Operating system concepts

Threads & Concurrency
Slides Set #6

By Prof K R Chowdhary

JNV University

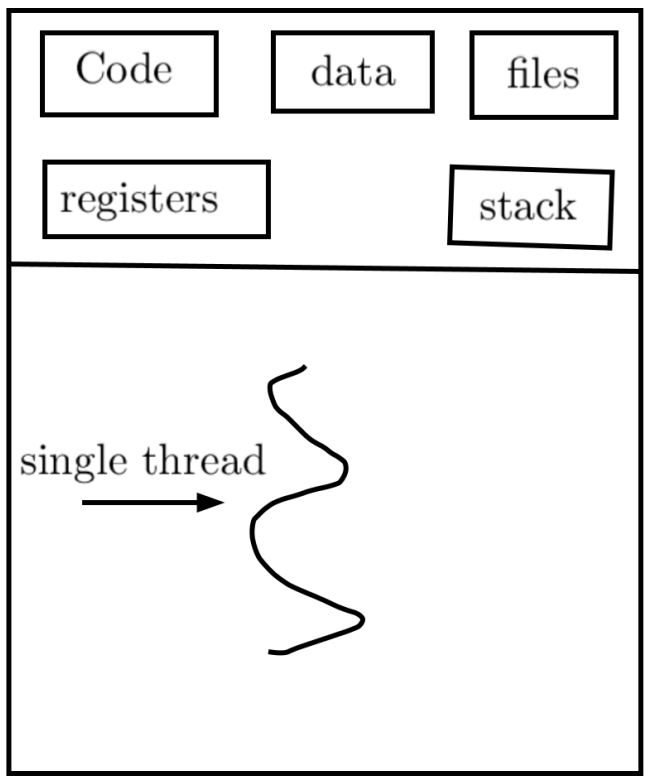
2023

Thread definition

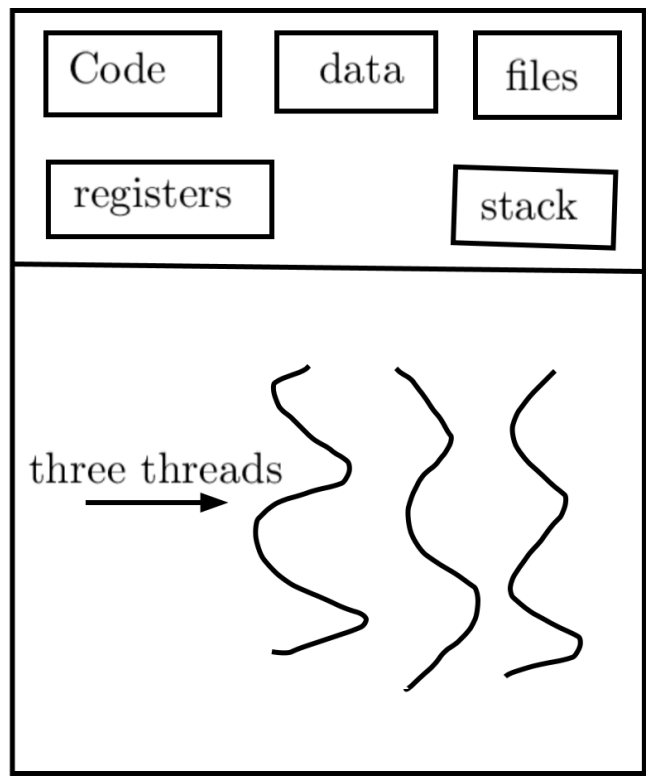
A thread is also known as a lightweight process. The idea is to achieve parallelism by dividing a process into multiple threads. For example, in a browser, multiple tabs can be different threads. MS Word uses multiple threads: one thread to format the text, another thread to process inputs, etc.

Threads and concurrency

- By default, a process executes a program with a single thread of control,
- All the modern OS have facility of multi-thread of control
- Thread is a basic unit of CPU utilization, it consists of a thread ID, a program counter, a register set, and a stack.



Single-threaded process



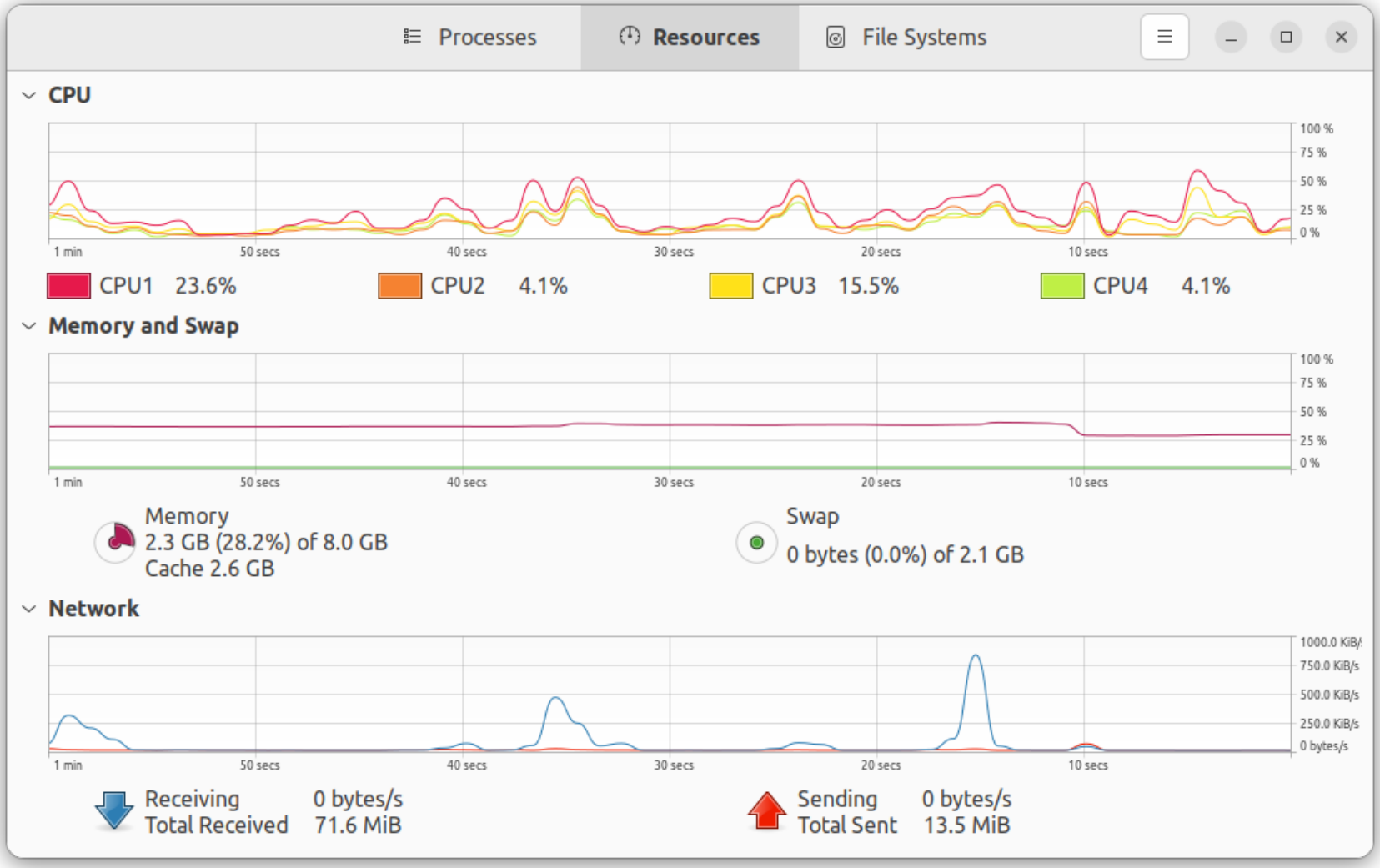
multi-threaded process

Gnome System Monitor

gnome-system-monitor: view and control processes

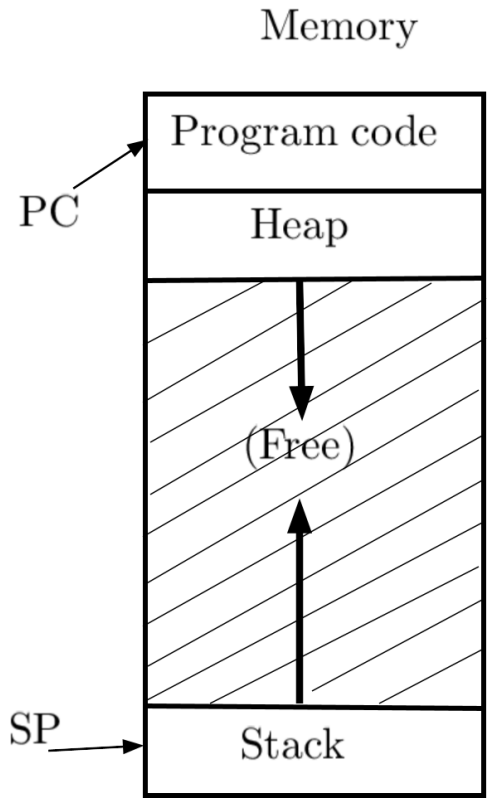
- The gnome-system-monitor allows you to view and control the processes running on your system.
- You can access detailed memory maps, send signals, and terminate the processes.
- In addition, the gnome-system-monitor provides an overall view of the resource usage on your system, including memory and CPU allocation.

Gnome System Monitor...



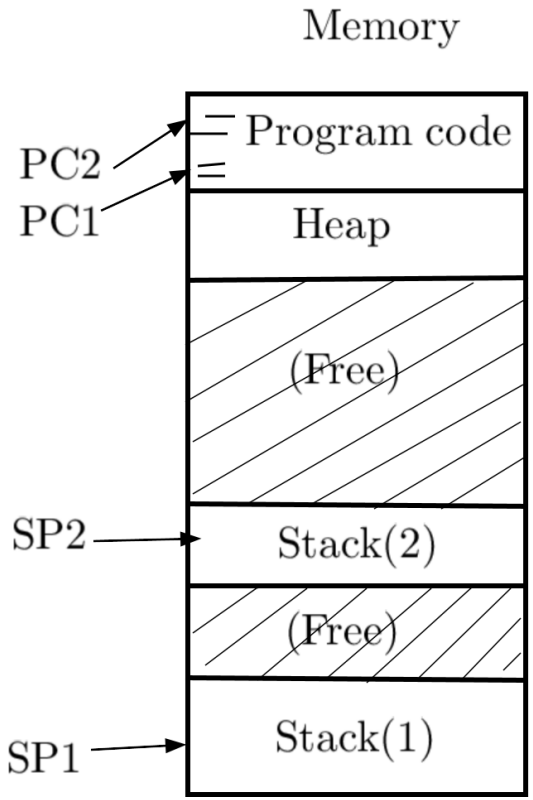
A single threaded Process

- Process execution:
  - PC points to current instruction being run
  - SP points to stack frame of current function call
- A program can have multiple threads of execution
- What is thread?



Multithreaded Process

- A thread is like another copy a process that executes independently
- Threads share the same address space (i.e., code, heap)
- each thread has separate PC
  - Each thread may run over different part of the program
- Each thread has separate stack for independent function calls



Process vs. threads

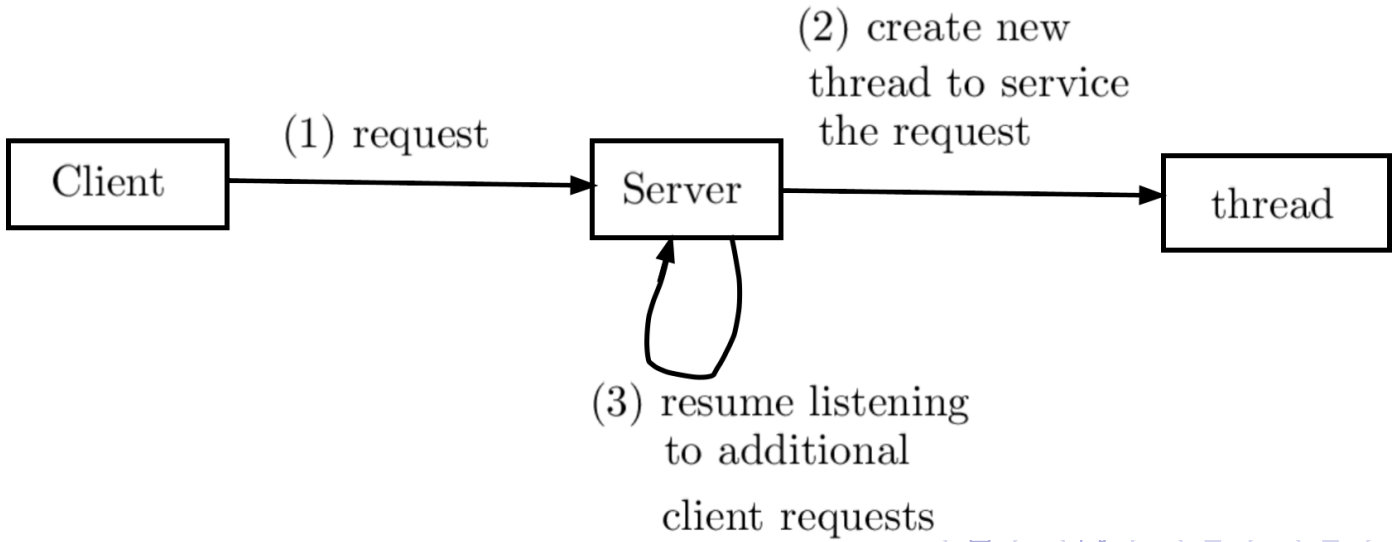
- Parent process P forks a child process C:
  - C is mirror image of P,
  - P and C do not share any memory,
  - Need complicated IPC (inter-process communication) mechanisms to communicate with each other,
  - Extra copies of code, data is created in memory.
- Parent process P executes two threads T1 and T2:
  - T1 and T2 share parts of address space,
  - Global variables can be used for communication,
  - Smaller memory footprints.
- Threads are like separate processes, except they share the *same address space*

Why threads?

- Parallelism: a single process can effectively utilize multiple CPU cores
  - *Concurrency*: running multiple threads/processes at the same time, even on single CPU core, by interleaving their executions
  - *Parallelism*: running multiple threads/processes in parallel over different CPU cores
- Even if no parallelism, concurrency of threads ensures effective use of CPU when one of the threads blocks (e.g., for I/O)

Examples: Threads and concurrency

- A web browser may have one thread to get data from network, while other thread may display the data
- E.g., MS word: one thread to respond key strokes, other for display on screen, other for spelling check, other for grammar check, one for display graphics.
- If the web server ran as a traditional single-threaded process, it would be able to service only one client at a time,
- One solution is to have the server run as a single process that accepts requests. When the server receives a request, it creates a separate process to service that request.



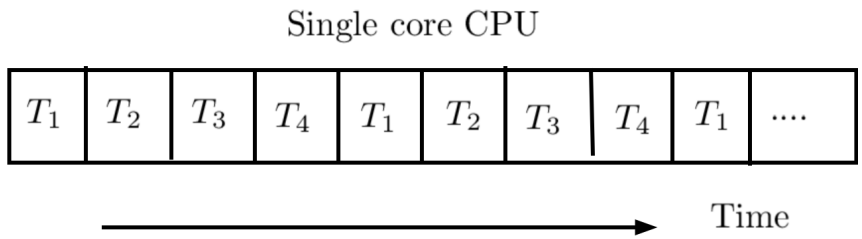
Advantages of Threads

- Threads are also used for remote procedure call (RPC) that provide inter-process communication.
- Most operating-system kernels are now multi-threaded. Several threads operate in the kernel,
- Advantages of threads:
  - *Responsiveness*. Multithreading an interactive application may allow a program to continue running even if part of it is blocked
  - *Resource sharing*: The benefit of sharing code and data is that it allows an application to have several different threads of activity within the same address space.
  - *Economy*: for example, *creating a process is about thirty times slower than is creating a thread,* and context switching is about five times slower.
  - *Scalability*: The benefits of multi-threading can be even greater in a multiprocessor architecture

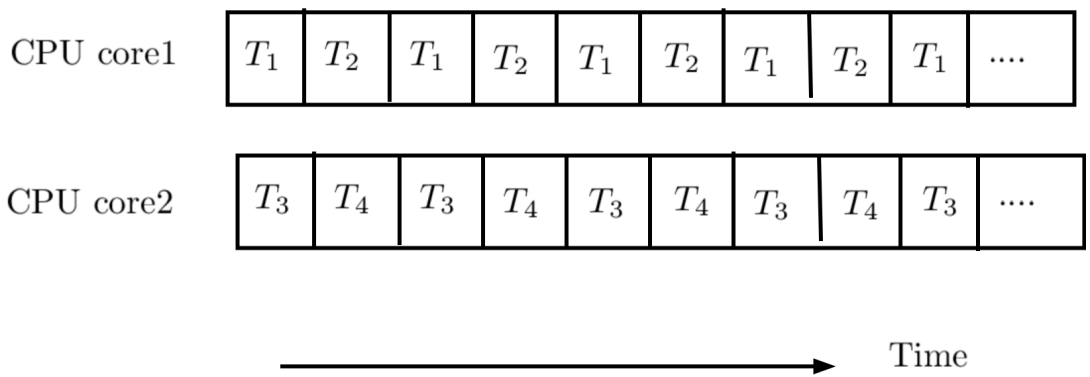
Multi-threads on single core vs. on multi-core systems

Earlier in the history of computer design, in response to the need for more computing performance, single-CPU systems evolved into multi-CPU systems.

- Concurrent execution on a single-core system vs



- Parallel execution on a multi-core system.



Creating a thread

- A thread library provides the programmer with an API for creating and managing threads.
- There are two primary ways of implementing a thread library. The *first approach* is to provide a library entirely in user space with no kernel support.
- *Other approach:* Implement a kernel-level library supported directly by the operating system.
- Three main thread libraries are in use today: *POSIX (Portable Operating System Interface) Pthreads*, *Windows*, and *Java*. Pthreads, the threads extension of the POSIX standard,