Problem Definition: Sleeping Barber Problem

In computer science, the **sleeping barber problem** is a classic inter-process Communication and synchronization problem between multiple operating system processes. The problem is analogous to that of keeping a barber working when there are customers, resting when there are none, and doing so in an orderly manner.

The analogy is based upon a hypothetical barber shop with one barber. The barber has one barber's chair in a cutting room and a waiting room containing a number of chairs in it. When the barber finishes cutting a customer's hair, he dismisses the customer and goes to the waiting room to see if there are others waiting. If there are, he brings one of them back to the chairs and cut their hair. If there are none, he returns to the chair and sleeps in it.

Each customer, when they arrive, looks to see what the barber is doing. If the barber is sleeping, the customer wakes him up and sits in the cutting room chair. If the barber is cutting hair, the customer stays in the waiting room. If there is a free chair in the waiting room, the customer sits in it and waits their turn. If there is no free chair, the customer leaves.

Based on a naïve analysis, the above decisions should ensure that the shop functions correctly, with the barber cutting the hair of anyone who arrives until there are no more customers, and then sleeping until the next customer arrives.

Introduction and method: Sleeping Barber Problem

Sleeping barber problem is based on the concept of **semaphore** which is one of the very important topics of the subject **Operating Systems** proposed by **Edsger Dijkstra** for **Inter process communication.** In computer science, a **semaphore** is a variable or abstract data type used to control access to a common resource by multiple processes in a concurrent system such as a multitasking operating system.; We decided to take on the project on the programming language which is the building block of all basic programming languages called **C**. We developed our own semaphore instead of using the package provided by the programming language. We implemented the **wait** and **signal** functions of the semaphore concept which is used to increase and decrease the count of instances in this case the barber. Whenever the barber is busy doing some task including sleeping the value of the **mutex**(variable used to control process synchronization) is set to 0 and when he is ready to take on a new customer the value of **mutex** is set to 1. In the similar fashion the value of **mutex** is 0 when the customer is waiting for the barber in the waiting room and the value of **mutex** is 1 when he enters the shop and checks for the customer. Whenever the barber goes to sleep it sets the value of the **flag** to **1** so whenever the customer comes it checks whether the customer is sleeping or not if it does it wakes him up and sets the value of the **flag** to **0.** Whenever the barber finished serving a customer it goes up to the waiting room and checks for the customers if there are no he goes back and sleep until a customer wakes him up. We used the delay function of the dos package to have a waiting impact on the output screen.

Programming language used: **C**

Tools used for development of project : **Turbo C++ Desktop Application**

OS concepts used: **Semaphores, Process Synchronization**

Operating system used: **Windows 10 Home basic**