

Project Title:

# Barber Shop Problem

## Submitted by:

## Qurat Ul Ain 2021-CE-02

## Noor Fatima 2021-CE-07

## Syeda Aliya Zahra 2021-CE-16

## Ansa Aslam 2021-CE-21

## Submitted to:

Darakhshan Abdul Ghaffar

## Course:

**Operating Systems Lab**

## Semester:

## **5th**

## Date of Submission:

11 November, 2023

# Department of Computer Engineering University of Engineering and Technology, Lahore

Table of Contents

Contents

[Barber Shop Problem 1](#_Toc150434246)

[Abstract 2](#_Toc150434258)

[Introduction 2](#_Toc150434259)

[Problem Statement 2](#_Toc150434260)

[Proposed methodology: 3](#_Toc150434261)

[Expected Outcome/Results 4](#_Toc150434262)

[Applications in Operating Systems 5](#_Toc150434263)

[References 7](#_Toc150434264)

# 

# Abstract:

# Introduction:

The barbershop problem is a classic synchronization challenge often used in computer science and operating systems courses. It illustrates how to coordinate the actions of multiple threads, representing customers and a barber, to simulate the functioning of a barbershop. The problem involves managing shared resources, such as waiting room chairs and the barber's state, to ensure proper synchronization and adherence to specified constraints. Key aspects include customers entering the shop, waiting for haircuts, and the barber attending to them, all while maintaining order and preventing conflicts in a concurrent environment.

# Problem Statement:

The original barbershop problem was proposed by Dijkstra. A variation of it appears in Silberschatz and Galvin’s Operating Systems Concepts. A barbershop consists of a waiting room with n chairs, and the barber room containing the barber chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop. If the barber is busy, but chairs are available, then the customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber. Write a program to coordinate the barber and the customers.

To make the problem a little more concrete, I added the following information:

* Customer threads should invoke a function named getHairCut.
* If a customer thread arrives when the shop is full, it can invoke balk, which does not return
* The barber thread should invoke cutHair.
* When the barber invokes cutHair there should be exactly one thread invoking getHairCut concurrently

Write a solution that guarantees these constraints.

# Proposed methodology:

1. **Identifying Shared Resources:**
   * The shared resources in our problem are following
     + The waiting room with a limited number of chairs.
     + The barber's chair and state (awake or sleeping).
2. **Selecting Synchronization Mechanisms:**
   * We have to choose appropriate synchronization primitives or mechanisms to control access to shared resources and coordinate the actions of customers and the barber.
   * Common synchronization mechanisms include semaphores, mutexes (mutual exclusion), condition variables, and events.
3. **Defining Thread Behaviors:**
   * We have to create functions or methods that represent the behaviors of customers and the barber. These functions should use synchronization mechanisms to ensure safe and orderly execution.
   * Like specifying how customers will react if the shop is full or if the barber is asleep.
4. **Initialization:**
   * We have to initialize the shared resources, synchronization mechanisms, and any other necessary variables.
   * Set initial states, such as the barber initially being asleep.
5. **Create and Start Threads:**
   * We have to create and start the barber thread that will be responsible for cutting hair.
   * In the same way we have to create and start multiple customer threads to simulate customer arrivals and behavior.
6. **Customer Behavior:**
   * In the customer behavior function the customers will
     + Attempt to enter the waiting room
     + If a chair is empty, they will sit on it
     + If no chair is available, they will not wait and left the shop
     + Wake up the barber if needed.
     + Get haircuts and leave the shop.
7. **Barber Behavior:**
   * In the barber behavior function, barber will:
     + Waits for customers to arrive.
     + If no customer is in the waiting room it will go to sleep
     + Cuts a customer's hair.
     + Manages the state of the waiting room and chairs.
8. **Synchronization:**
   * We have to make sure that according to the chosen synchronization mechanisms customers and the barber access shared resources and execute their tasks in a coordinated manner.
   * Ensure that only one customer gets a haircut at a time it will prevent race conditions.
9. **Testing and Debugging:**

We have to:

* + Test the program with different scenarios to ensure that it meets the specified constraints.
  + Debug any synchronization issues, race conditions, or unexpected behaviors.

1. **Optimization and Customization:**
   * We have to fine-tune the program and synchronization mechanisms as needed to make it more efficient, robust, or aligned with specific requirements.

# Expected Outcome/Results:

The expected outcomes of a successful solution to the Barbershop problem include the following:

1. Customers are able to enter the waiting room and wait if there are available chairs.

2. Customers leave the barbershop if the waiting room is full.

3. The barber serves customers in a first-come, first-served order.

4. The barber waits if no customers are present in the waiting room.

5. Customers are served one at a time, and new customers can enter the waiting room as others leave.

6. There are no race conditions, deadlocks, or other synchronization issues that can disrupt the barbershop's operation.

The specific implementation of a solution may vary depending on the programming language and synchronization mechanisms used. Common synchronization tools like semaphores, mutexes, and condition variables help ensure that the barbershop problem is solved correctly.

Solving the Barbershop problem is a valuable exercise for learning about concurrency and synchronization in operating systems and multithreaded applications. Different solutions may exist, but the key is to achieve mutual exclusion and coordination to ensure the correct operation of the barbershop.

# Applications in Operating Systems:

Some of the applications of this problem is operating systems are given below:

1. **Resource Allocation:**

* The barbershop problem can represent scenarios where resources, such as CPU time or memory, need to be allocated to multiple processes or threads in an operating system. Customers correspond to processes or threads requesting resources, and the barber represents the resource manager, ensuring fair allocation.
* Queue Management: This problem models scenarios where tasks or processes are queued and executed in an orderly manner. Customers represent tasks waiting in the queue, and the barber symbolizes the scheduler that selects and processes tasks.

1. **Concurrency Control:**

* In multi-threaded or multi-process environments, the barbershop problem exemplifies the importance of synchronization for preventing race conditions and ensuring safe concurrent access to shared resources.

1. **Real-time Systems:**

* The problem can be applied in real-time operating systems to represent the scheduling of real-time tasks with specific priorities and deadlines. Customers with higher priority tasks are serviced first.
* Resource Sharing: The barbershop problem illustrates the challenges of sharing limited resources equitably among competing entities. It's relevant in memory management, file access, and network communication, where multiple processes need access to shared res

# References:

# [https://www. https://www.os-book.com/geeksforgeeks.org/](https://www.geeksforgeeks.org/)

# <https://stackoverflow.com/>

# <https://github.com/>

# <https://www.os-book.com/>