Processes and threads

Theory | Practice | 100% completed, 0 problems solved

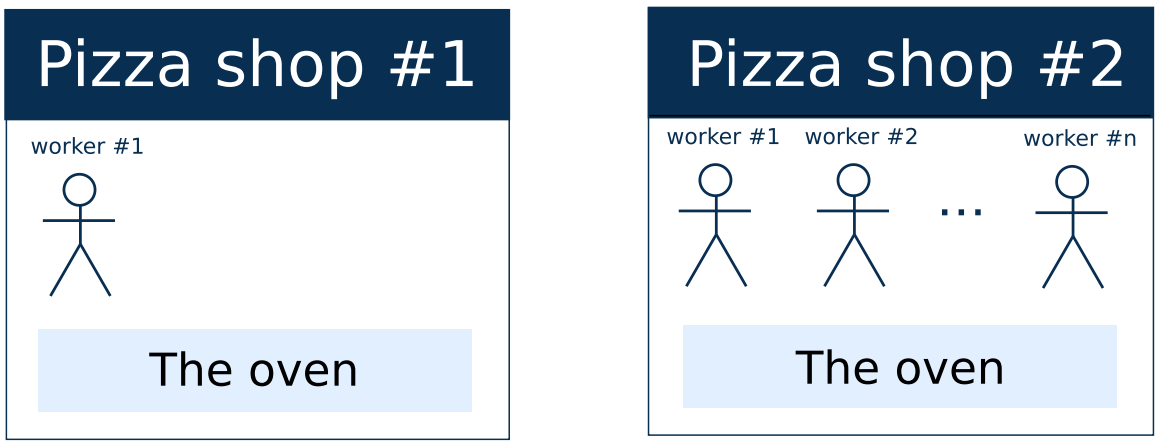
Theory

8 minutes reading

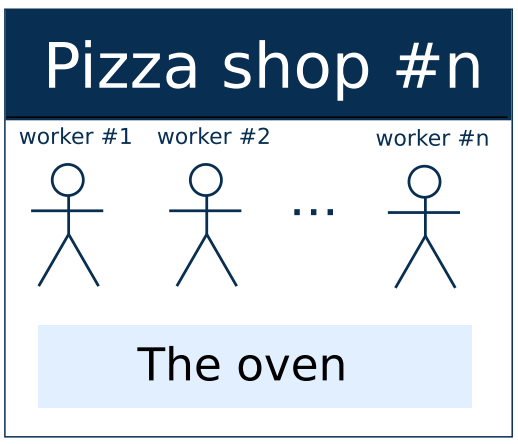
Unskip this topic | Start practicing

Imagine that you come to a food court at lunchtime and see a line of pizza shops there. Each shop's mission is to sell pizza and each of them has several workers. Their purpose is to sell pizza too, but they can't sell it by themselves without the equipment provided by the shop. On the other hand, any pizza shop can't sell anything without its staff. There has to be at least one worker to do the job.

It's similar to how a computer runs applications and manages multitasking and parallel execution. To understand it better, let's consider such concepts as *processes* and *threads*, as well as some similarities between these notions of computer science and the properties of a pizza shop.



, , ,



§1. Process

The **process** is the self-contained unit of execution that has all the things necessary to accomplish the mission. In short, the process is the container for its threads, all necessities for their work, and their shared resources. It’s cheaper to arrange access to shared resources once than to do it every time you spawn a new thread. The process has to have at least one thread as they do all the work. There is no such thing as a thread without its process or a process without at least one thread.

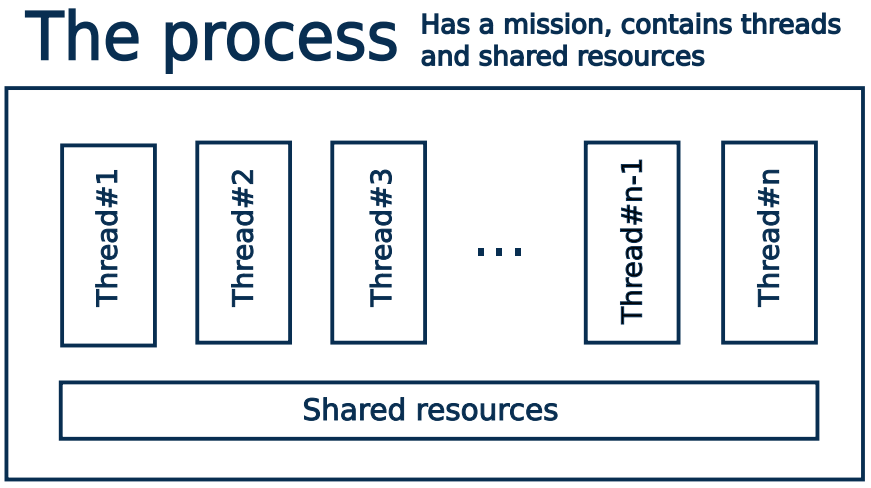
1 required topic

- Synchronous, asynchronous, parallel

10 dependent topics

- Threads as objects
- Goroutines
- Intro to multithreading
- More on multithreading
- Simultaneous tasks
- Introduction to NIO
- Environment variables
- Async methods
- Processes and threads inside computers
- Reactive programming concepts

Table of contents:



If we look at the pizza business, a single pizza shop would serve as an analogy for the process. It has all the environment and equipment required for a worker to do the job. Equipment is expensive, so it's cheaper and more efficient when workers share it. There is no need for each worker to acquire personal equipment. On the other hand, the shop can't do anything without the workers. It is essential to have at least one worker because without them all the equipment would be useless. Altogether, these things form a process of making and selling pizza.

§2. Thread

In computer science, the **thread** of execution is a stream of instructions inside a process that can be scheduled and run independently. Each thread has its executor, and this executor can perform only one thread at a time. Several threads inside the same process can run concurrently or in parallel.

To understand what the term *thread* means, think about employees in a pizzeria. They work according to their job descriptions. They complete various tasks according to the rules stated by the shop using shared resources granted by the shop.

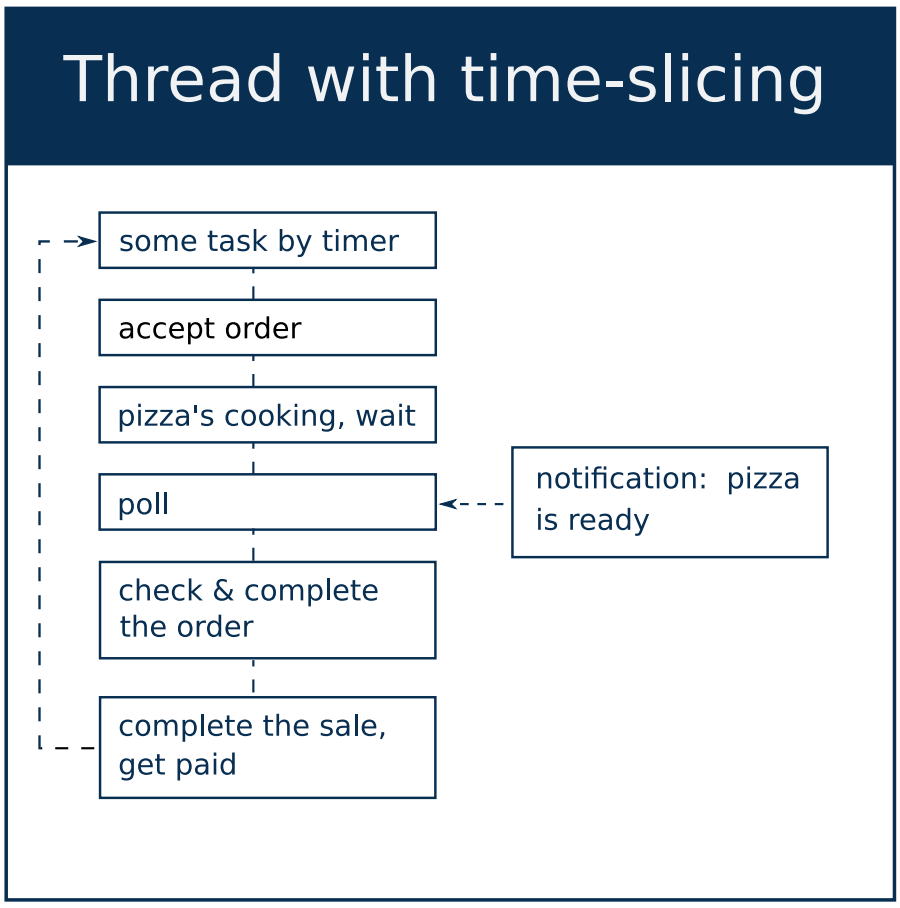
Workers in a pizzeria play the role of thread executors. Tasks that workers accomplish are the threads in the pizza shop "process".

§3. Internal or lightweight concurrency

Workers can play different roles during the process of selling pizza. Each of them can concurrently be a vendor, a cashier, or a cook at different points of the process. This concurrency is not among the workers but among the roles each worker plays. An important thing about these roles is that their tasks are typically fast enough and don't require a considerable amount of time and shared resources. They are **lightweight**.

If tasks are lightweight and don't require access to any shared resources except the executor's time and attention, there's no need to run them in different threads. It's cheaper to arrange their concurrent execution with time-slicing *inside one thread*. The concurrency of this sort is called internal for obvious reasons. Often it is also called lightweight because the tasks inside such a thread are typically small and quick.

The following image represents an example of a possible worker's thread with lightweight concurrency using time-slicing:



§4. Conclusion

- Processes are some sort of containers for workers' threads, shared resources, and parameters united by a common goal. Each of them always has at least one thread.
- Threads are independent execution units inside a process; they can run concurrently or in parallel with each other.
- Concurrent tasks that compete only for executor's time and don't require a lot of resources to be completed can run concurrently inside the same thread. These tasks are called lightweight, and this type of concurrency is called an internal or lightweight one. The execution inside threads can be synchronous or asynchronous but never parallel.

Report a typo

392 users liked this piece of theory. **29** didn't like it. **What about you?**

Start practicing Unskip this topic

Comments (28) Useful links (3) Show discussion