Concurrency

Introduction

So far we have seen the virtualization of CPU and Main memory by the Operating System. Now, in this module, we will be talking about a feature in modern day OS that amazes us all. We are talking about the ability to run multiple processes within a process simultaneously.

This is called **Concurrency** and it will be the second piece of OS.

To elaborate more on it, let us take an example. When we run a browser on our system, then you can open multiple tabs. Each tab can be used to open a different website. It may happen that one tab is playing music and simultaneously another tab is used to create a document.

Crux of this Section

Now, the question arises how does OS virtualizes memory and CPU for such processes and what are the new challenges that it comes across and how does it solve it.

Flow of Lecture

Before we begin the flow of this lecture, let's use the correct term for multiple processes within a process. These internal processes are called **threads**. In this lecture, we will be answering following questions and discussing following concepts:

1. How are threads different from processes in terms of memory allocation?
2. What are the advantages of using threads over processes?
3. What are the issues that come when multiple threads are updating the value of a shared variable?
4. What are the various solutions and strategies used to solve these issues? In this lecture, we will discuss the following three solutions: **Locks, Conditional Variables and Semaphores**.
5. These solutions give rise to a bug called **Deadlock**. We will discuss its causes, avoidance and prevention techniques.