Part II

Concurrency

24

A Dialogue on Concurrency

Professor: *And thus we reach the second of our three pillars of operating systems:* ***concurrency****.*

Student: *I thought there were four pillars...?*

Professor: *Nope, that was in an older version of the book.*

Student: *Umm... OK. So what is concurrency, oh wonderful professor?*

Professor: *Well, imagine we have a peach –*

Student: *(interrupting) Peaches again! What is it with you and peaches?*

Professor: *Ever read T.S. Eliot? The Love Song of J. Alfred Prufrock, "Do I dare to eat a peach", and all that fun stuff?*

Student: *Oh yes! In English class in high school. Great stuff! I really liked the part where –*

Professor: *(interrupting) This has nothing to do with that – I just like peaches. Anyhow, imagine there are a lot of peaches on a table, and a lot of people who wish to eat them. Let's say we did it this way: each eater first identifies a peach visually, and then tries to grab it and eat it. What is wrong with this approach?*

Student: *Hmmm... seems like you might see a peach that somebody else also sees. If they get there first, when you reach out, no peach for you!*

Professor: *Exactly! So what should we do about it?*

Student: *Well, probably develop a better way of going about this. Maybe form a line, and when you get to the front, grab a peach and get on with it.*

Professor: *Good! But what's wrong with your approach?*

Student: *Sheesh, do I have to do all the work?*

Professor: *Yes.*

Student: *OK, let me think. Well, we used to have many people grabbing for peaches all at once, which is faster. But in my way, we just go one at a time, which is correct, but quite a bit slower. The best kind of approach would be fast and correct, probably.*

Professor: *You are really starting to impress. In fact, you just told us everything we need to know about concurrency! Well done.*

Student: *I did? I thought we were just talking about peaches. Remember, this is usually a part where you make it about computers again.*

Professor: *Indeed. My apologies! One must never forget the concrete. Well, as it turns out, there are certain types of programs that we call* ***multi-threaded*** *applications; each* ***thread*** *is kind of like an independent agent running around in this program, doing things on the program's behalf. But these threads access memory, and for them, each spot of memory is kind of like one of those peaches. If we don't coordinate access to memory between threads, the program won't work as expected. Make sense?*

Student: *Kind of. But why do we talk about this in an OS class? Isn't that just application programming?*

Professor: *Good question! A few reasons, actually. First, the OS must support multi-threaded applications with primitives such as* ***locks*** *and* ***condition variables****, which we'll talk about soon. Second, the OS itself was the first concurrent program – it must access its own memory very carefully or many strange and terrible things will happen. Really, it can get quite grisly.*

Student: *I see. Sounds interesting. There are more details, I imagine?*

Professor: *Indeed there are...*