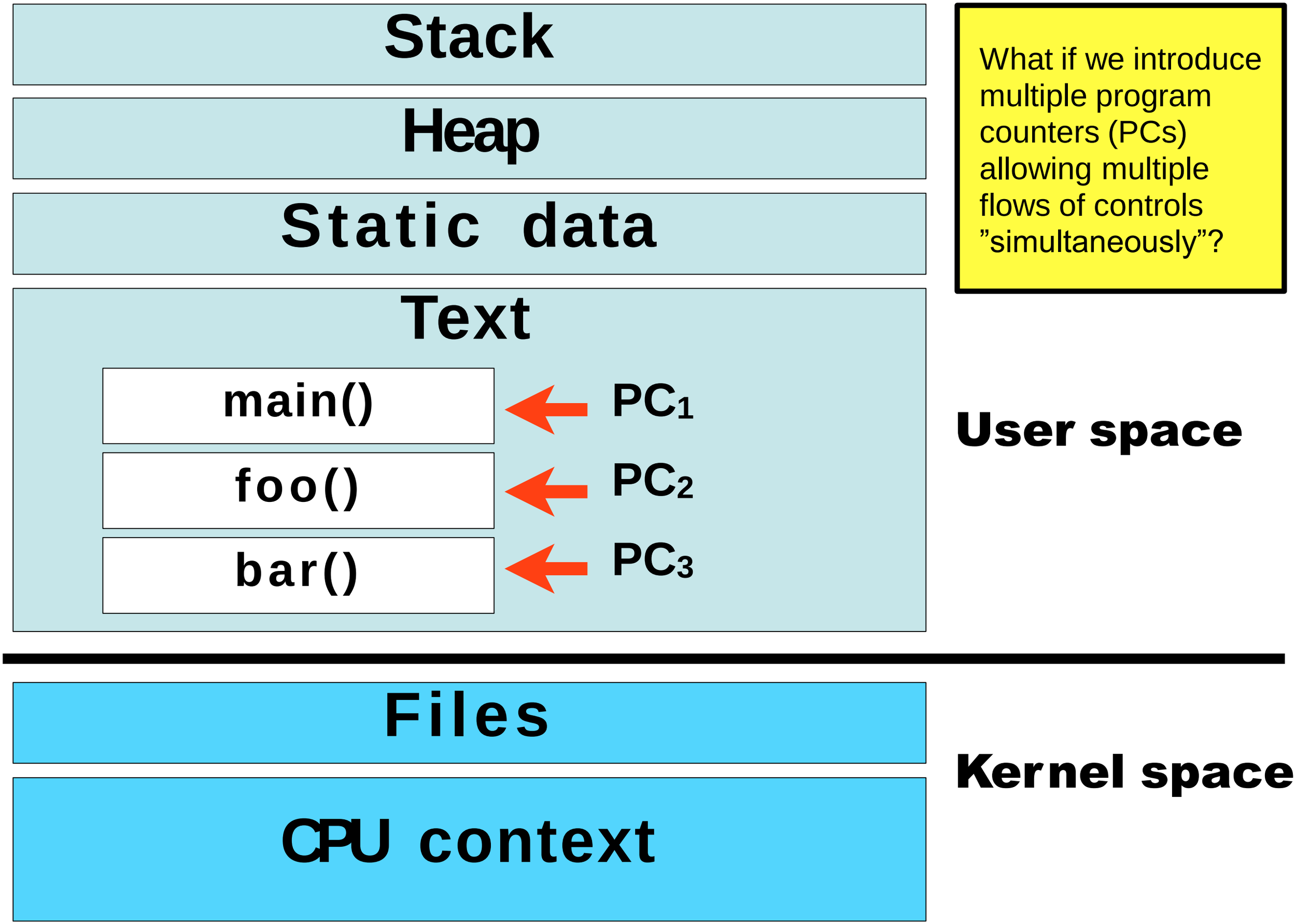
User space

Files

CPU context

PC (program counter) and other registers

Kernel space


Stack
Heap
Static data
Text
main()
foo()
bar()
PC1
PC2
PC3
What if we introduce multiple program counters (PCs) allowing multiple flows of controls ”simultaneously”?
User space
Files
CPU context
Kernel space

Stack ?

Heap ?

Static data ?

Text

main() ← PC_1

foo() ← PC_2

bar() ← PC_3

If multiple program counters are used, how does this affect the use of the **stack**, **heap**, **static data**, **files** and **registers** (context)?

User space

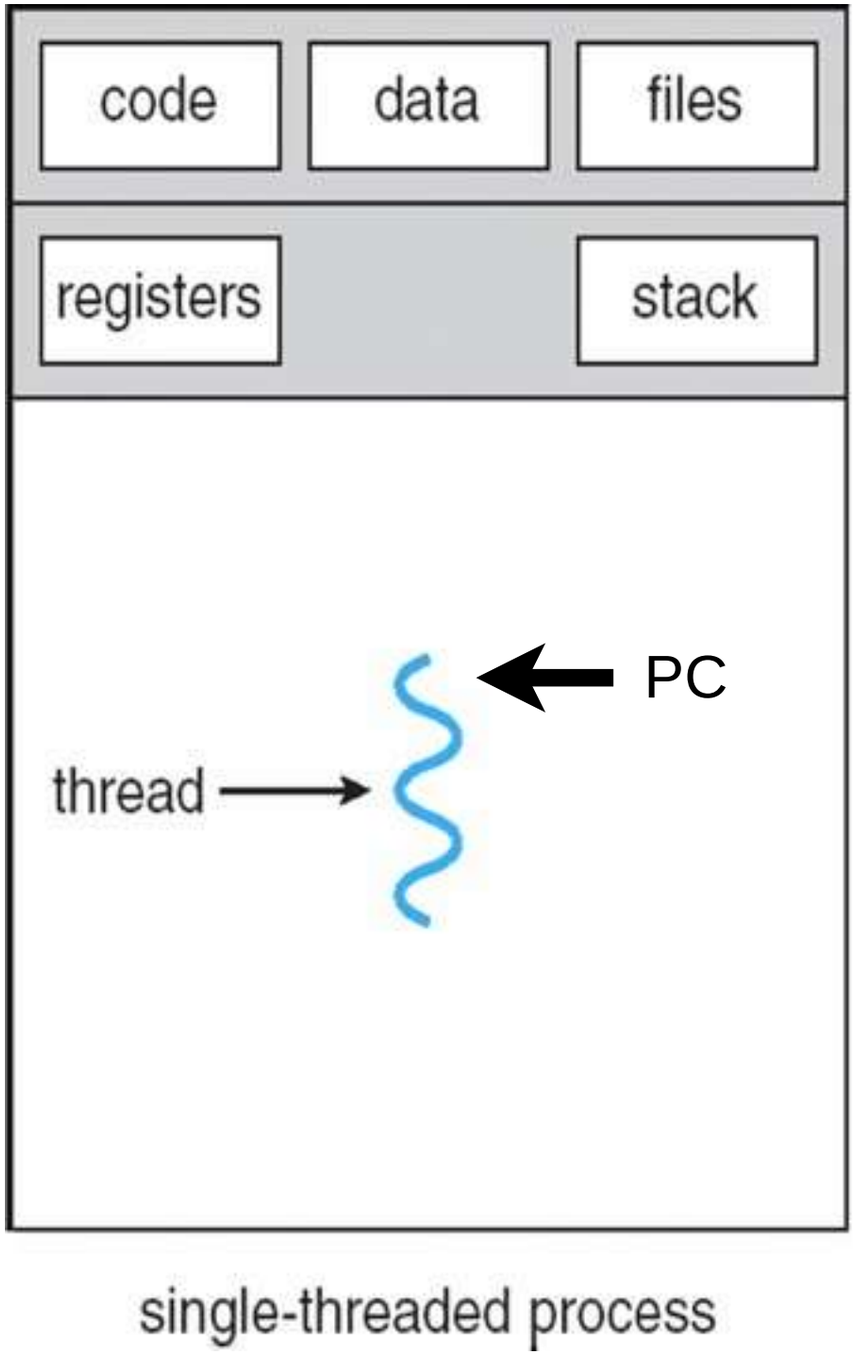
Files ?

CPU context ?

Kernel space

Threads

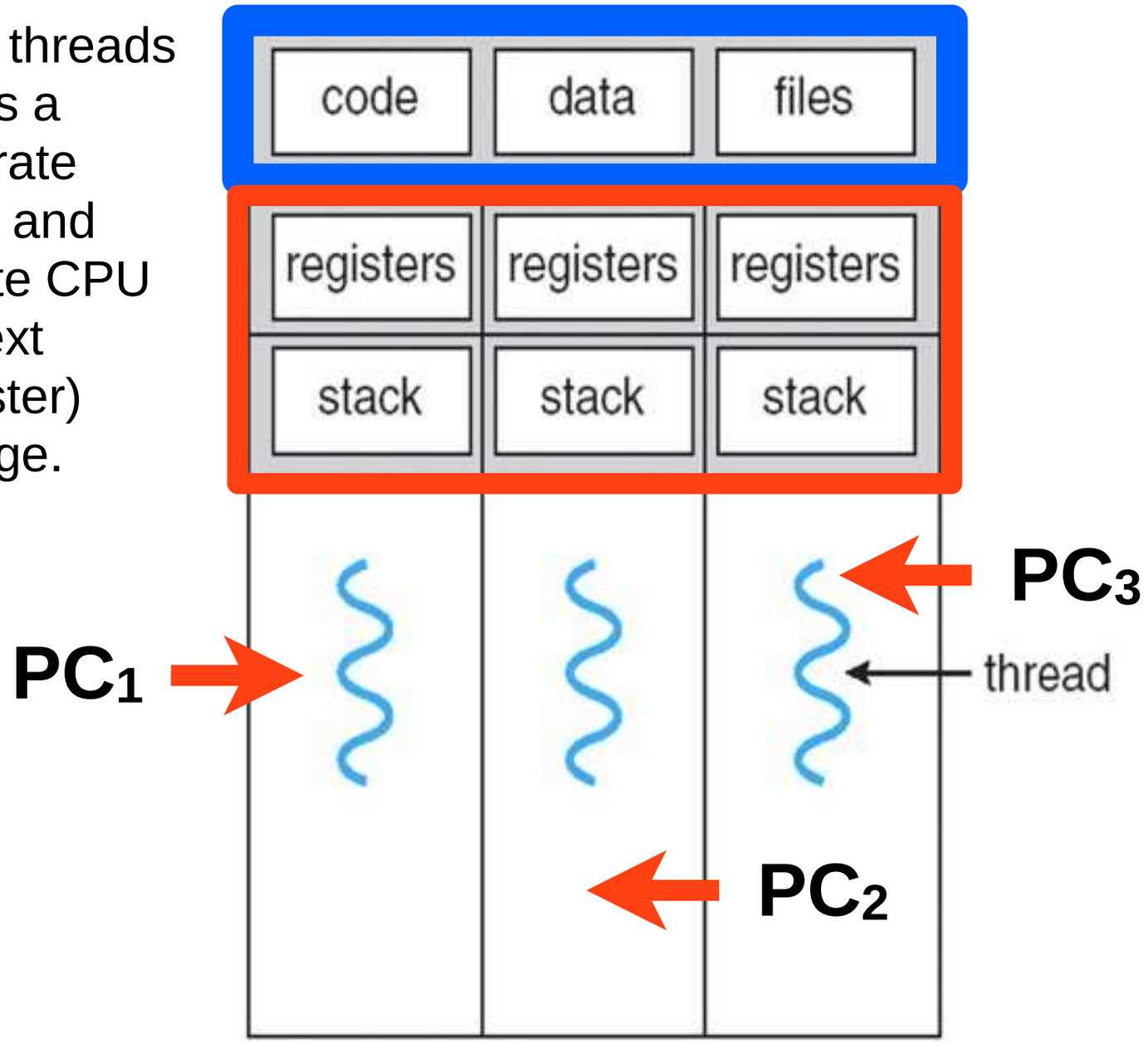
Single and multithreaded processes



single-threaded process

Each threads needs a separate stack and private CPU context (register) storage.

Threads share code, data and open files.

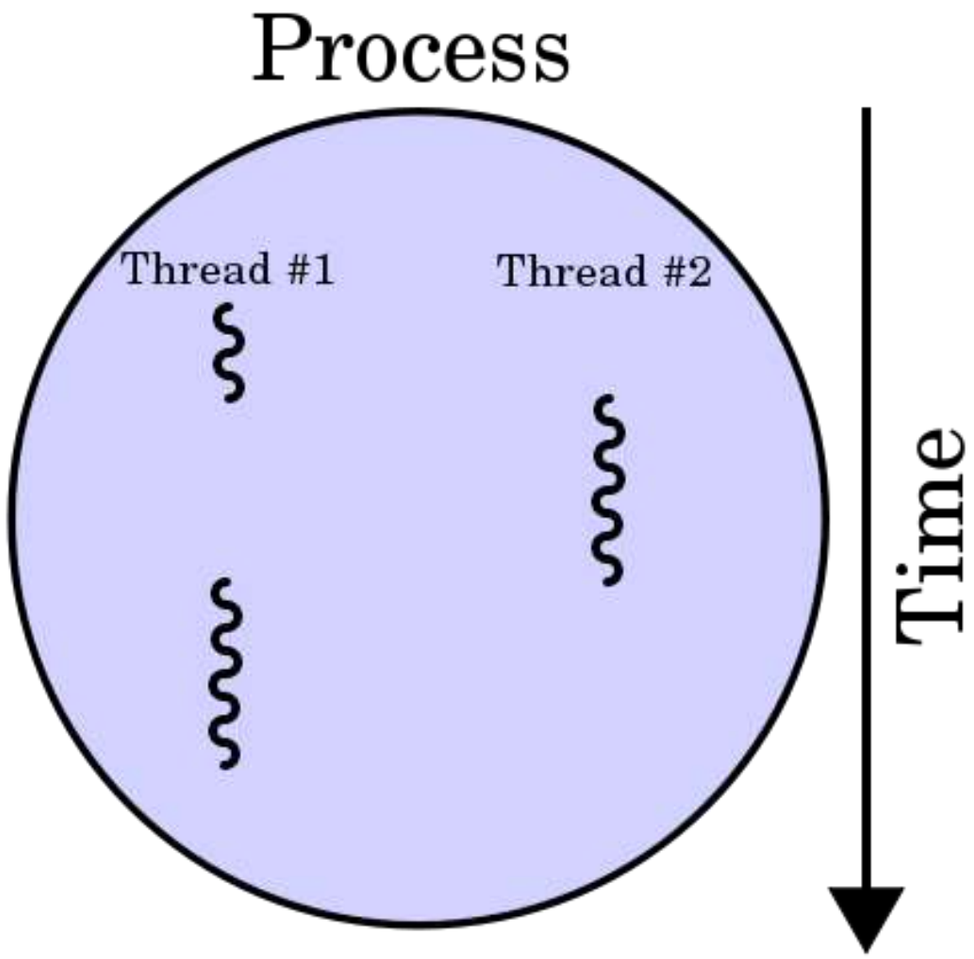


multi-threaded process

Note: this is an example of a process with three threads. Generally, a process may have n > 0 threads.

Thread

A thread of execution is the **smallest sequence of instructions** that can be **managed independently by a scheduler**, which is typically a part of the operating system.

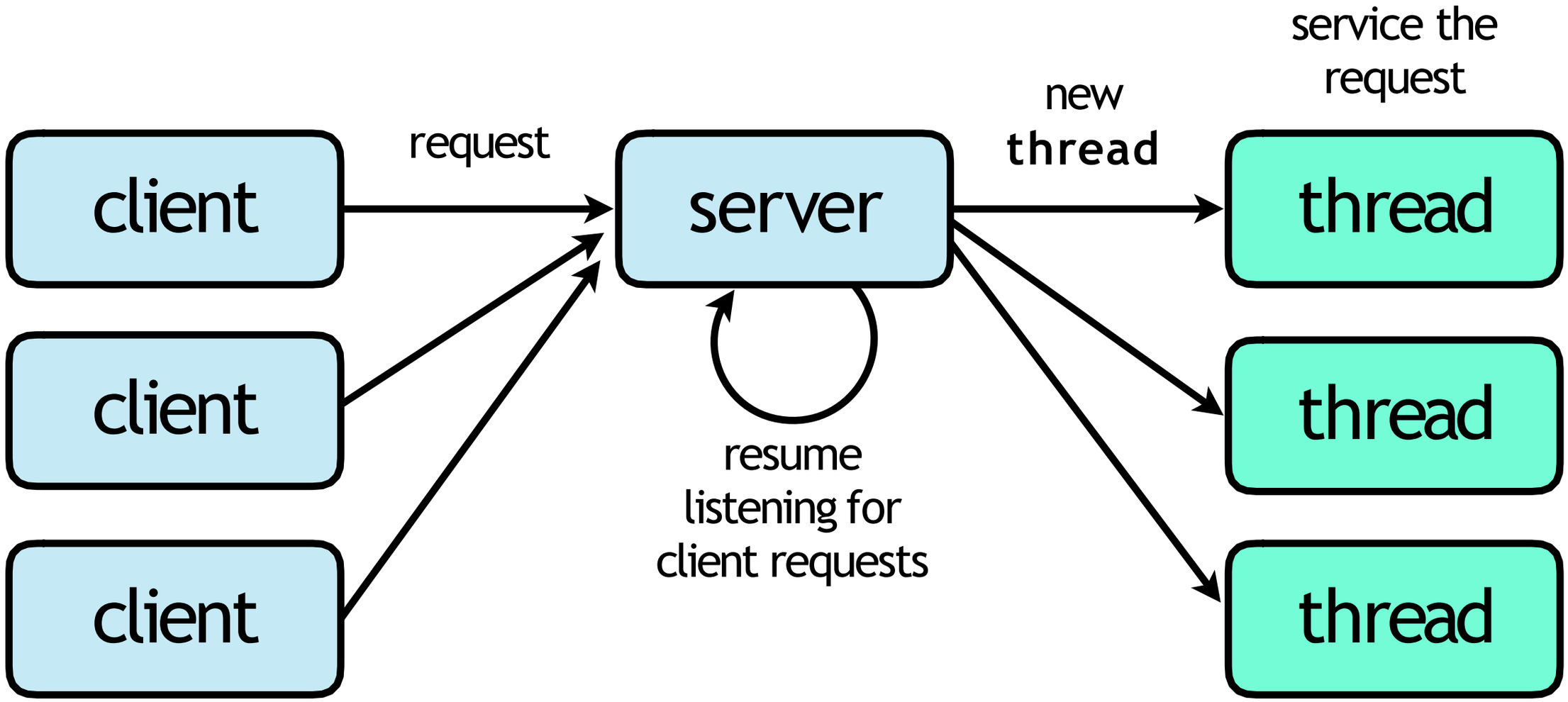


Multiple threads can exist within the same process:

- **Executing concurrently** (one starting before others finish)
- **Sharing** resources such as **memory**, while different processes do not share these resources.
- **Sharing** instructions (executable **code**) and memory (the values of its variables at any given moment).

A word processor may have to perform the following tasks *”at the same time”*, **using separate threads** for:

- Displaying graphics (GUI).
- Responding to key strokes.
- Spelling and grammar checking in the background.


client
client
client
request
server
resume
listening for
client requests
new
thread
service the
request
thread
thread
thread

Benefits of using threads

Responsiveness

Multithreading an interactive application may allow a program to continue running even if part of it is blocking or performing a lengthy operation, thereby increasing responsiveness to the user. '

Resource sharing

Processes may only share resources through techniques such as shared memory or message passing. Threads (by default) share memory and resources which allows an application to have several threads of execution in the same address space.

Economy

Allocating memory and resources for process creation is costly. Because threads share the resources of the process to which they belong, it is (at least in theory) more economical to create and context-switch threads.

Scalability

The benefits of multithreading can be greatly increased in a multiprocessor (multicore) architecture, where threads may be running in parallel on different processors (cores).

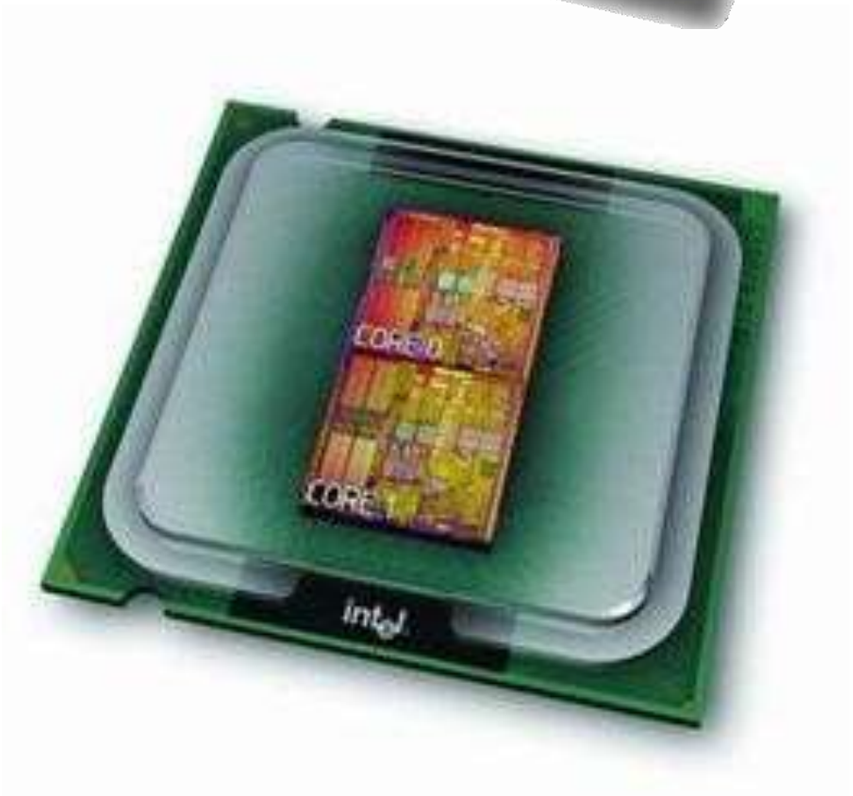
Single-Core CPUs

The core is the part of the processor that actually performs the reading and executing of instructions.

★ Processors were originally developed with only one core.

★ A **Single-core** processor can process only **one instruction at a time**.

Multi-Core CPUs

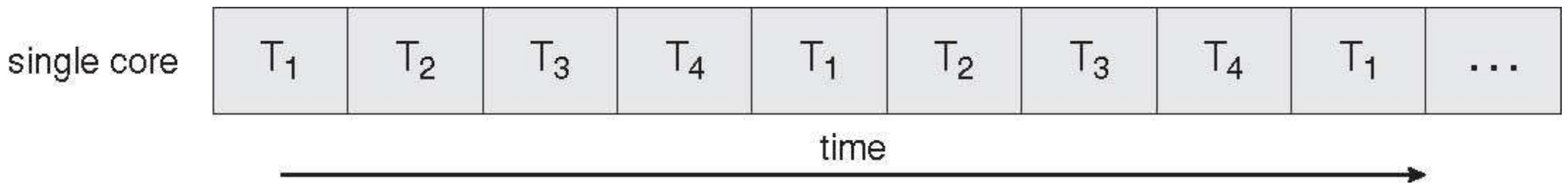


- A multi-core processor is composed of two or more independent cores.
- One can describe it as an integrated circuit which has two or more individual processors (called cores in this sense).

To make full use of multicore computers, programmers will need to learn how to use concurrent programming.

Concurrent execution of threads

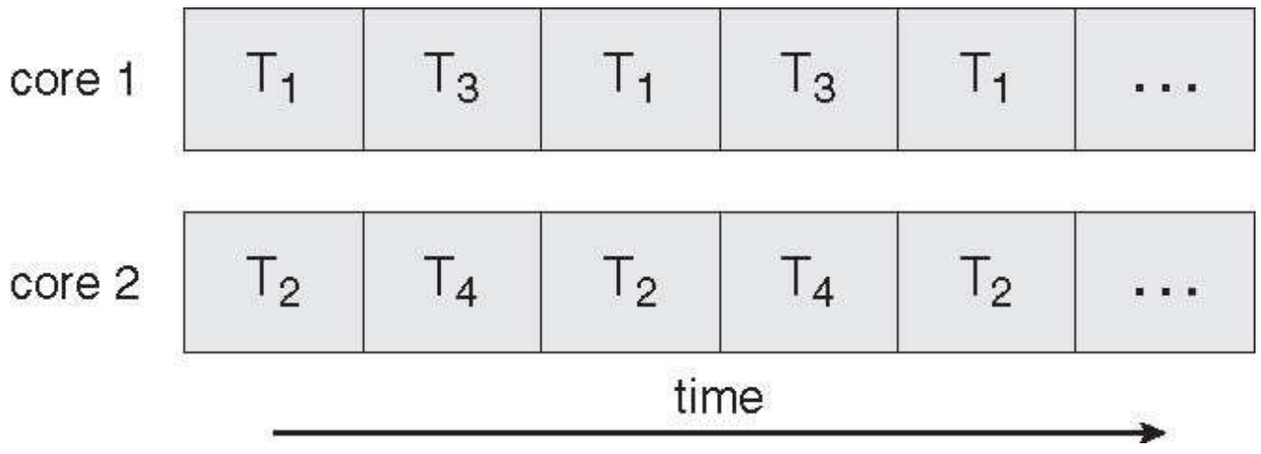
On a single core CPU



Threads take turn executing on the single CPU core. By switching fast enough between the threads they appear to be executing ”at the same time”.

concurrent ≠ parallel

On a dual core CPU

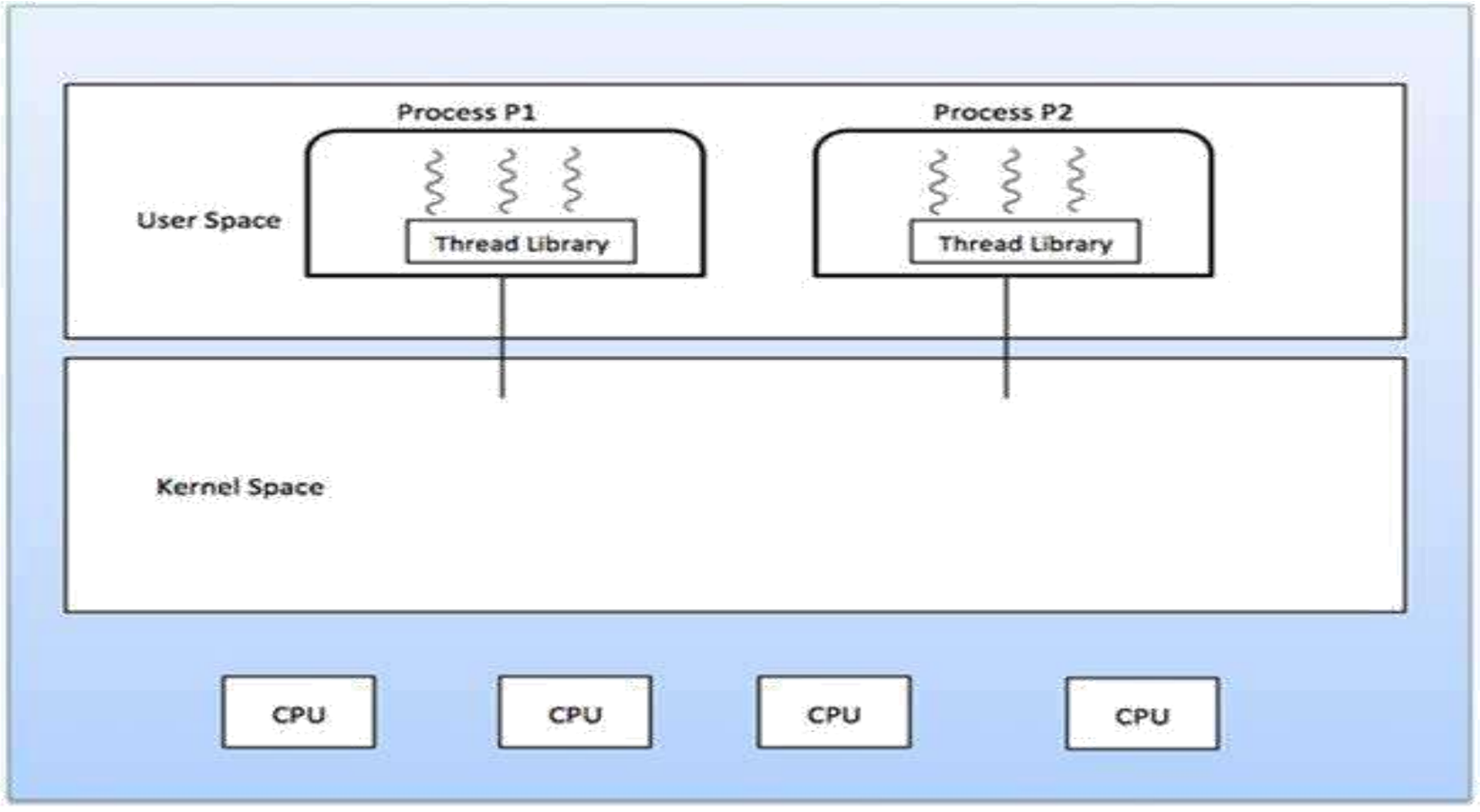


Remark 3

At most two threads can execute in parallel (truly at the same time) on a dual core CPU. On every core, threads take turn executing just as on a single core CPU.

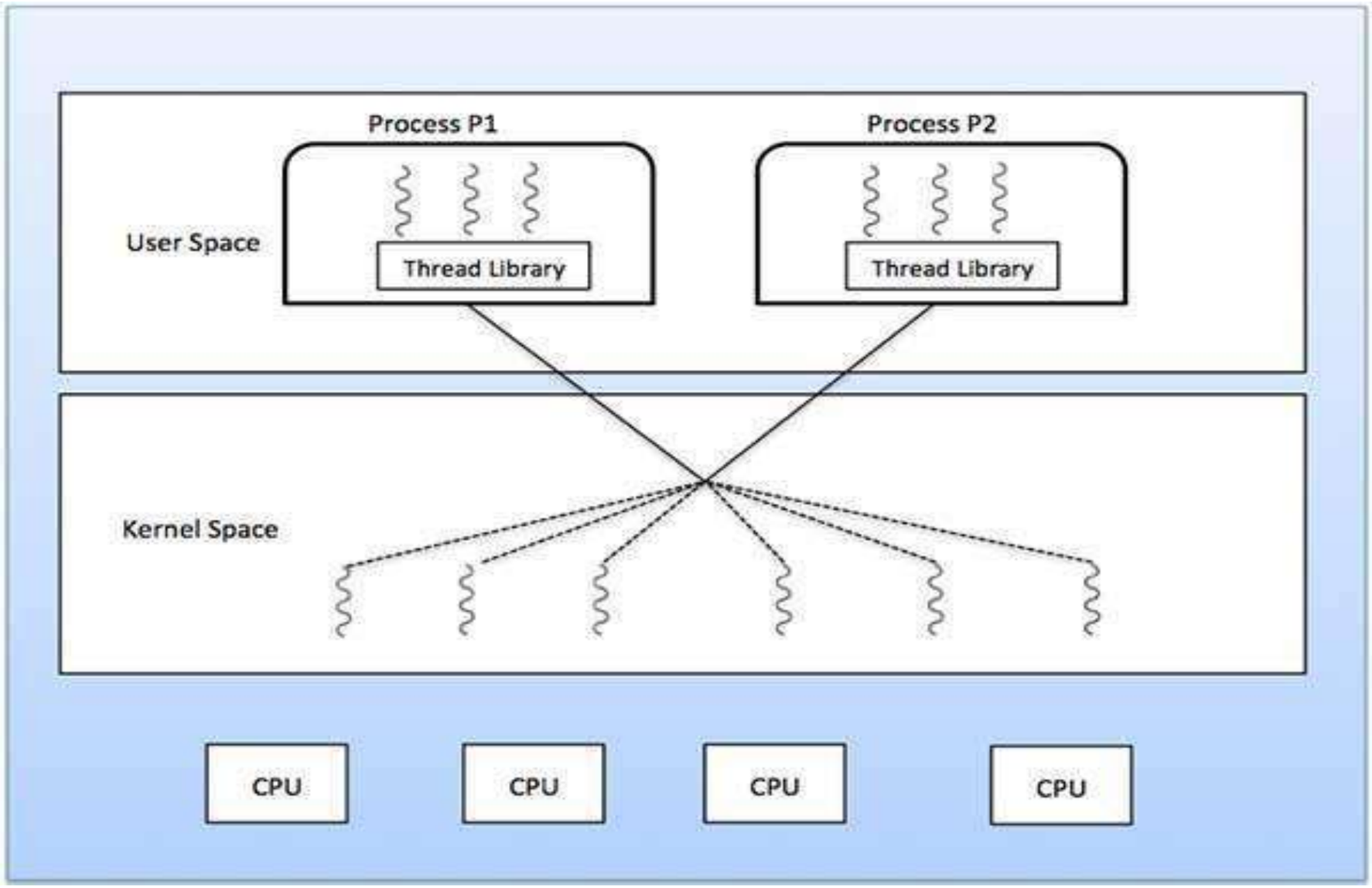
Types of Thread

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.

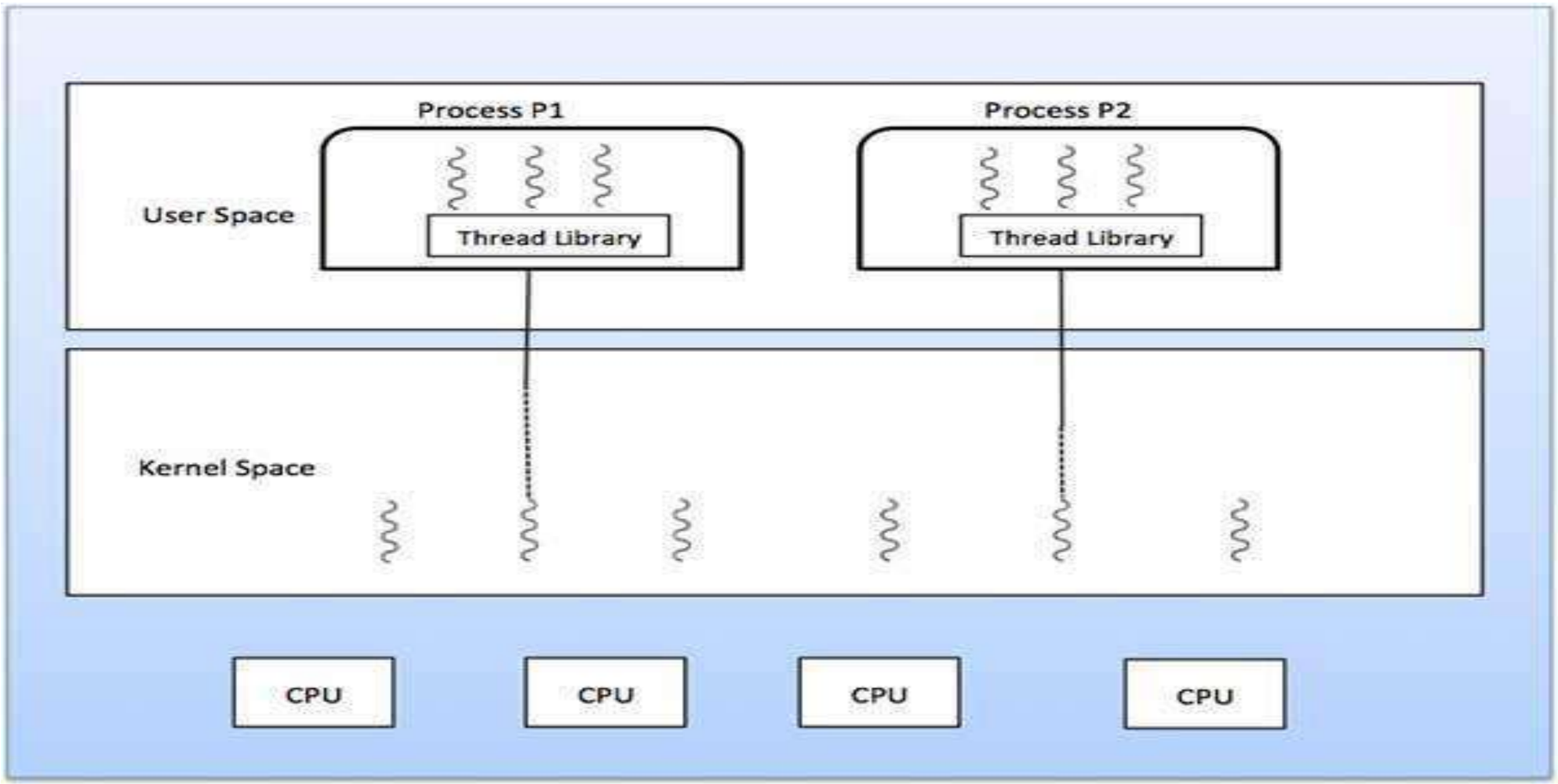


Multithreading Models

Many-to-Many Model



Many-to-One Model



One-to-One Model