Asynchronous vs synchronous execution, what does it really mean?

Asked 10 years, 3 months ago Active 2 months ago Viewed 792k times

1070

What is the difference between asynchronous and synchronous execution?

asynchronous execution synchronous

635

edited Sep 23 '18 at 17:09
nisarg parekh
363 1 18

asked Apr 14 '09 at 15:39
tush1r
7,704 14 29 33

... or "sequential vs parallel" execution ... although true parallelism requires multiple cores, otherwise it's logically interleaved. – samis May 30 '18 at 16:36

3 I used to confuse both terms, and the way I did to remember the difference is throw the first "A" in "**A**JAX" which stands for asynchronous, in JavaScript when you do requests with AJAX in a loop they don't wait for each others or block the process, because the browser doesn't want to impact the user experience with a frozen website, all requests are sent almost at the same time without waiting the response of the previous request. That is **asynchronous** – Accountant م Jan 15 at 10:19

22 Answers

1595

When you execute something synchronously, you wait for it to finish before moving on to another task. When you execute something asynchronously, you can move on to another task before it finishes.

That being said, in the context of computers this translates into executing a process or task on another "thread." A thread is a series of commands (a block of code) that exists as a unit of work. The operating system can manage multiple threads and assign a thread a piece ("slice") of processor time before switching to another thread to give it a turn to do some work. At its core (pardon the pun), a processor can simply execute a command, it has no concept of doing two things at one time. The operating system simulates this by allocating slices of time to different threads.

All of this is about allowing the operating system to manage the completion of your task while you can go on in your code and do other things.

Asynchronous programming is a complicated topic because of the semantics of how things tie together when you can do them at the same time. There are numerous articles and books on the subject; have a look!

edited Sep 14 '17 at 15:19
MeanwhileInHell
2,483 12 34 68

answered Apr 14 '09 at 15:43

Adam Robinson
153k 27 256 324

203 What absolutely confuses me is that synchronous means "at the same time", yet when used in the sense above, it means *sequential*, and asynchronous means "not at the same time"...?? Can somebody explain this conflict? – Damien Roche Aug 7 '13 at 17:53

37 @Zenph: In this context, an entire block of code is what we're concerned with. Synchronous means that the block is executed at the same time (though, yes, the components are executed sequentially). Asynchronous means that the block is not all executed at the same time. – Adam Robinson Aug 7 '13 at 18:15

6 Asynchronous execution also happens when a program sends a message to a queue (as in messaging systems, such as ActiveMQ, WebSphere MQ, HornetQ, MSMQ, etc.). In this case, the asynchronous call doesn't involve multithread programming or handling concurrency at the OS level. – Paulo Merson Jun 15 '15 at 12:07

235 Oddly enough "Synchronously" means "using the same clock" so when two instructions are synchronous they use the same clock and must happen one after the other. "Asynchronous" means "not using the same clock" so the instructions are not concerned with being in step with each other. That's why it looks backwards, the term is not referring to the instructions relationship to each other. It's referring to each instructions relationship to the clock. Hope that helps. – Tom Padilla Jan 28 '16 at 15:31

13 The terms come from engineering. en.wikipedia.org/wiki/Asynchronous_system – Tom Padilla Jan 28 '16 at 15:50

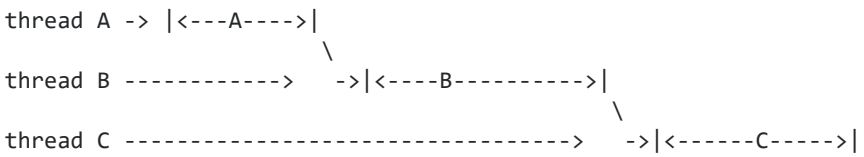
1054

Synchronous/Asynchronous HAS NOTHING TO DO WITH MULTI-THREADING.

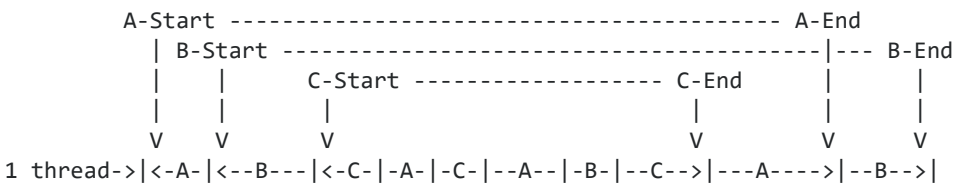
Synchronous, or *Synchronized* means "connected", or "dependent" in some way. In other words, two synchronous tasks must be aware of one another, and one task must execute in some way that is dependent on the other, such as wait to start until the other task has completed.
Asynchronous means they are totally independent and neither one must consider the other in any way, either in initiation or in execution.

Synchronous (one thread):

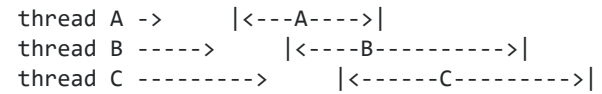
Synchronous (multi-threaded):



Asynchronous (one thread):



Asynchronous (multi-Threaded):



- Start and end points of tasks A, B, C represented by `<` , `>` characters.
- CPU time slices represented by vertical bars `|`

Technically, the concept of synchronous/asynchronous really **does not have anything to do with threads**. Although, in general, it is unusual to find asynchronous tasks running on the same thread, it is possible, (see below for examples) and it is *common* to find two or more tasks executing synchronously on *separate* threads... No, the concept of synchronous/asynchronous has to do *solely* with whether or not a second or subsequent task can be initiated before the other (first) task has completed, or whether it must wait. That is all. What thread (or threads), or processes, or CPUs, or indeed, what hardware, the task[s] are executed on is not relevant. Indeed, to make this point I have edited the graphics to show this.

ASYNCHRONOUS EXAMPLE. In solving many engineering problems, the software is designed to split up the overall problem into multiple individual tasks, and then execute them asynchronously. Inverting a matrix, or a finite element analysis problem, are good examples. In computing, sorting a list is an example. The quick sort routine, for example, splits the list into two lists, and sorts each of them by calling itself recursively. In both of the above examples, the two tasks can (and often were) executed asynchronously. They do

inputs to the other task. As long as the start and end times of the tasks overlap, (possible only if the output of neither is needed as inputs to the other), they are being executed asynchronously, no matter how many threads are in use.

SYNCHRONOUS EXAMPLE. Any process consisting of multiple tasks where the tasks must be executed in sequence, but one must be executed on another machine (Fetch and/or update data, get a stock quote from a financial service, etc.). If it's on a separate machine it is on a separate thread, whether synchronous or asynchronous.

edited Jul 25 '17 at 4:06

NH.
1,031 1 15 31

answered Apr 14 '09 at 15:53

Charles Bretana
117k 20 128 201

84 why in the world do words mean different things in computer...always leave me coming back to this...from dictionary.. synchronous: **occurring at the same time.** asynchronous: **not occurring at the same time.** – Muhammad Umer Sep 8 '13 at 14:56

15 but as you can see in computers it means the opposite – Muhammad Umer Sep 8 '13 at 14:57

5 Maybe the nomenclature is based on whether initiation of tasks is "synchronized" with completion of other tasks? – Charles Bretana Sep 18 '13 at 16:10

11 @MuhammadUmer: in computer world, occurring at the same time is called concurrency. – Roy Ling Oct 18 '13 at 1:12

5 IMHO, these pictures do not all describe the synchronous vs asycnhronous execution of tasks For example, the second picture implies that asynchronous tasks require several threads. Actually, it does not. And that tasks have to run in parallel, which is not a requirement as well. Or, the picture for "synchronous" very well shows how tasks have been asynchronously dispatched form some call-site and now execute in a serial task scheduler ;) IMO, the pictures are misleading. – CouchDeveloper Sep 15 '15 at 7:35

586

In simpler terms:

SYNCHRONOUS

You are in a queue to get a movie ticket. You cannot get one until everybody in front of you gets one, and the same applies to the people queued behind you.

ASYNCHRONOUS

You are in a restaurant with many other people. You order your food. Other people can also order their food, they don't have to wait for your food to be cooked and served to you before they can order. In the kitchen restaurant workers are continuously cooking, serving, and taking orders. People will get their food served as soon as it is cooked


Ryan
549 6 17


themightysapien
6,571 1 12 14

9 If someone wants apples compared with apples; if you wanted the restaurant scenario to be synchronous, then when you order food, everyone else in the restaurant would have to wait for your food to arrive before they can order their food etc. Now this seems like a really dumb scenario to be in, but in the computing world this scenario could be useful. Say each customer cant decide what they want, and instead want to look at what the previous customer orders to decide if they want that or not, then it makes sense that they have to wait for the food to arrive before ordering. – Fonix Feb 1 '17 at 3:42

Just to add on... it could be so that the operations execute like in a queue in Asynchronous operations... But that is not mandatory at all. – Sreekanth Karumanaghat Sep 13 '17 at 13:05

2 very simple and real life example – Manish Sep 20 '18 at 7:03

291

Simple Explanation via analogy

Synchronous Execution

My boss is a busy man. He tells me to write the code. I tell him: Fine. I get started and he's watching me like a vulture, standing behind me, off my shoulder. I'm like "Dude, WTF: why don't you go and do something while I finish this?"

he's like: "No, I'm *waiting right here* until you finish." This is synchronous.

Asynchronous Execution

The boss tells me to do it, and rather than waiting right there for my work, the boss goes off and does other tasks. When I finish my job I simply report to my boss and say: "I'm DONE!" This is Asynchronous Execution.

(Take my advice: NEVER work with the boss behind you.)

edited Jan 27 at 2:45

answered Jun 8 '15 at 11:13


BKSpurgeon
14.8k 4 59 51

20 I'm DONE... its even funnier when you take this as a resignation. – Daedric Dec 14 '16 at 8:16

87

Synchronous execution means the execution happens in a single series. `A->B->C->D` . If you are calling those routines, `A` will run, then finish, then `B` will start, then finish, then `C` will start, etc.

With **Asynchronous execution**, you begin a routine, and let it run in the background while you start your next, then at some point, say "wait for this to finish". It's more like:

Start `A->B->C->D->` *Wait* for `A` to finish

The advantage is that you can execute `B` , `C` , and or `D` while `A` is still running (in the background, on a separate thread), so you can take better advantage of your resources and have fewer "hangs" or "waits".

edited Nov 12 '15 at 22:26

Jimi
636 5 10

answered Apr 14 '09 at 15:43

Reed Copsey
480k 60 1003 1287

@ Reed Copsey ..... Thanks for such a good explanation ..... Just wanted some more info on Async-Exec ..... Based on your answer in Async Exec .... Start A->B->C->D->Wait for A to finish ... So all A,B, C, D starts at a time ..... and they wait for A to finish ..... So does B will only finish after A finishes , and C after B and so on ..... ? Or can B first finish and then A can finish ? – Devrath Jul 20 '13 at 16:34

8 @Devrath The operations can finish in any order. – Reed Copsey Jul 22 '13 at 15:45

53

Synchronous means that the caller waits for the response or completion, asynchronous that the caller continues and a response comes later (if applicable).

As an example:

```
static void Main(string[] args)
{
    Console.WriteLine("Before call");
    doSomething();
    Console.WriteLine("After call");
}

private static void doSomething()
{
    Console.WriteLine("In call");
}
```

```
Before call
In call
After call
```

But if we were to make doSomething asynchronous (multiple ways to do it), then the output *could* become:

```
Before call
After call
In call
```

Because the method making the asynchronous call would immediately continue with the next line of code. I say "could", because order of execution can't be guaranteed with asynch operations. It could also execute as the original, depending on thread timings, etc.

edited May 26 '15 at 16:40

Nick Orlando
442 1 5 16

answered Apr 14 '09 at 15:47

Ragoczy
2,454 15 17

This is the most practical explanation I see here. – Steve Rowe Apr 14 '09 at 16:39

52

In a nutshell, synchronization refers to two or more processes' **start** and **end** points, ***NOT*** their **executions**. In this example, Process A's endpoint is synchronized with Process B's start point:

```
SYNCHRONOUS
   |--------A--------|
                     |--------B--------|
```

Asynchronous processes, on the other hand, do *not* have their start and endpoints synchronized:

```
ASYNCHRONOUS
   |--------A--------|
         |--------B--------|
```

Where Process A overlaps Process B, they're running concurrently or synchronously (dictionary definition), hence the confusion

edited Jul 16 '18 at 22:18

answered Aug 17 '15 at 14:11

entr0p3te

521 4 4

2 copy of Charles Bretana answer – Dinesh Saini Aug 18 '15 at 4:37

2 @DineshSaini - My diagram is slightly different . For clarity, I placed A on top of B in both cases, emphasizing whether their start & endpoints are synchronized. Charles Bretana's diagram places the synchronous processes in sequence without "syncing" anything. (I was going to comment below his answer to "improve" it, but realized it would be easier just to show the new diagram.) – entr0p3te Aug 19 '15 at 14:00

Ok then upvote to u – Dinesh Saini Aug 21 '15 at 5:55

Great diagrams. I think the way to call the top one SYNC, is that the start and end of A in the top diagram are effectively at the same time, in the sense that no other events have intervened, or could have interfered with A's completion. Sync can refer to a single task in isolation, like adding to CPU registers, whose start and end are so close, as to effectively be actually dictionary-synchronous. – Dean Radcliffe Dec 28 '16 at 15:32

33

I think this is bit round-about explanation but still it clarifies using real life example.

Small Example:

Let's say playing an audio involves three steps:

1. Getting the compressed song from harddisk
2. Decompress the audio.
3. Play the uncompressed audio.

If your audio player does step 1,2,3 sequentially for every song then it is synchronous. You will have to wait for some time to hear the song till the song actually gets fetched and decompressed.

If your audio player does step 1,2,3 independent of each other, then it is asynchronous. ie. While playing audio 1 (step 3), if it fetches audio 3 from harddisk in parallel (step 1) and it decompresses the audio 2 in parallel. (step 2) You will end up in hearing the song without waiting much for fetch and decompress.

edited Aug 11 '16 at 4:10

user207421

268k 28 224 380

answered Apr 14 '09 at 16:27

aJ.

25k 20 73 117

22

Simply said asynchronous execution is doing stuff in the background.

For example if you want to download a file from the internet you might use a synchronous function to do that but it will block your thread until the file finished downloading. This can make your application unresponsive to any user input.

Instead you could download the file in the background using asynchronous method. In this case the download function returns immediately and program execution continues normally. All the download operations are done in the background and your program will be notified when it's finished.

edited Apr 14 '09 at 15:54

answered Apr 14 '09 at 15:41

Michał Piaskowski

3,291 2 27 45

1 how is your example gonna be faster. In the end you can't play the file until it is done downloading period. Can you explain? I guess I don't understand async then and it's probably me but what would that other step or process be doing while the other process is running (getting the download)? I mean what can you do until you receive that async process back (download) in your app...I don't get it. So what, you'd still have to show the user some kind of wait mechanism no mater what in either situation? – PositiveGuy Dec 10 '12 at 5:48

4 It doesn't have to faster. It's about not blocking the main thread, so that it can process other kind of user input. For example user might want to cancel the download or start downloading another file simultaneously. – Michał Piaskowski Dec 10 '12 at 15:19

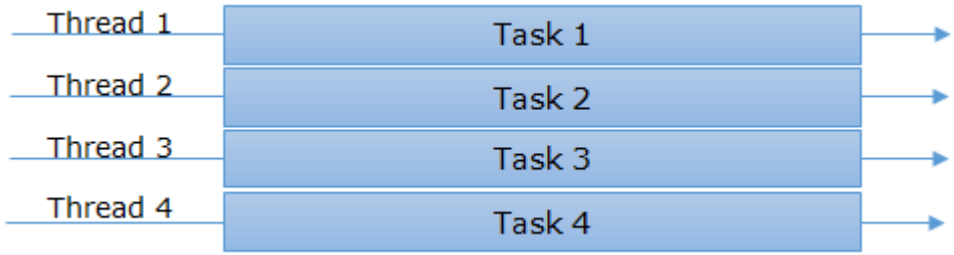
21

Synchronous Programming model – A thread is assigned to one task and starts working on it. Once the task completes then it is available for the next task. In this model, it cannot leave the executing task in mid to take up another task. Let's discuss how this model works in single and multi-threaded environments.

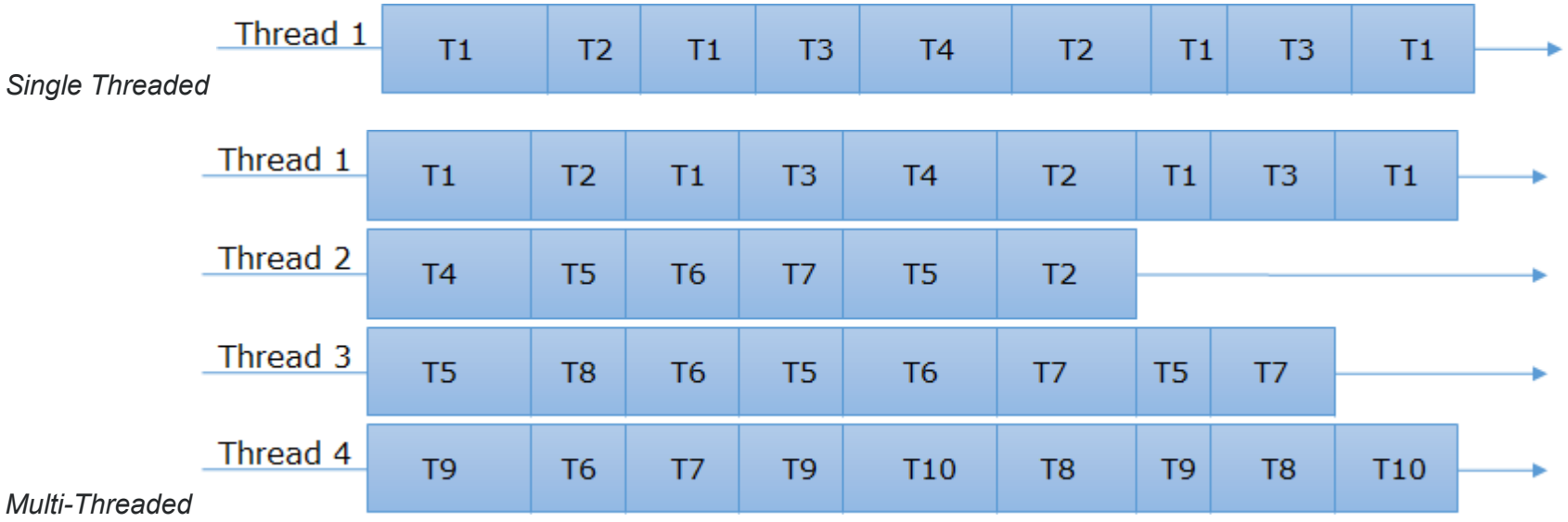
Single Threaded – If we have couple of tasks to be worked on and the current system provides just a single thread, then tasks are assigned to the thread one by one. It can be pictorially depicted as

Multi-Threaded – In this environment, we used to have multiple threads which can take up these tasks and start working on that. It means we have a pool of threads (new threads can also be created based on the requirement and available resources) and bunch of

tasks. So these thread can work on these as



Asynchronous Programming Model – In contrary to Synchronous programming model, here a thread once start executing a task it can hold it in mid, save the current state and start executing another task.



Read more here

edited May 17 at 7:29

answered Mar 20 '18 at 13:54

yoAlex5

6,142 2 39 30

The diagram in the synchronous multi-threaded example appears to depict concurrently executing threads? – samis May 30 '18 at 19:11

@sɑmosΛris you can consider Thread like X axis that indicates timeline – yoAlex5 May 30 '18 at 19:33

14

When executing a sequence like: a>b>c>d>, if we get a failure in the middle of execution like:

```
a
b
c
fail
```

Then we re-start from the beginning:

```
a
b
c
d
```

this is synchronous

If, however, we have the same sequence to execute: a>b>c>d>, and we have a failure in the middle:

```
a
b
c
fail
```

...but instead of restarting from the beginning, we re-start from the point of failure:

```
c
d
```

...this is know as asynchronous.

edited Dec 15 '12 at 14:03

Jerry Coffin
396k 58 492 936

answered Dec 15 '12 at 13:38

mohamed tharik
181 1 2

11

As a really simple example,

SYNCHRONOUS

Imagine 10 school students instructed to walk as a queue on a road.

The 3rd student got her shoelace untied. Now she has stopped and tying up again.

All students behind her have stopped, and are waiting now for her to get it tied. The 1st and 2nd students have walked past them all, and continuing in their usual pace.

```
10-->9-->8-->7-->6-->5-->4-->3.     2-->1-->
```

ASYNCHRONOUS

Just Imagine 10 random people walking on the same road. They're not on a queue of course, just randomly walking on different places on the road in different paces.

3rd person's shoelace got untied. She stopped to get it tied up again.

But nobody is waiting for her to get it tied. Everyone else is still walking the same way they did before, in that same pace of theirs.

```
10-->    9-->
   8--> 7-->   6-->
 5-->  4-->  3. 2-->
1-->
```

answered Oct 2 '17 at 11:22

Dasun Nirmitha
144 3 10

When the person who stopped for the shoelace, resumes after tieing the shoelace, he supposed to give the return value to the next person behind him to continue(continuation task), but he already crossed him, so the example looks no meaning to this! – SmartestVEGA Nov 2 '17 at 10:03

1 you can think of 3 as 4's mom and hence 4 stops, when 3 finishes tieing shoelace 4 is notified, as 4's step are dependent on 3 – jamsujan Dec 6 '17 at

7

You are confusing Synchronous with Parallel vs Series. Synchronous mean all at the same time. Syncronized means related to each othere which can mean in series or at a fixed interval. While the program is doing all, it it running in series. Get a dictionary...this is why we have unsweet tea. You have tea or sweetened tea.

answered Jan 11 '13 at 15:18

Joe
95 1 1

3 Actually, "synchronized" refers to the relationship between the instructions and the clock. NOT the relationship between the instructions themselves. That's why it looks backwards "synchronous" actually means one after another: but the instructions are SYNCHRONIZED to the clock. "Asynchronous" means "at any time, I don't care when it happens": the instructions need not be synchronized to the clock. Yes, there is a dictionary definition, but you must make sure you are defining the correct situation. – Tom Padilla Jan 28 '16 at 15:41

1 Synchronous does *not* mean 'all at the same time' in computing. You are confusing synchronization with synchronous, and 'parallel versus series' with tea and sweet tea. Answer makes no sense whatsoever. – user207421 Aug 11 '16 at 4:09

6

Synchronous basically means that you can only execute one thing at a time. Asynchronous means that you can execute multiple things at a time and you don't have to finish executing the current thing in order to move on to next one.

answered Nov 7 '15 at 16:33

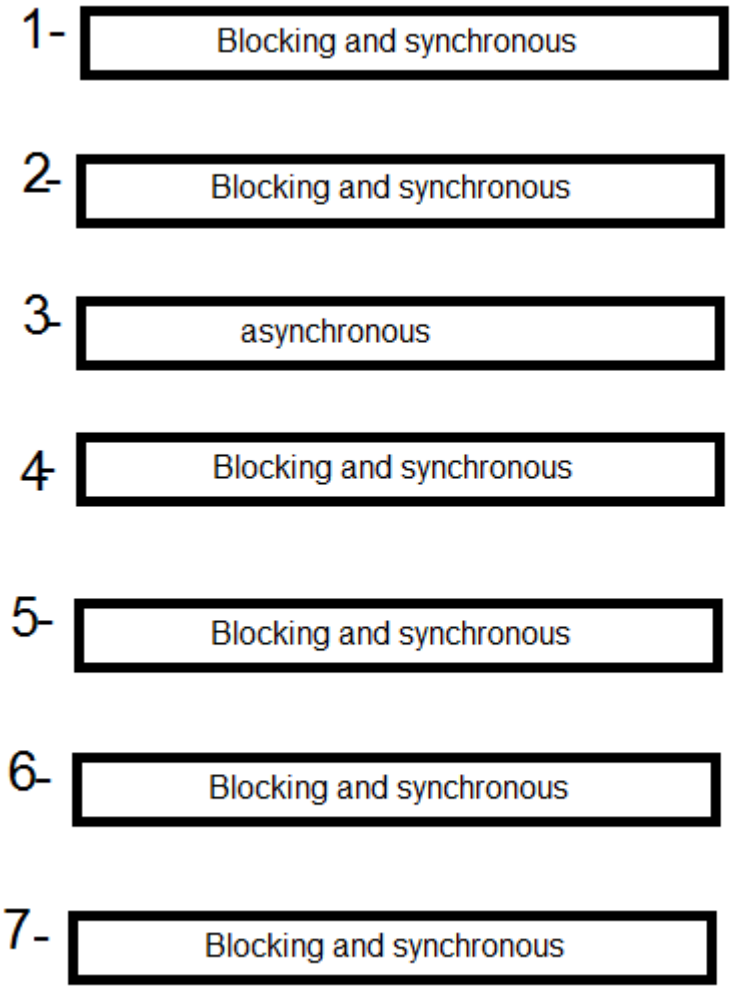
Mike
61 1 1

Isn't multiple things executing at the same time called Multithreading rather than asynchronous. – Sreekanth Karumanaghat Sep 13 '17 at 13:03

6

I created a gif for explain this, hope to be helpful: look, line 3 is asynchronous and others are synchronous. all lines before line 3 should wait until before line finish its work, but because of line 3 is asynchronous, next line (line 4), don't wait for line 3, but line 5 should wait for

line 4 to finish its work, and line 6 should wait for line 5 and 7 for 6, because line 4,5,6,7 are not asynchronous.



edited Mar 31 at 23:08

answered Mar 31 at 23:01

Abolfazl Miadian
631 7 9

4

An asynchronous operation does (most or all of) its work after returning to the caller.

edited May 5 '17 at 0:50
HopefullyHelpful
865 2 12 33

answered Apr 9 '14 at 14:45
Maxim Eliseev
1,380 2 22 31

the link is dead. – HopefullyHelpful May 4 '17 at 9:03

2

Use an example of instructions for making a breakfast

1. Pour a cup of coffee.
2. Heat up a pan, then fry two eggs.
3. Fry three slices of bacon.
4. Toast two pieces of bread.
5. Add butter and jam to the toast.
6. Pour a glass of orange juice.

If you have experience cooking, you'd execute those instructions asynchronously. you'd start warming the pan for eggs, then start the bacon. You'd put the bread in the toaster, then start the eggs. At each step of the process, you'd start a task, then turn your attention to tasks that are ready for your attention.

Cooking breakfast is a good example of **asynchronous** work that isn't parallel. One person (or thread) can handle all these tasks. Continuing the breakfast analogy, one person can make breakfast asynchronously by starting the next task before the first completes. The cooking progresses whether or not someone is watching it. As soon as you start warming the pan for the eggs, you can begin frying the bacon. Once the bacon starts, you can put the bread into the toaster.

For a parallel algorithm, you'd need multiple cooks (or threads). One would make the eggs, one the bacon, and so on. Each one would be focused on just that one task. Each cook (or thread) would be blocked synchronously waiting for bacon to be ready to flip, or the toast to pop.

Reference from Asynchronous programming concepts

edited May 16 at 10:05

answered Apr 10 at 7:46

1

In regards to the "*at the same time*" definition of synchronous execution (which is sometimes confusing), here's a good way to understand it:

Synchronous Execution: *All tasks within a block of code are all executed at the same time.*

Asynchronous Execution: *All tasks within a block of code are not all executed at the same time.*

answered Jul 11 '16 at 1:30

docta_faustus
586 1 12 30

I would agree with this more if you said 'effectively at the same time', or 'for practical purposes'.. I think the downvote was for the inaccurate statement that things *actually* are getting done at the same tiem. – Dean Radcliffe Dec 28 '16 at 15:27

1

I think a good way to think of it is a classic running Relay Race

Synchronous: Processes like members of the same team, they won't execute until they receive baton (end of the execution of previous process/runner) and yet they are all acting in sync with each other.

Asynchronous: Where processes like members of different teams on the same relay race track, they will run and stop, async with each other, but within same race (overall program execution).

Does it make sense?

answered Jan 11 '18 at 15:24

Sharif
169 1 10

1

A different english definition of Synchronize is Here

> Coordinate; combine.

I think that is a better definition than of "Happening at the same time". That one is also a definition, but I don't think it is the one that fits the way it is used in Computer Science.

So an asynchronous task is not co-coordinated with other tasks, whereas a synchronous task IS co-coordinated with other tasks, so one finishes before another starts.

How that is achieved is a different question.

answered May 23 '18 at 16:04

Greg Gum
12.9k 19 96 145

0

Synchronous means queue way execution one by one task will be executed. Suppose there is only vehicle that need to be share among friend to reach their destination one by one vehicle will be share.
In asynchronous case each friend can get rented vehicle and reach its destination.

edited Jan 13 '18 at 17:47

סטנלי גרונן
1,797 8 26 46

answered Jan 11 '18 at 15:29

Rohit
106 6

0

Yes synchronous means at the same time, literally, it means doing work all together. multiple human/objects in the world can do multiple things at the same time but if we look at computer, it says synchronous means where the processes work together that means the processes are dependent on the return of one another and that's why they get executed one after another in proper sequence. Whereas asynchronous means where processes don't work together, they may work at the same time(if are on multithread), but work independently.

edited Oct 4 '18 at 18:44

answered Oct 4 '18 at 15:48

Saptarshi
111 1 3
