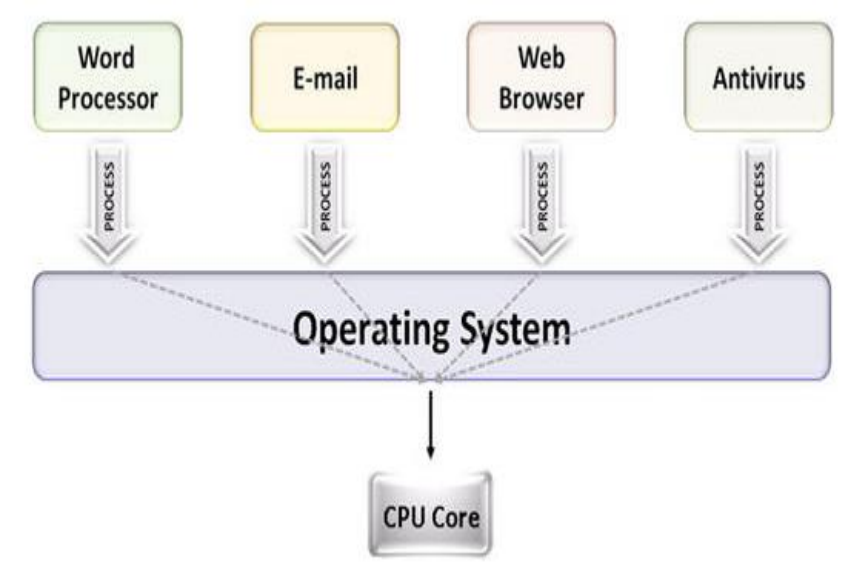
CS35L – Spring 2019

Slide set:	6.1
Slide topics:	Multithreaded Performance
Assignment:	6

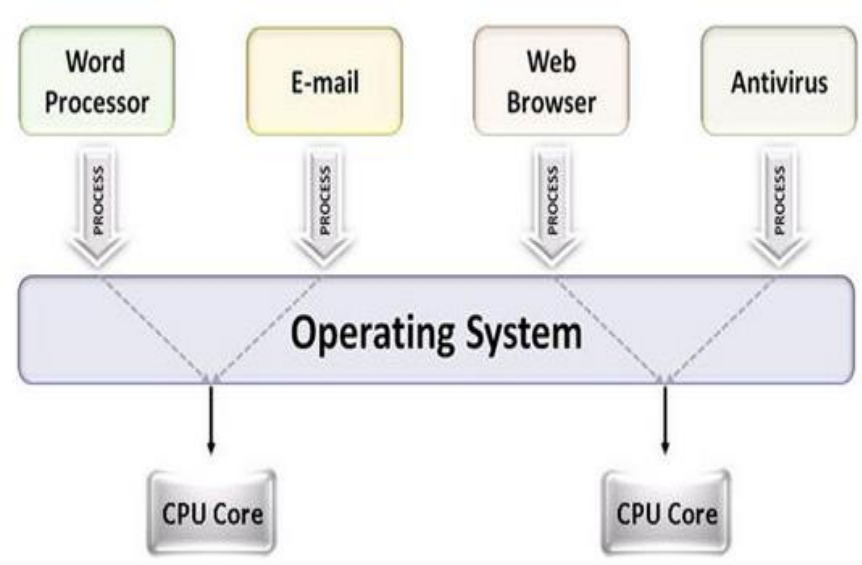
Multiprocessing

- The use of multiple CPUs/cores to run multiple tasks simultaneously

Uniprocessing system



Multiprocessing system



Parallelism

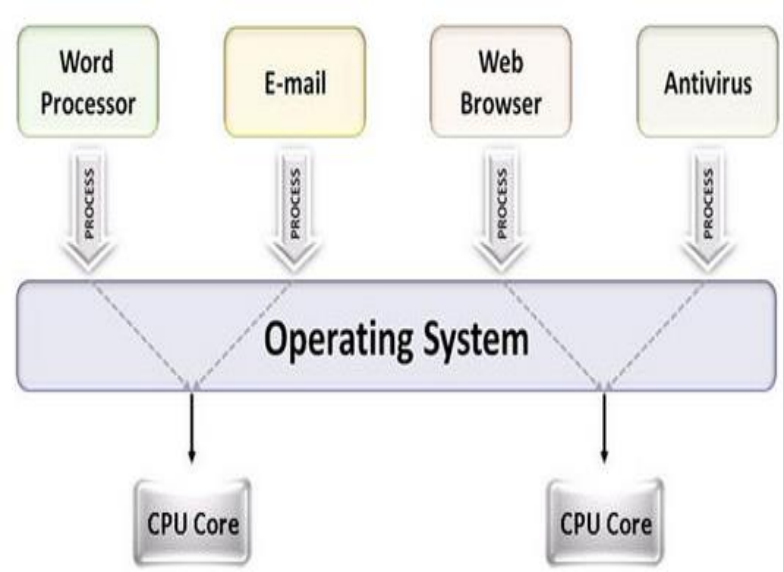
- Executing several computations simultaneously to gain performance
- Different forms of parallelism
  - **Multitasking**
    - Several processes are scheduled alternately or possibly simultaneously on a multiprocessing system
  - **Multithreading**
    - Same job is broken logically into pieces (threads) which may be executed simultaneously on a multiprocessing system

What is a thread?

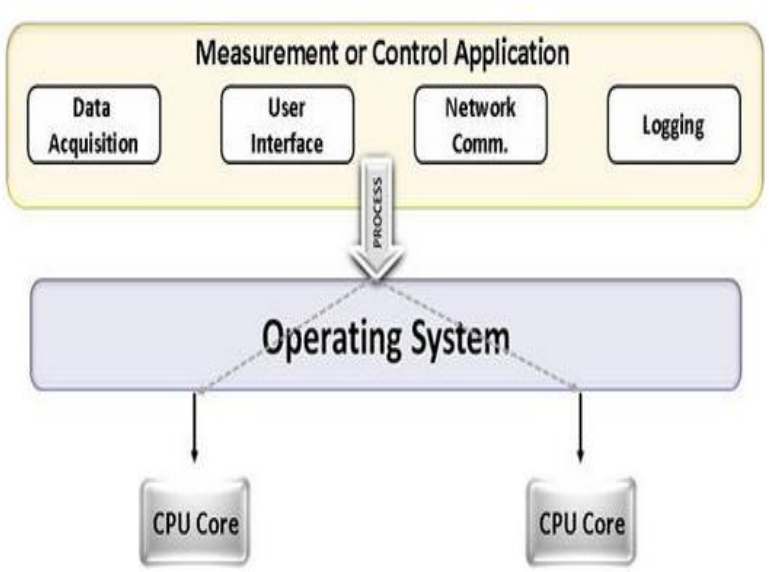
- A flow of instructions, path of execution within a process
- The smallest unit of processing scheduled by OS
- A process consists of at least one thread
- Multiple threads can be run on:
  - **A uniprocessor (time-sharing)**
    - Processor switches between different threads
    - Parallelism is an illusion
  - **A multiprocessor**
    - Multiple processors or cores run the threads at the same time
    - True parallelism

Multitasking vs. Multithreading

Multitasking

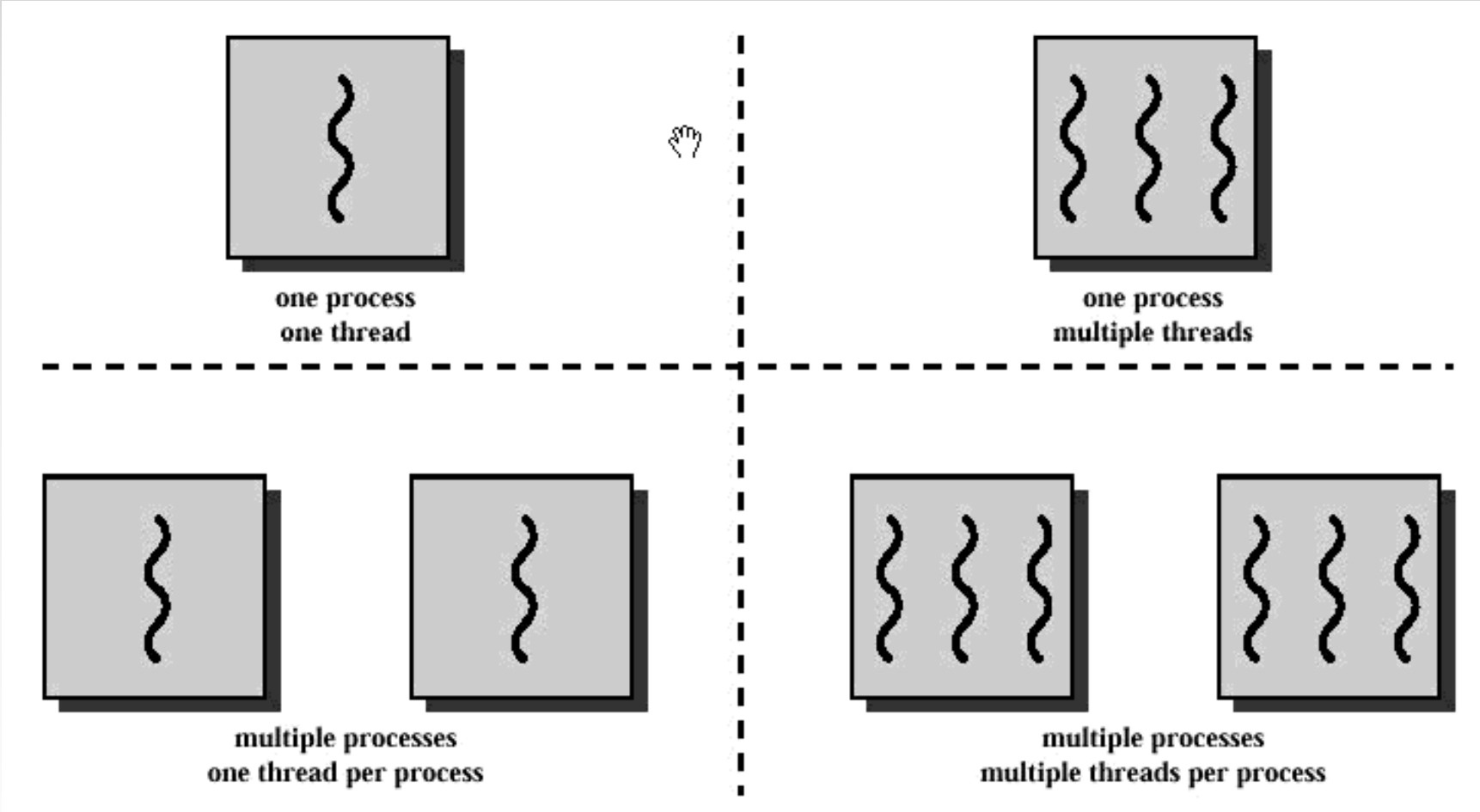


Multithreading

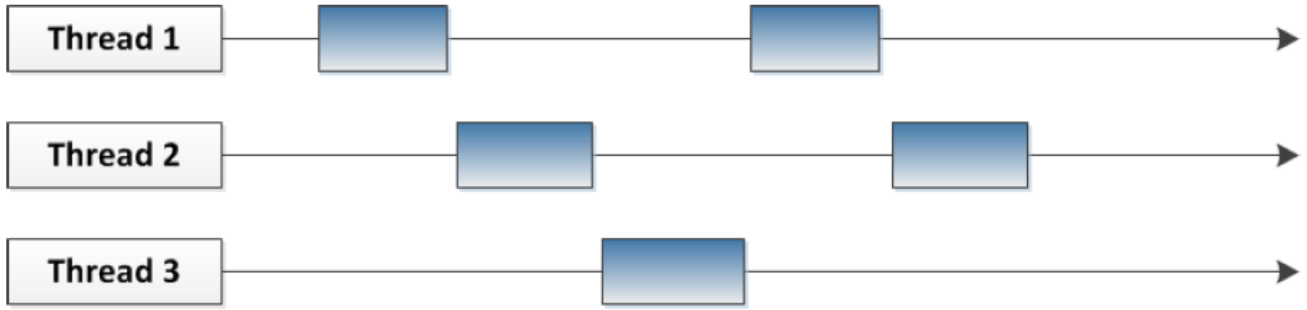


Multitasking is sharing of computing resources(CPU, memory, devices, etc.) among processes

Multithreading is sharing of computing resources among threads of a single process.

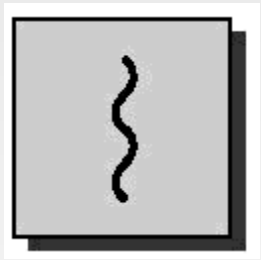

one process
one thread
one process
multiple threads
multiple processes
one thread per process
multiple processes
multiple threads per process

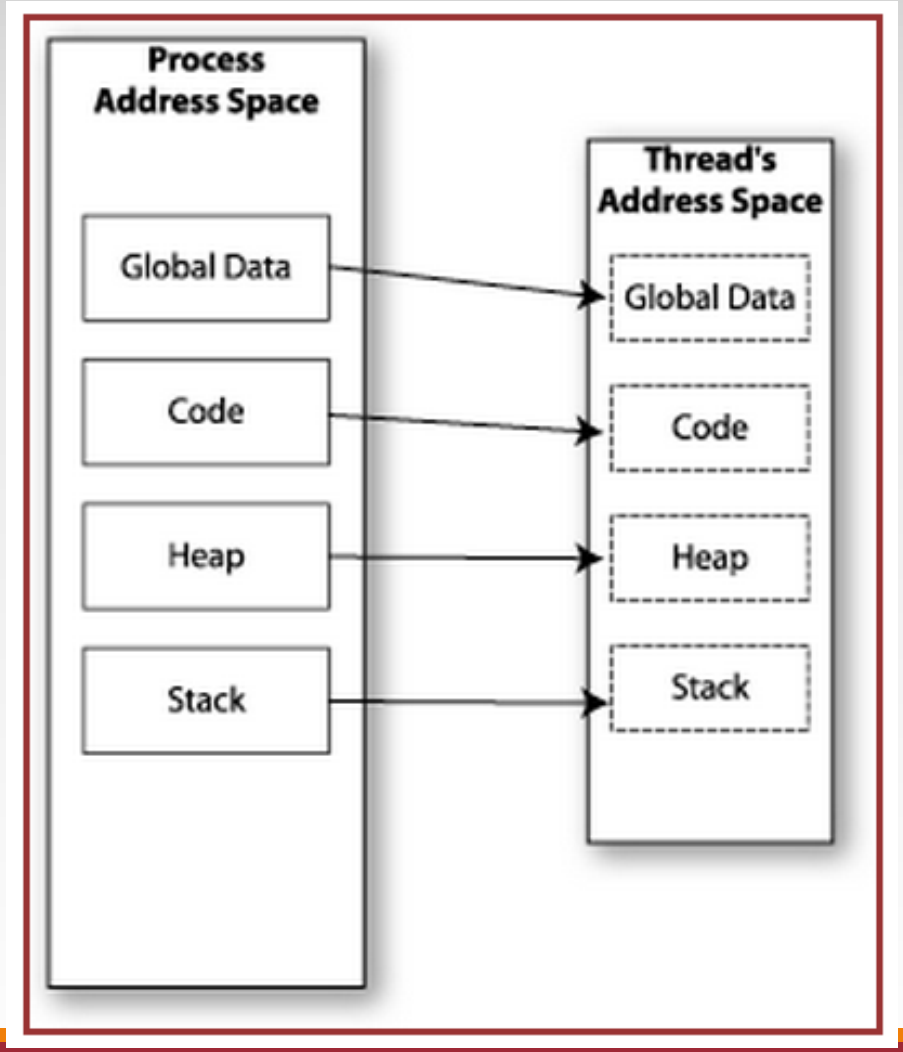
Multiple threads sharing a single CPU



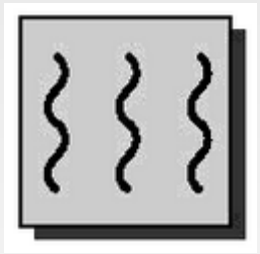
Multiple threads on multiple CPUs

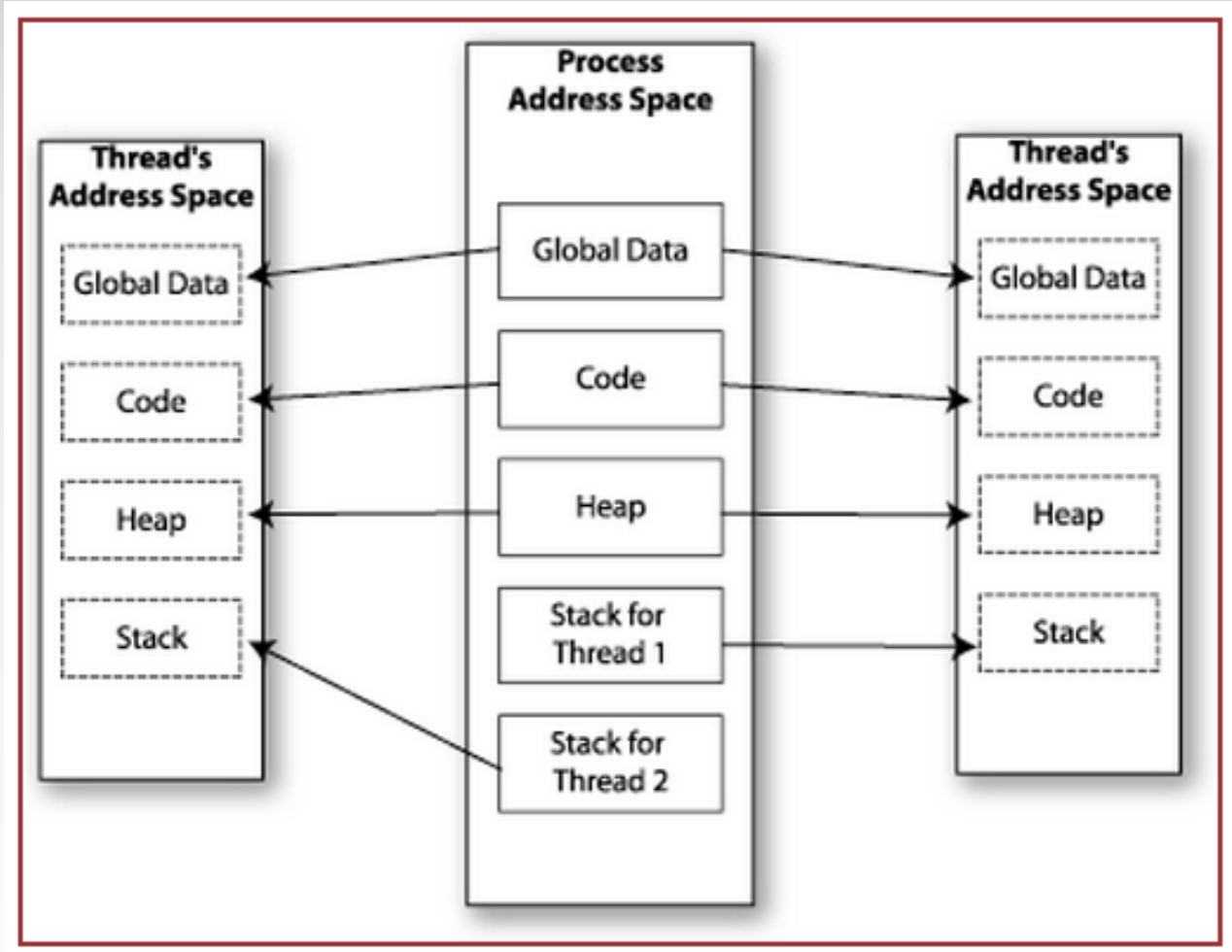
Memory Layout: Single-Threaded Program





Memory Layout: Multithreaded Program





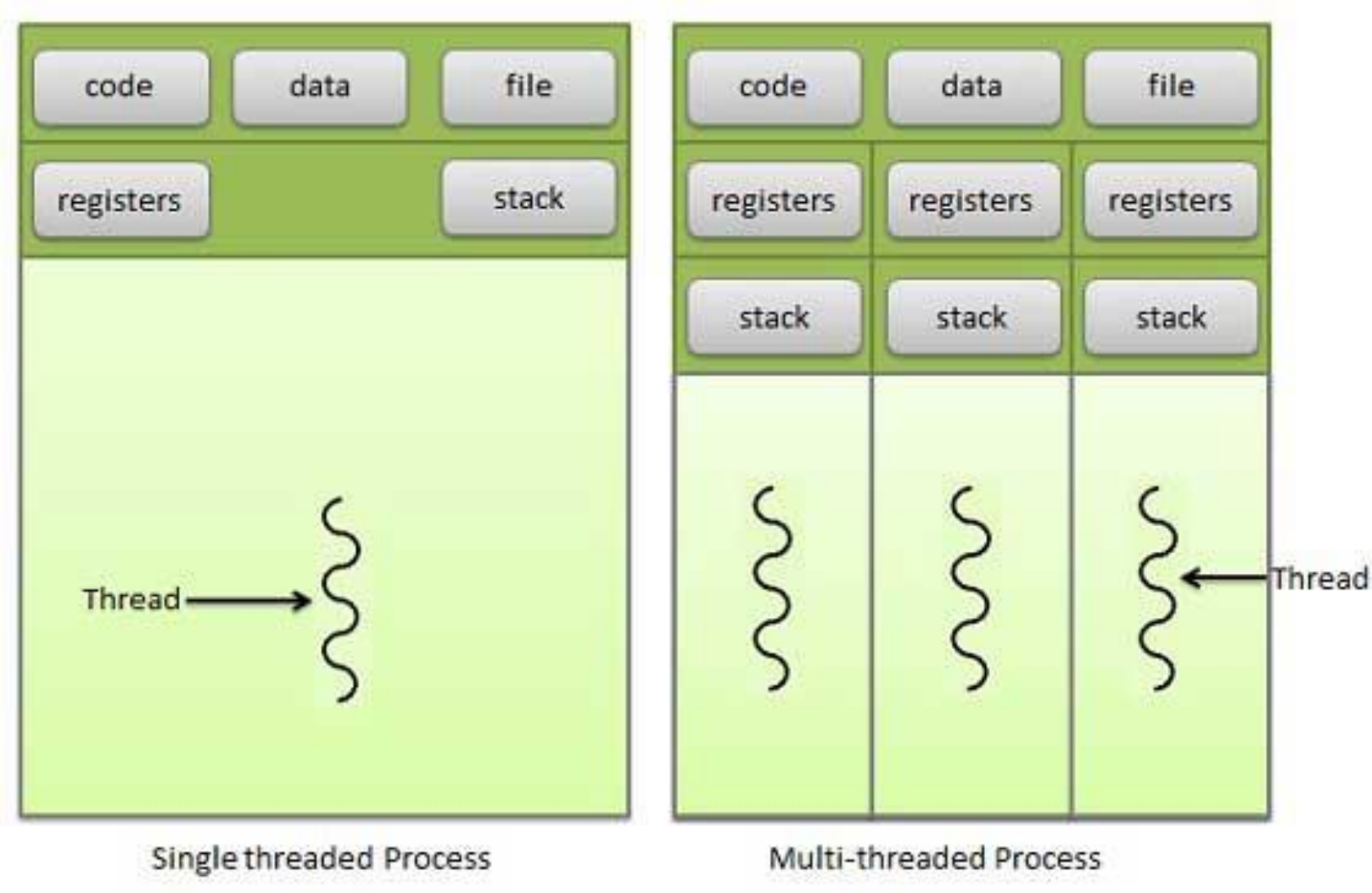
Multitasking

- $ tr ’abc’ ‘xyz’ | sort -u | comm -23 file1 -
  - Process 1 (tr)
  - Process 2 (sort)
  - Process 3 (comm)
- Each process has its own address space
- How do these processes communicate?
  - Pipes/System Calls

Multithreading

- Threads share all of the process's memory except for their stacks
- => Data sharing requires no extra work (no system calls, pipes, etc.)

Multithreading Memory Layout

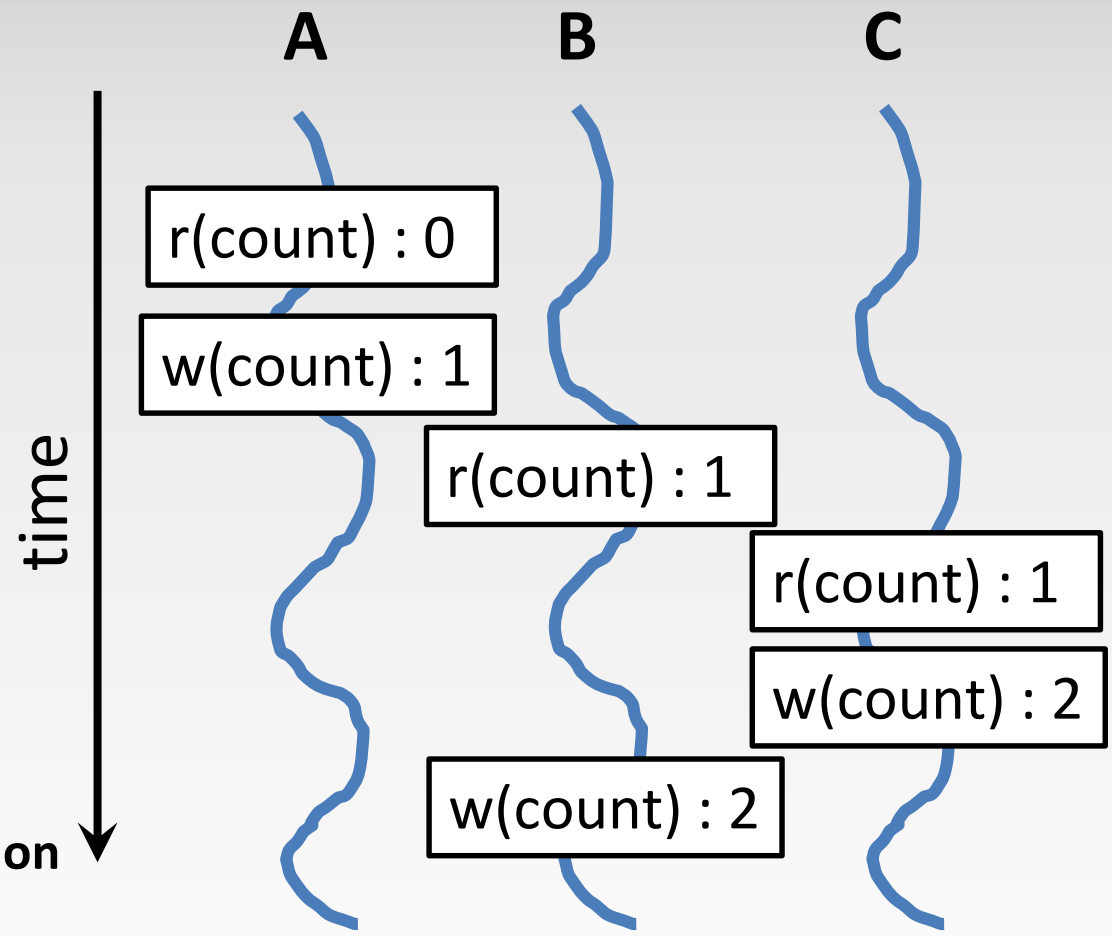


Shared Memory

- Makes multithreaded programming
  - **Powerful**
    - can easily access data and share it among threads
  - **More efficient**
    - No need for system calls when sharing data
    - Thread creation and destruction less expensive than process creation and destruction
  - **Non-trivial**
    - Have to prevent several threads from accessing and changing the same shared data at the same time (synchronization)

Race Condition

```
int count = 0;
void increment()
{
  count = count + 1;
}
```



Result depends on order of execution

=> Synchronization needed

Multithreading & Multitasking: Comparison

- **Multithreading**
  - Threads share the same address space
    - Light-weight creation/destruction
    - Easy inter-thread communication
    - An error in one thread can bring down all threads in process
- **Multitasking**
  - Processes are insulated from each other
    - Expensive creation/destruction
  - Expensive IPC (interprocess communication)
    - An error in one process cannot bring down another process