IIT Madras

ONLINE DEGREE

Computational Thinking
Professor. Madhavan Mukund
Department of Computer Science
Chennai Mathematical Institute
Professor. G Venkatesh
Indian Institute of Technology, Madras
Concurrency Involved in Input/Output Operations

Professor. Madhavan Mukund: So, till now we have been assuming that we are already inside our program and the program is, or our code is calling procedures. And so there is no, I would say, interaction between the code and users as we normally use our see when we are normally using our computer to do something, we are using a browser, we type some address that goes in, or we say type a search query and then get a result and then we click on it, and then we go somewhere, or we fill a form, then you then you say, I am done filling it, you want to, when you say submit, it will give you an answer. So, that part of this interaction between…

Professor. G. Venkatesh: What is called IO?

Professor. Madhavan Mukund: This input and output. Between the user…

Professor. G. Venkatesh: Our so far we have done everything has cards is program has the cards. So, we are on the card. We give it a problem to work on it works on the problem, and it gives a result.

Professor. Madhavan Mukund: and the result we have not really transferred it out of the system.

Professor. G. Venkatesh: It is in Variable.

Professor. Madhavan Mukund: Everything is kind of sitting inside, but we have to eventually get it to interacted. So, we have to feed it some interaction…

Professor. G. Venkatesh: But define interaction between objects.

Professor. Madhavan Mukund: Interaction between objects, as you saw. So there, we assume that there is some that well, the main difference I would expect is that when the interaction happens between objects, so they are all programmed, so they know what to expect. So, you have parameters, you know that if you call this procedure, you have to write these 3, protocol

variables. So, they agree on what is what the what one program will get from each other as the input that is the parameters and what is the format in which the return output will come. So, that is fixed. And there is no confusion as to whether it is completed or not completed or something, whereas here,…

Professor. G. Venkatesh: You try and create a protocol procedure with the user.

Professor. Madhavan Mukund: So, we have to do that. So, they have to have a way of doing that. So, typically, what do we do?

Professor. G. Venkatesh: Forms and you said the form.

Professor. Madhavan Mukund: so in a form, usually that sort. So, you type all this thing, and then…

Professor. G. Venkatesh: It create it and tells you fill this here, fill the name here, fill the

Professor. Madhavan Mukund: so it tells you what you can fill in the sometimes it will say, you know, a telephone number and you start putting some letters, only digits are allowed. So. some annoying things it will say. But then finally to the to send all that information, there'll be a button.

Professor. G. Venkatesh: Submit it will say.

Professor. Madhavan Mukund: And when we submit if you have left or something, it will say this field should not be blank, it reset it, but also it will make sure that everything is filled. So, it will say this is not this is compulsory, you have not put your name, you have not put your email address, not put your date of birth, something it will say. So, there is a protocol there. That is input.

Professor. G. Venkatesh: So, it is so this form. So, it is the this program is asking the user for some data. So, program, the user and the program are concurrent? There Yeah, kind of right.

Professor. Madhavan Mukund: Kind of Yeah.

Professor. G. Venkatesh: So, program basically is executing, it needs some input from the user. So it will, it will request user for some input.

Professor. Madhavan Mukund: So, it displays a form displays a form.

Professor. G. Venkatesh: Displays a form program could continue also, but for the moment, assume that the program is waiting for this input. So, if it is waiting, it just presents the form. User fills the form,

Professor. Madhavan Mukund: Then presses Submit.

Professor. G. Venkatesh: That is how the user tells the reader. So, that so the user is telling the program that is finished?

Professor. Madhavan Mukund: finish writing into your tray in some sense. So, this the tray,

Professor. G. Venkatesh: this is the tray. The form is a tray.

Professor. Madhavan Mukund: the form is the program's stray. And you are writing it into the tray. So, you are producing the information, which it will then consume.

Professor. G. Venkatesh: So, when he says submit, you are indicating that whatever he said is ready. Dosa and Chutney business.

Professor. Madhavan Mukund: Because he does not know he does not know how many letters go. Name, you might have a long name, short name. That kind of information is there it can decide that after 75 things have been typed 75 is over.

Professor. G. Venkatesh: So, that is why the submit button is there. So, when you press the submit button, at this point in time the user has finished now user cannot do anything. Over and now he cannot write anymore. If made mistake, then mistake over. Now the data is consumed by them.

Professor. Madhavan Mukund: And now maybe it might be that it wants more information. Like for example, you tried to book a ticket. So first, it will ask you to select the train on this railway website. Now, if it happens that on the day in which you are booking the train, there is a seat available and you can proceed, then it will give you a new thing saying now enter your name, your age,…

Professor. G. Venkatesh: or decided to check whether there is space thing available.

Professor. Madhavan Mukund: So, it is a multistage thing. Whenever it needs it, put it will put out a tray and ask.

Professor. G. Venkatesh: Similarly, it may want to give us a result also. Your booking is successful. We want to say. And this is your this thing. What is it your reservation number.

Professor. Madhavan Mukund: PNR.

Professor. G. Venkatesh: PNR. So, that it has to display. So, there will be an output of also. So, it is first make sure that you have see the output, it cannot just present it and go.

Professor. Madhavan Mukund: So, usually, so sometimes it will put something on the screen and it will say, it will be okay. So, you can till you say okay it will remain on screen. So, it cannot guarantee that you have seen it. But when you say, okay, it knows that you have seen it consciously decided to go past.

Professor. G. Venkatesh: So, this input from the user to the program, and output from the program to the user is exactly the same thing. Concurrency.

Professor. Madhavan Mukund: It is very, very similar. Yes,

Professor. G. Venkatesh: Same thing? And we basically have tray so communicate. And there is some atomicity model or here. Only when I finished writing, I will submit then the signal consume while I am writing he is not reading?

Professor. Madhavan Mukund: Not from this tray, but you might be doing something else,

Professor. G. Venkatesh: You may be doing something else. So, input output generally is like concurrent programming. So, there is nothing further to know about Input Output nothing fancy about.

Professor. Madhavan Mukund: But it is important to realize that there is concurrency, that is the only think.

Professor. G. Venkatesh: But sometimes actually, when the guy can write he may not give okay. And sometimes he writes and he goes off. And he writes again, and he keeps writing and then my screen will keeps scrolling and showing data, right.

Professor. Madhavan Mukund: So, for example, if you are doing something like a chat,…

Professor. G. Venkatesh: That it does not wait for it.

Professor. Madhavan Mukund: It does not wait, you just keep showing more. And then you can do but you have a button scroll by which you can go back and go back and see. And then chat also, for instance, that no submit usually, so there may be a kind of an arrow Submit. But usually if you say return, then your message goes as it goes. So, there is some special convention there also that sometimes it might, so sometimes you might actually want to put a new line or say return and it is gone.

So, you have to say control return or something careful to do that, so they could be different. Depending on the program, you are interacting with this signal for end of input at your site could be an explicit button, or it could be a special character like return and all that. But once it is done, it goes. And when it comes back again, it could persistently display something until you press a button saying, Okay, I am done. Or it could just keep on putting things and then it is up to you to decide how to go back and read the older ones. But it does not stop you from getting new messages.