CS3.301 Operating Systems and Networks

Concurrency - Introduction

Karthik Vaidhyanathan

https://karthikvaidhyanathan.com

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

SERC
Software Engineering Research Centre

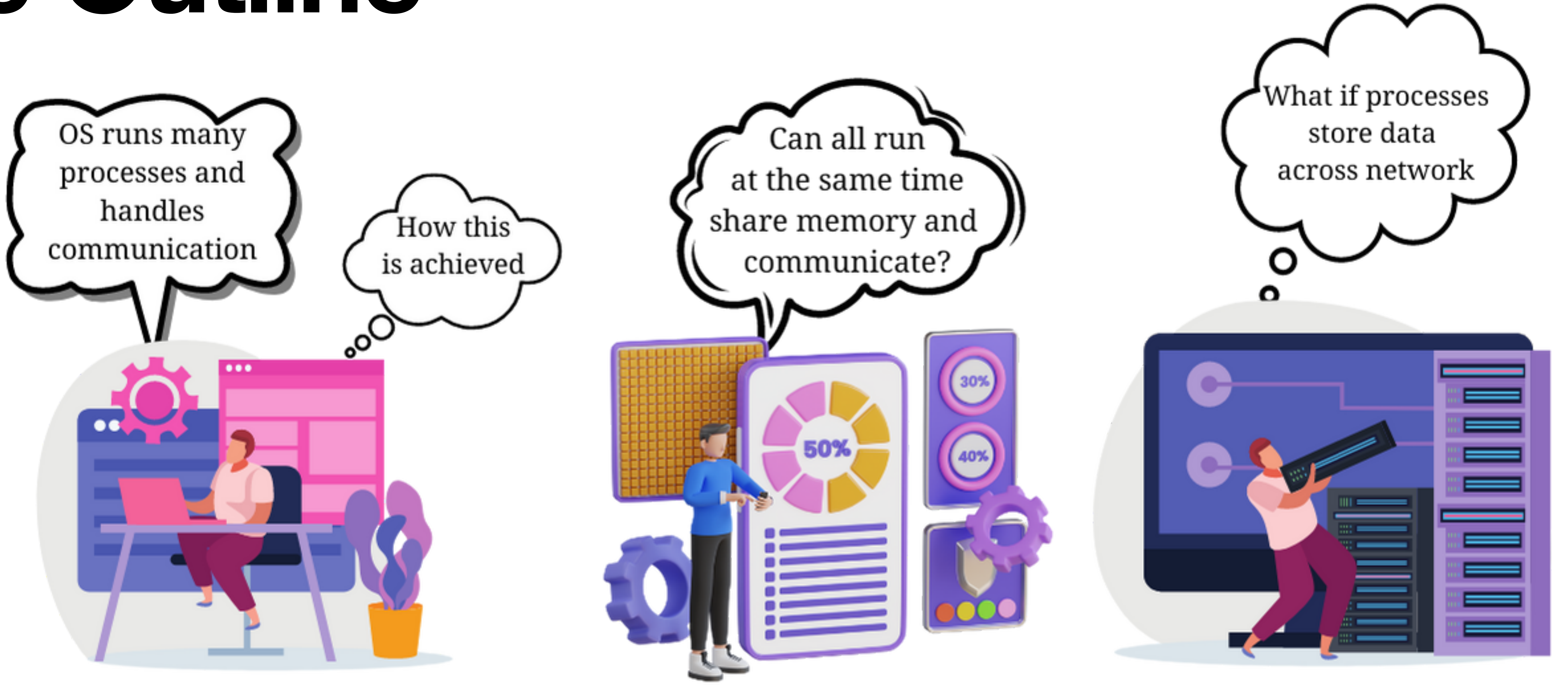
Acknowledgement

The materials used in this presentation have been gathered/adapted/generate from various sources as well as based on my own experiences and knowledge -- Karthik Vaidhyanathan

Sources:

- Operating Systems in three easy pieces by Remzi et al.

Course Outline



Getting the base right

Building concurrent processes that can communicate

Building a network file system

Timeline

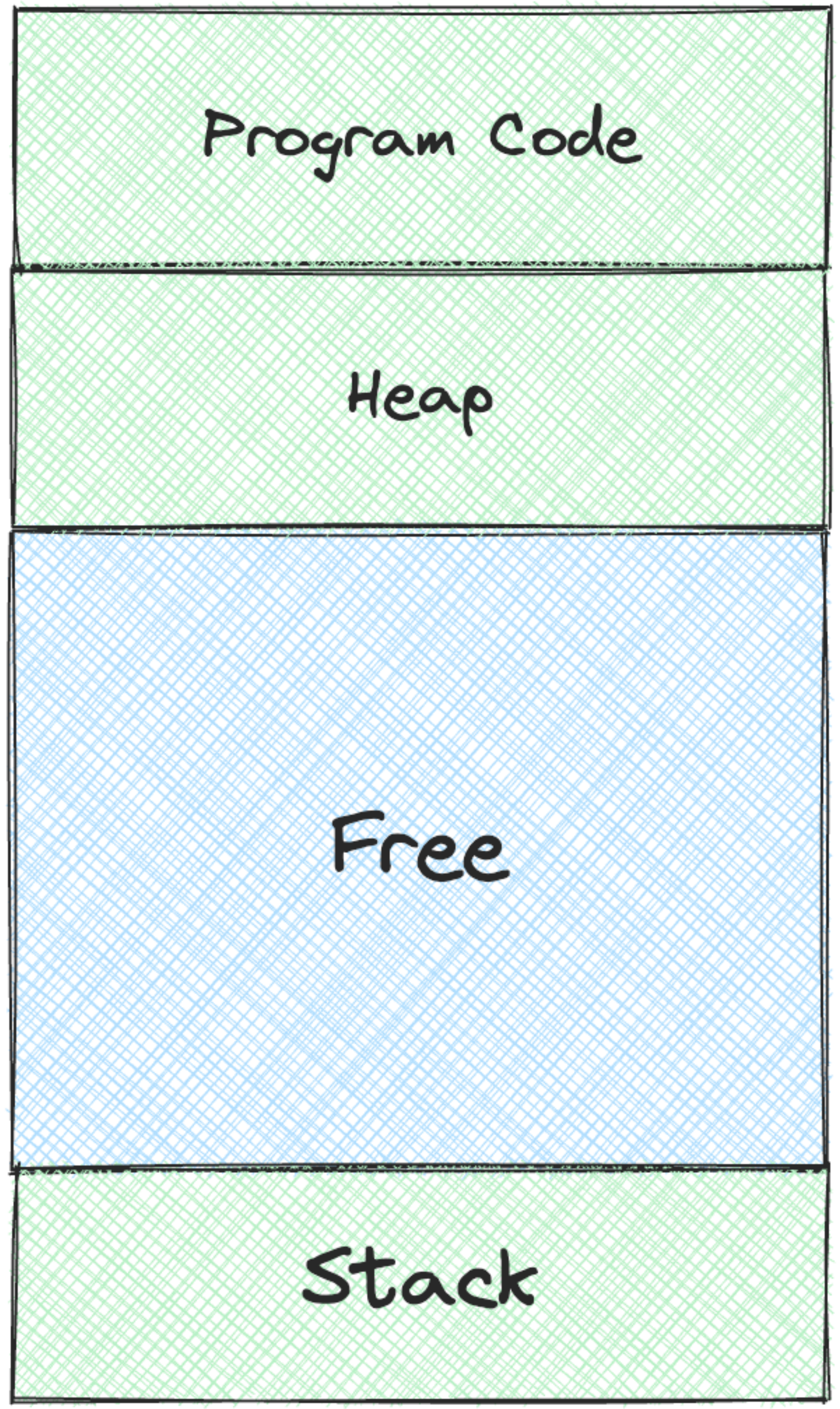
This Course

Process and Memory Virtualization	Concurrency	Persistence
Networking intro	Addressing and Routing	Network file Systems

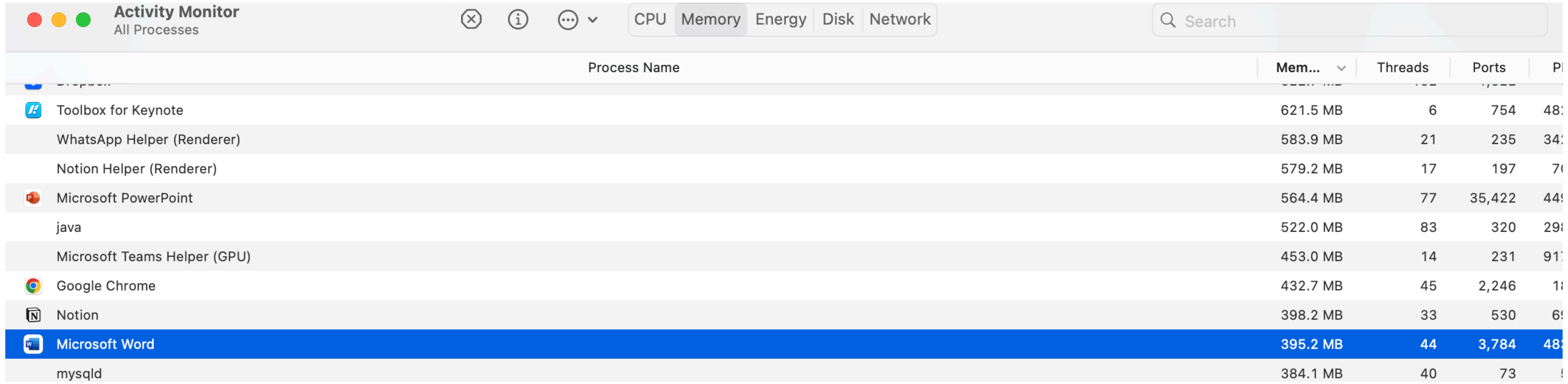
The Type of Process we have seen so far!

Some Recap

- Process during execution
  - Program Counter (PC): Points to the current instruction that is being run
  - Stack Pointer (SP): Points to the current frame of the function call
- What about the memory? - Paging!
- This is a single thread execution
- But in reality process is more than a single thread of execution



In reality a process does more things!

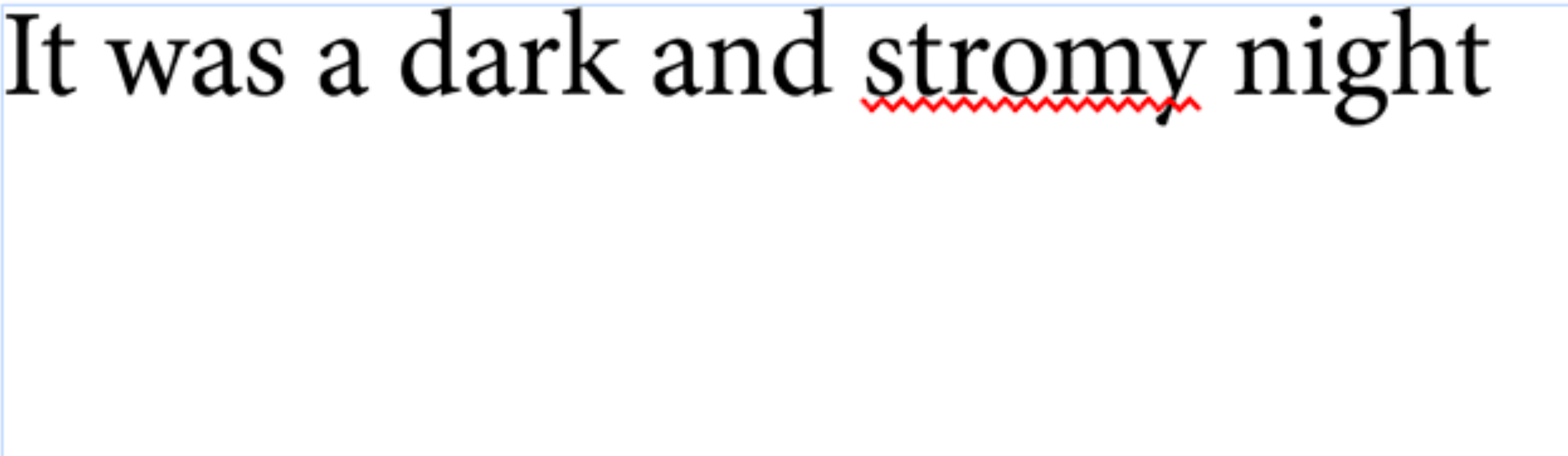


Process Name	Mem...	Threads	Ports	P
Toolbox for Keynote	621.5 MB	6	754	48
WhatsApp Helper (Renderer)	583.9 MB	21	235	34
Notion Helper (Renderer)	579.2 MB	17	197	7
Microsoft PowerPoint	564.4 MB	77	35,422	44
java	522.0 MB	83	320	29
Microsoft Teams Helper (GPU)	453.0 MB	14	231	91
Google Chrome	432.7 MB	45	2,246	1
Notion	398.2 MB	33	530	6
Microsoft Word	395.2 MB	44	3,784	48
mysqld	384.1 MB	40	73	

Check the processes running in your OS

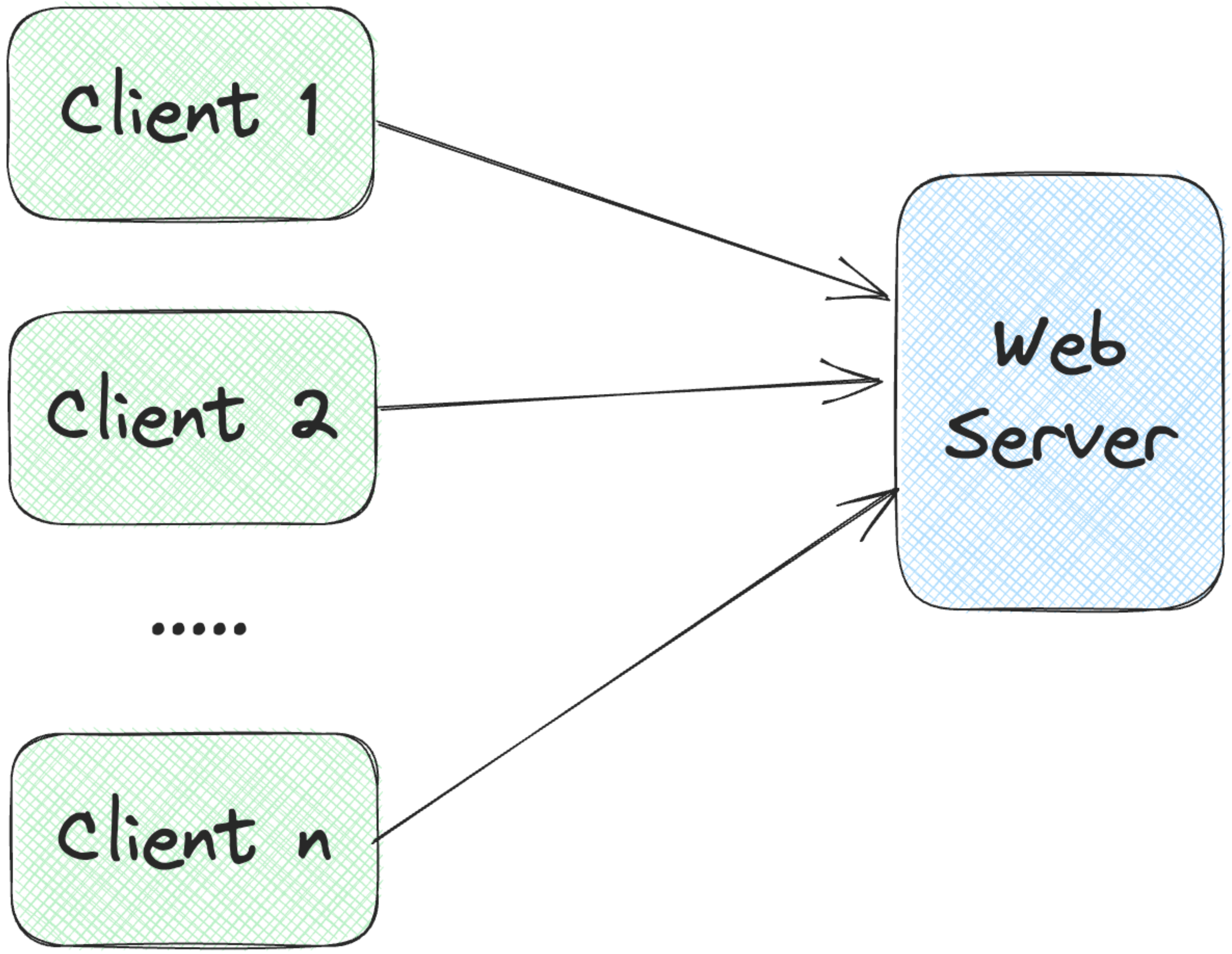
Microsoft Word is a process, while using it:

1. Spell checker works
2. Auto save happens
3. Auto formatting happens
4. ....



Think about a web server

- Web server runs a process to serve the clients
- Multiple clients may sent request to web server at the same time
- If the process handles each client sequentially - What can be an issue?
- How to make it more faster and better performing?
- What mechanism do we need? - Does multiple processes work?



An Analogy: Classrooms and Courses

Two Classrooms, two faculties teaching two different courses

Classroom 1: CS3.315 OS

Classroom 2: CS3.390 Networks

This is very similar to two separate processes

An Analogy: What if two faculties teach one course?

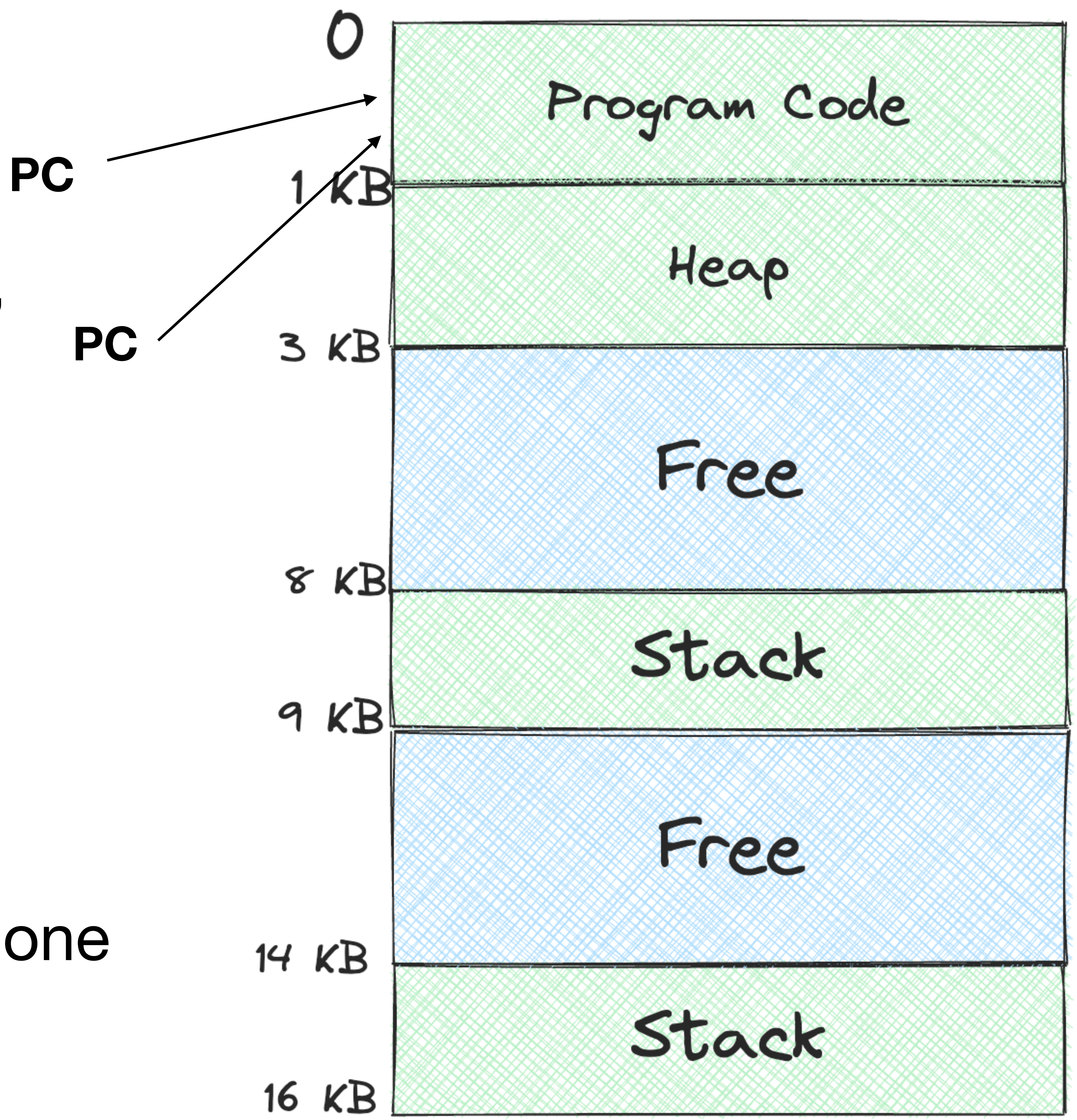
Two faculties teaching one course

Classroom 1: CS3.398 OS and Networks

- Can they teach at the same time?
  - Imagine such a scenario :-D
- Each teacher may take turns
- They may be at the class at the same time as well!
- There is only one attendance sheet, one course ID, one mark sheet
  - **Each faculty teaches in their style**
  - When question paper is set, they may take turns
  - The respective course content may be different
  - Somethings are shared!!

Process can have Threads!

- **Thread:** Another copy of the process that executes independently (lightweight process)
- Threads share the same address space (code, heap)
- Each thread:
  - Has separate Program counter
  - Separate stack for managing independent function calls
- In single thread, it was just about one PC and one stack



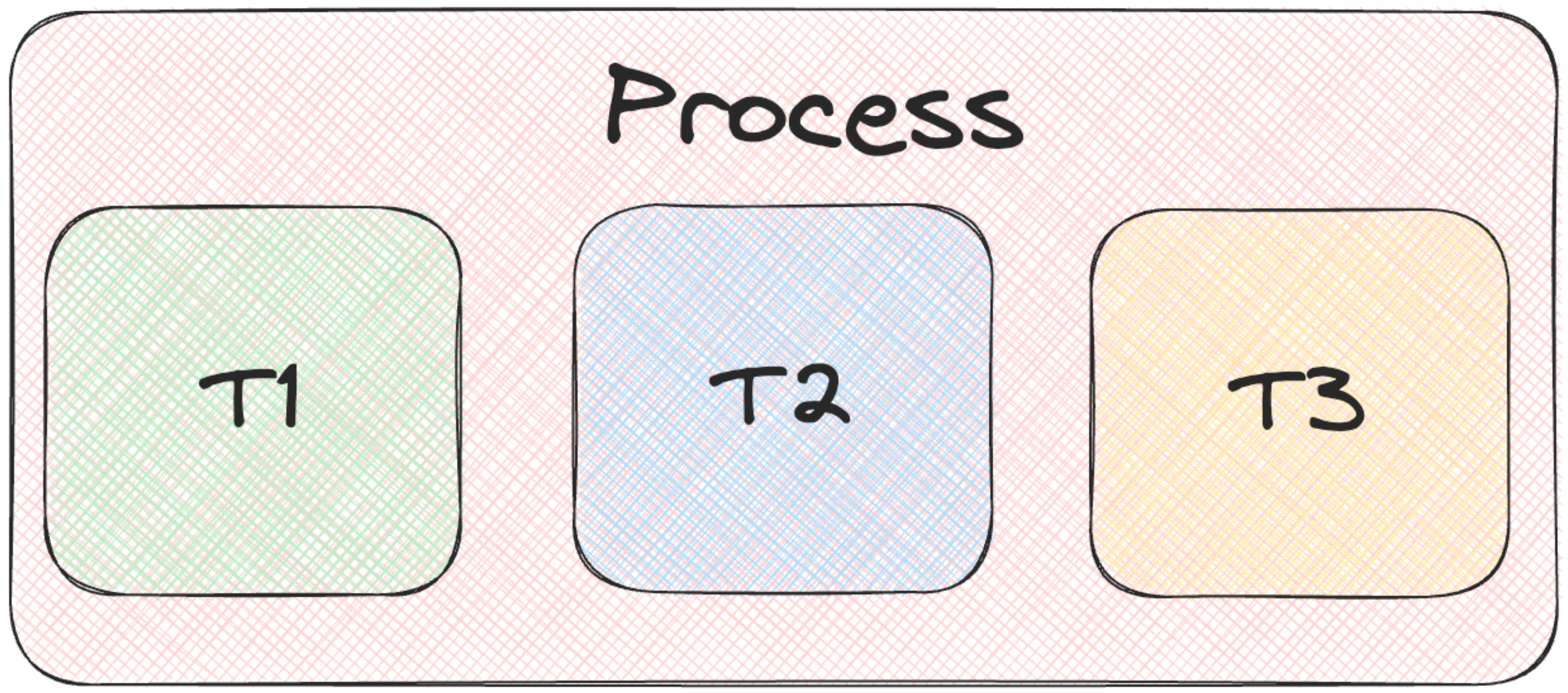
Wait, what about Process vs Threads?

Lets revisit parent and child - forks!

- What happens in a fork?
    - Parent and Child **do not share any memory**
        - Page tables are not shared, shared until changes - **Copy on Write (CoW)**
        - Subtle variations exist to improve efficiency but essentially parent and child are two different process
        - What about exec? - **Think!**
    - If they have to communicate, complicated inter process communication needs to be done (sockets, pipes, etc)
    - Extra copies of data, code, etc needs to be done

Threads

- Threads are another copy of process that executes independently
- Any process (parent process) can have multiple threads
  - Eg: Two threads T1 and T2
  - Both share the same address space - No separate page table, same code and same variables
  - Communication happens through shared variables (global)
  - Smaller memory footprint
- Threads are like separate process but share same address space

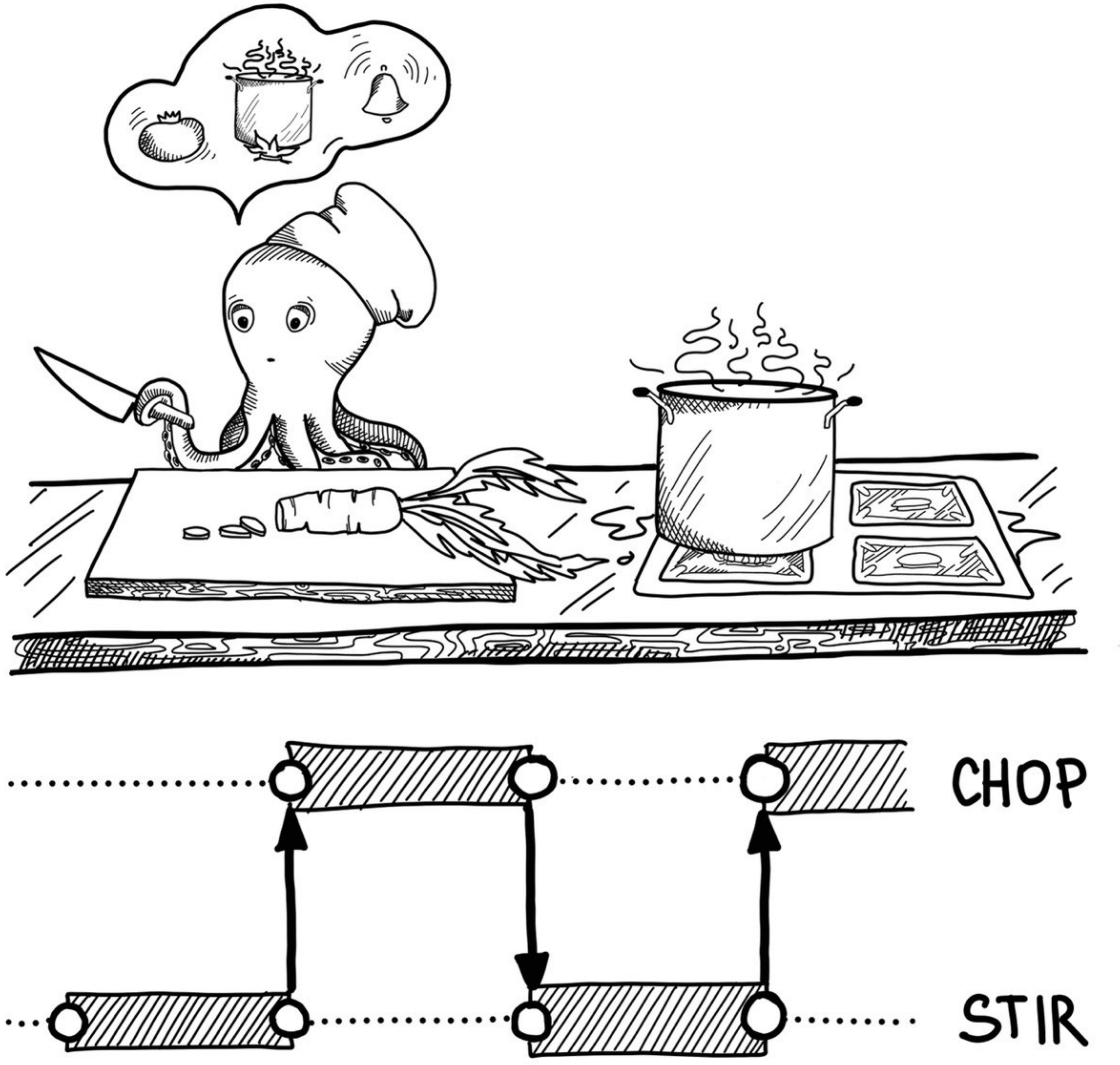


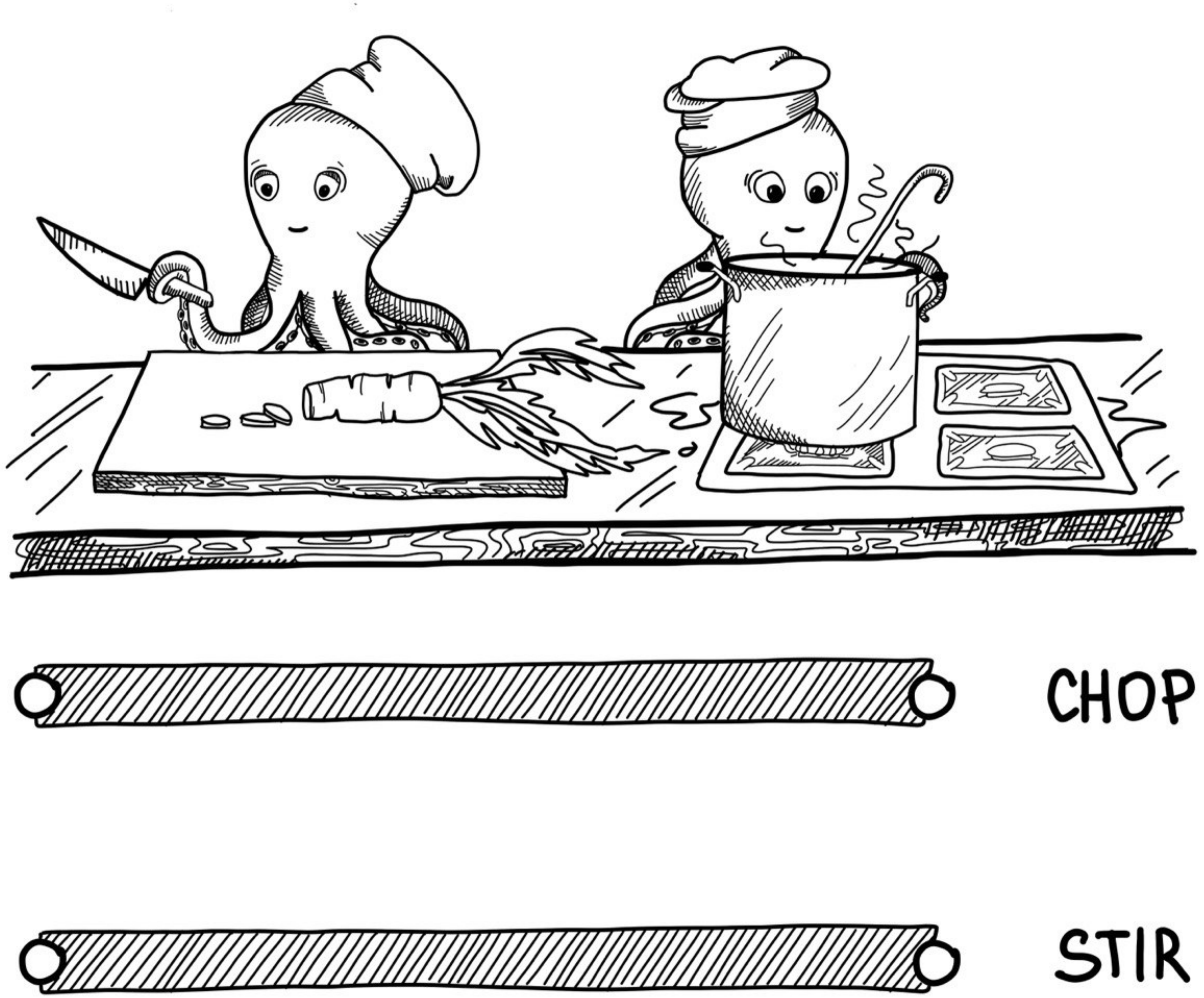
Why to do all these? Why Threads?

- Machine can be single core or multi-core:
  - Single process can effectively use multiple or even single CPU cores
  - Each thread can run independently and call different routines
  - Multi-threaded program has more than one point of execution
  - Within a process: one thread can perform I/O, one can perform computation, etc.
  - Scheduling happens between the threads - **Parallelism?**

Concurrency and Parallelism

What is what?





Source: https://freecontent.manning.com/concurrency-vs-parallelism/

Concurrency Vs Parallelism

Concurrency is about dealing with lot of things at once while parallelism is doing lot of things at once

- **Concurrency:** Running multiple threads/processes at the same time, even on a single CPU by interleaving their executions
- **Parallelism:** Running multiple threads/processes in parallel over different CPU cores
- Concurrent computations can be parallelized without changing correctness of result
- Concurrency by itself does not imply parallelism and vice versa
- Parallelism can be thought of as subclass of concurrency

Scheduling Threads

- OS schedules threads that are **ready**, similar to scheduling processes
- The context of thread (PC, registers) is saved into/restored from **Thread Control Block (TCB)**
    - Every PCB can have one or more linked TCBs corresponding to threads
- OS also has kernel level processes, each has threads - **Kernel threads**
    - **Kernel threads** can perform various tasks - system calls, handling interrupts, background tasks, etc. Execute in kernel mode. Eg: Linux pthreads
- **User threads** - managed by user level libraries. Execute in user mode
    - Eg: POSIX threads, anything that need not be managed by kernel

SCAN ME

Thank you

Course site: karthikv1392.github.io/cs3301_osn
Email: karthik.vaidhyanathan@iiit.ac.in
Twitter: @karthi_ishere