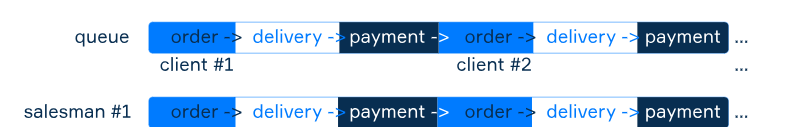
When we are considering some complex process, let's call it **workflow**, various parts of it may run differently. Sometimes actions go one by one, sometimes they go in random order overlapping each other, and sometimes things go simultaneously and in parallel. The workflow can evolve differently. There are three sorts of workflow executions sequence: synchronous, asynchronous, and parallel.

Many terms related to computer program processing are not just technical ones. They describe a wide variety of real-world phenomena. In some sense, the processes taking place inside a computer are not that different from those in real life. Moreover, on some level of abstraction, they are practically identical. So, let's try to use them and explore their base concepts using real-life examples.

An appropriate example of a complex process is customer service. Let's use it to study some basic types of workflow from the point of view of the sequence of execution.

**Synchronous workflow**

There are many models to manage customer flows. The simplest approach is one shop with one seller. The seller deals with each client from the beginning to the end of each sale *and* performs all the roles from storekeeper to cashier.

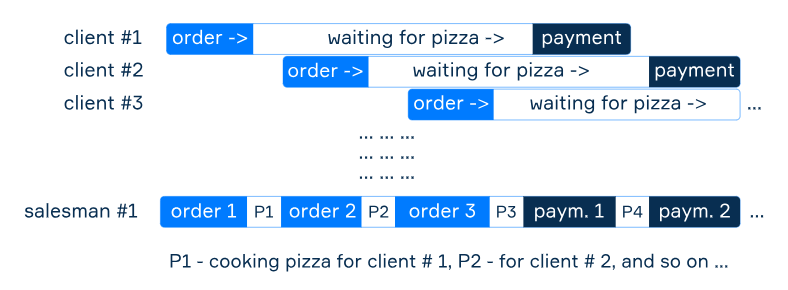


When there are many customers at the same time, this approach is very far from perfect as the seller can deal only with one client per time, while others have to wait in line. They serve each client separately one by one which means starting to serve the next client only after finishing with the current. We name this type of action a **synchronous** one.

Synchronous workflows are very common. Most of the activities should go synchronously if their goal is to achieve some specific results. The number of examples is enormous. Scenes in a movie plot, car assembling, words in a sentence, cooking, you name it.

**Asynchronous workflow**

Let's imagine our old shop becomes fancier, this is a pizza shop now. After the first client has ordered their pizza, they need to wait for it to be cooked. At this point, the seller leaves the first customer alone for a while, and now the second one can make their order, then the third, and so on.



When the first client's pizza is ready, the seller returns to them to complete the sale. That's how this story will repeat again and again.

Our old friend seller can serve several customers simultaneously in overlapping periods. We call such behavior **asynchronous**.

Operations of this kind often emerge when there is a need for waiting. Imagine you are reading on an aircraft while flying, or you do the dishes while something is cooking; those pairs of activities are asynchronous.

**Parallel processing**

As the pizza shop sales are growing, now one worker is not enough for the whole business. So, we should hire several. If each seller has a separate compact oven for preparing exactly one pizza at a time, then we can divide the queue of buyers among the sellers.



Now each of them works independently, and this is a case of **parallel** processing. Each task in parallel processing is running in a continuous period as a whole unit process. Parallel execution is possible only if there is more than one executor. Cashiers in a supermarket are an example of parallel processing in everyday life, as well as highways.

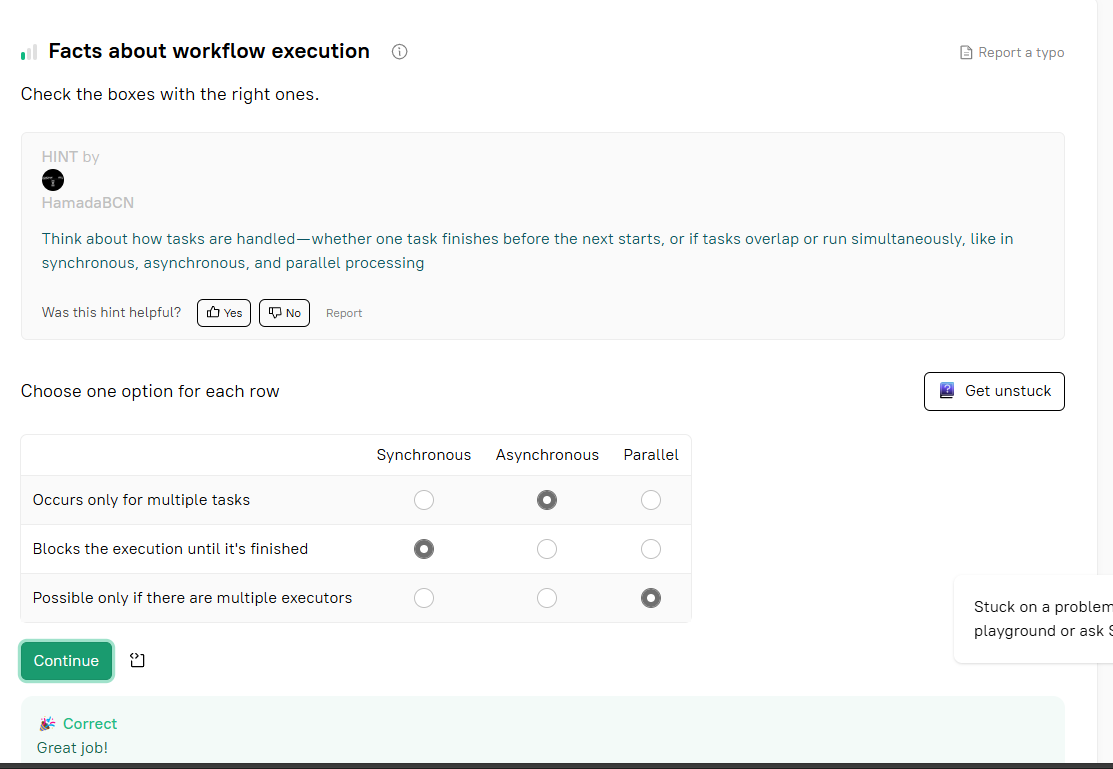
**Conclusion**

So, there are three types of workflow processing. The first is synchronous, the second is asynchronous, and the third is parallel.

* Synchronous: one task at a time, the next starts when the current is done.
* Asynchronous: multiple tasks at the same time in overlapping periods, executed by little parts.
* Parallel: multiple or one task split into parts, being executed continuously by different executors in parallel.

What does synchronous execution mean?

Tasks execution goes one by one; the next task starts only if the previous one finished.

****

The **correct statements** are:

* ✅ **Asynchronous tasks execution process can come in parallel**
* ✅ **A synchronous task blocks the whole process till the task is completed**

**Incorrect statements:**

* ❌ **Asynchronous tasks can't run one by one; they must overlap each other in time**  
  → Asynchronous tasks *can* run one by one if needed; they don't *have to* overlap.
* ❌ **Synchronous tasks are outdated and are rarely used nowadays**  
  → Synchronous tasks are still widely used where appropriate (e.g., for ordered execution or simpler logic).

What does the asynchronous execution mean?

Tasks can start at any time; they don't have to wait until the previous one finished.

Consider a scenario where you are a developer tasked with optimizing an app's performance. Which of the following strategies utilize multithreading, multiprocessing, and asynchronous execution to enhance the app performance? Choose all that apply.

The **correct strategies** that utilize **multithreading**, **multiprocessing**, and **asynchronous execution** to enhance app performance are:

* ✅ **Dividing data parsing task into multiple subtasks handled by different threads**  
  → This is **multithreading**.
* ✅ **Implementing asynchronous API calls allowing users to continue using the app while the data is being fetched from the server**  
  → This is **asynchronous execution**.
* ✅ **Allowing multiple instances of the app to run on different cores of the processor**  
  → This is **multiprocessing**.

## Synchronous vs Asynchronous Processing

 Report a typo

Consider a server application that handles multiple client requests. In the context of multithreading and multiprocessing in operating systems, what would be the differences between handling these requests synchronously and asynchronously?

**Hint**

Asynchronous processing: multiple requests can be processed simultaneously.Asynchronous processing uses buffer and signals to process the requests.Synchronous processing doesn't require signals, the request are processed immediately.

The **correct statements** are:

* ✅ **In synchronous processing, the server handles one request at a time, blocking all others until it is fully processed, while in asynchronous processing, multiple requests can be processed at the same time without blocking.**
* ✅ **In asynchronous processing, the server places incoming requests into a buffer, and signals are used to start processing them. In synchronous processing, requests are processed immediately as they come in and no signals are required.**

**Incorrect statements:**

* ❌ **In synchronous processing, the server can handle multiple requests in parallel...**  
  → This is false. **Synchronous** means **one at a time**, blocking others.
* ❌ **In synchronous processing, signals are used...**  
  → This is reversed. **Signals** are typically used in **asynchronous** processing.

## Data fetching application

 Report a typo

In a web application, multiple requests come in for data fetching. The application has one executor handling tasks. Which of the following are correct statements regarding its execution? Select all that apply.

**Hint**

Consider how tasks are managed by a single executor and whether it allows handling multiple tasks at the same time. Think about whether asynchronous processing requires changes to the executor setup and whether parallel execution is possible with just one executor

The **correct statements** are:

* ✅ **It can handle requests asynchronously.**  
  → A single executor can manage asynchronous tasks using non-blocking I/O or event loops (e.g., in async frameworks).
* ✅ **It would require multiple executors for processing the requests in parallel.**  
  → True. **Parallel processing** requires multiple threads or processes (executors).
* ✅ **It can process requests sequentially, one by one.**  
  → A single executor by default processes tasks one at a time unless asynchronous logic is used.

**Incorrect statement:**

* ❌ **It can process the requests in parallel without any changes required.**  
  → False. A **single executor** cannot perform true parallel execution without modification.