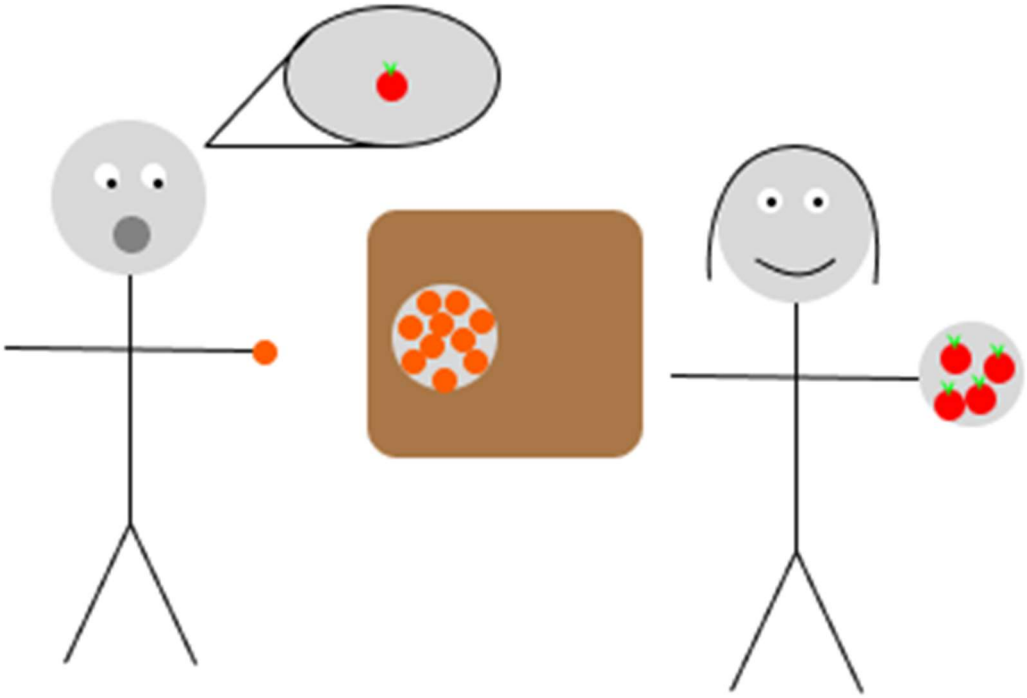
Samuel Nygard

The Critical Fruit Bowl Problem

Imagine you are in the kitchen, and you are about to take an apple from a fruit bowl. You start reaching out and when you are about to grab your apple your mum switches the bowl for oranges. Both you and your mum needed to interact with the fruit bowl, and this generally works perfectly. Unfortunately, when you are both interacting with the fruit bowl at the same time, a problem occurs. You could look at the fruit bowl and see apples, but when you reach out and grab a fruit you get an orange. This problem is called the critical section problem.



Let's get a definition, SmarterStudy UK has a nice one, "the critical section problem arises when shared resources are accessed by concurrent processes" ("Critical Section," n.d.). Whoa, what are all those words? What do you mean by share resources? What are concurrent processes? These are all fair questions you might be asking. Let's tackle those questions one at a time. Shared resources are things that multiple programs are going to use. In our analogy, it is the fruit bowl. Both you and your mum need access to it. Ok you might now think that makes a little sense, but you still might not understand the words concurrent or processes. Concurrent means, at the same time. Process is another word for a program. Back to our analogy, you and your mum are doing your tasks concurrently because you are doing them at the same time, and you are both processes. Your "program" is taking fruit and your mum's is switching the fruit.

Let us get a brief history of the critical section problem and the background you need to understand its solutions. When computers were young and inflexible, this problem did not exist. Computers could not run more than one process at a time, so two processes could not access the same data. In our analogy you and your

mum could reach for the fruit bowl at the same time because only you or your mum could exists at any one point. Since there was no critical section problems, programmers lived in a happy utopia. Until one day some smart person dared to ask, what if computers could be more flexible? This was one of the first steps that made computers go from amazing calculators to something that is in everyone's homes and now hands: iPhones. This newfound power of flexibility introduced lots of problems: one of which is the critical section problem.

Let's circle back to this concurrent thing mentioned a moment ago. What does "at the same time" mean for a computer? Well, a computer has a brain, it is called the central processing unit: the CPU. A single CPU can still only do one thing at a time; that fact has not changed since before flexible computers. Nowadays, computers have multiple CPUs, but let's pretend they only have one to keep things simple. So, processes that run concurrently aren't being done at the same time because that is impossible. So how does it work? The secret is speed, since your computer's brain is so fast it can rapidly switch between two processes making it look, and feel, like it is running both at the same time. In reality, your computer is just running one process at a time and, almost randomly, switching between them.

To make it clearer why this rapid and random switching causes the critical section problem, let's bring our analogy back. This time both you and your mum are only allowed to move when you are being pointed at by a puppet master. The puppet master knows you want to take fruit from the bowl, and they also know your mum wants to switch the fruit. The puppet master would switch who they were pointing to at random times. When pointing at you, you can grab a fruit. When they are pointing at your mum, she can switch the fruit. This works well until the random switch happens as you are about to grab a fruit. Leading to the problem described at the start. You expect an apple, but get an orange.

So how do we fix this? The first answer is to take the fruit really fast. Computers can only run one basic action at a time, so if "take fruit" were a basic action, the switching would not be able to happen during the action. Unfortunately taking fruit is complicated; you have to move your arm, grab the fruit with your hand and then pick it up. That is a lot of steps. So unfortunately, most tasks cannot be done in a single basic action on a computer, making this solution impractical.

The second answer, that is more useful, is a fruit pass, like a hall pass in elementary school. So how does the fruit pass work? The first thing you do when you want to interact with the fruit is grab the fruit pass or this solution does not work. Then once you have the fruit pass you can interact with the fruit, grab a fruit or change the fruit. Once you are done interacting with the fruit, put the fruit pass back or you are hogging the fruit pass and no one else can interact with the fruit. If the fruit pass was not there in the first place, you must wait until it shows up and take it before interacting with the fruit. Importantly, getting the fruit pass is really fast, and only takes one basic action, the first solution. This basic action was implemented because computer processes were struggling with the critical sections problem.

The last solution fixes the problem of having someone greedily keeping the fruit pass. The solution is to get a butler. The butler is the only person allowed to interact with the fruit. Instead of interacting with the fruit yourself, you ask the butler to get you a fruit. Your mum must also ask the butler to swap the fruit. Since there is only one butler, they cannot do both at once. If you show up and the butler is already helping someone else, you will have to wait until they are available. Also, while the butler is interacting with the fruit for you, you will have to wait until they say the task is done. In reality, the butler would be a web server and the people would be processes asking to access data. The process would ask the web server to get some data. Then the process would wait for the web server to give the data to the process, solving the critical section problem.

With the critical section problem solved, it opened the door for more complex processes that run on a single computer. Most websites primarily implement their logic with a language called JavaScript. To speed things up, JavaScript won't wait for a task to be completed before starting others, so the critical section problem is very relevant. The solutions to the critical section problem are not automatically applied so programmers must keep them in mind while programming JavaScript amongst other things.

Reference

Critical Section. StudySmarter UK. (n.d.). https://www.studysmarter.co.uk/explanations/computer-science/computer-programming/critical-section/