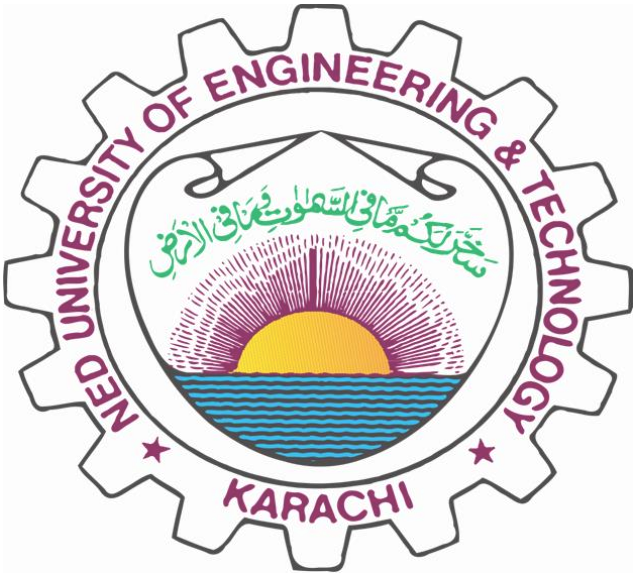
CS - 317: Operating System.



Implementing Barber-Customer Problem In Process Synchronization.

Group members:

1. ZOBIA KHAN (CS - 100)
2. MUHAMMAD SAAD (CS - 092)
3. AIMEN EJAZ (CS - 074)

SUBMITTED TO:

MISS UROOJ

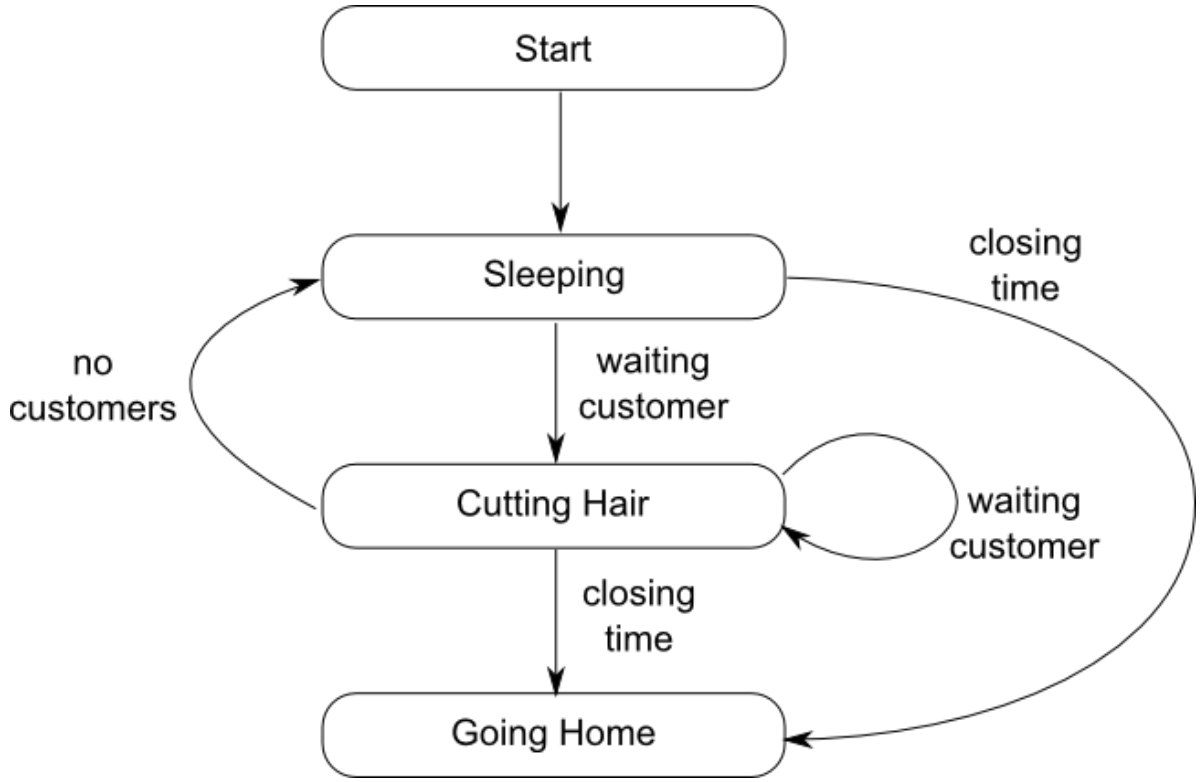
DATE: 12 AUGUST 2020.

What is the sleeping barber process?

It is an Inter Process Communication Problem.

How does it work?

The analogy is based upon a hypothetical barber shop with one barber. There is a barber shop which has one barber, one barber chair, and n chairs for waiting for customers if there are any to sit on the chair. If there is no customer, then the barber sleeps in his own chair. When a customer arrives, he has to wake up the barber. If there are many customers and the barber is cutting a customer's hair, then the remaining customers either wait if there are empty chairs in the waiting room or they leave if no chairs are empty.



Language : Python 3

```
import threading
import time
import random
import queue

barber = 1
customers = int(input("Enter the number of customers for a day : "))  #user input customers
seats = int(input("Enter the number of seats available for sitting : "))  #user input seating options
customer_arrival_wait = 2    #time for next customer

print('*************************************************')
print('               BARBER SHOP IS OPENING             ')
print('*************************************************')

print('*************************************************')
print('              CUSTOMER STARTS ARRIVING            ')
print('*************************************************')

def arrival_wait():
        time.sleep(customer_arrival_wait * random.random())   #time for next customer * rand function to disturb the arrival timings.

class Barber(threading.Thread):
        condition = threading.Condition() # barber either sleeping aur wake up
        should_stop = threading.Event() # waiting room empty, every customer is served

        def __init__(self, ID):
                super().__init__()
                self.ID = barber

        def run(self):
                while True:
                        try:
                                current_customer = wait_room.get(block=False) #thread won't wait/block in queue
                        except queue.Empty:             #actives when waiting room = 0
                                if self.should_stop.is_set():       #when customer count gets 0
                                        return

                                print(f"No customers in the waiting area, barber is sleeping :)")
                                with self.condition:
```

```
                                        self.condition.wait()  #sleep/wait for customer to wake up
                                        print(f"Customer wakes up barber")
                                else:
                                        current_customer.cutHair(self.ID) # customer getting hair cut

class Customer(threading.Thread):
        time_duration_haircut = 6      #time for one haircut

        def __init__(self, ID):
                super().__init__()
                self.ID = ID + 1

        def getHairCut(self):
                time.sleep(self.time_duration_haircut * random.random())

        def cutHair(self, barber_ID):  #called from barber thread
                print(f"Customer {self.ID}'s turn arrives, jumps towards barber's room and sits on the barber chair '")
                print(f"Barber started cutting hair of customer {self.ID}")
                self.getHairCut()
                print(f"Barber finished cutting hair of customer {self.ID}")
                self.serviced.set()    #customer leaves after getting serviced

        def run(self):
                self.serviced = threading.Event()

                try:        #checking space in wait room
                        wait_room.put(self, block=False)
                except queue.Full:  #wait room is full, leave
                        print(f"Waiting room is full, {self.ID} is leaving")
                        return

                print(f"Customer {self.ID} arrived, sitting in the waiting room")
                with Barber.condition:
                        Barber.condition.notify() # barber waking up if sleeping

                self.serviced.wait()  #waiting for haircut

if __name__ == "__main__":

```

```

        global locks
        locks = threading.Lock()    #initiating lock in threads
        total_customers = []        #list of total customers
        wait_room = queue.Queue(seats)        #number of seats

        barber_thread = Barber(1)   #barber thread
        barber_thread.start()

        for order in range(customers): #customer thread
                arrival_wait()
                customer = Customer(order)
                locks.acquire()
                total_customers.append(customer)
                locks.release()
                customer.start()

        for customer in total_customers:
                customer.join()  # waiting for all customers to leave

        time.sleep(1) #time to clean the shop before closing
        Barber.should_stop.set()        #tough day is finised for the barber :)

print('*********************************************************')
print('               BARBER SHOP IS CLOSED                     ')
print('*********************************************************')
```

TEST CASES:

CEP: Barber Shop Implemention Semaphores			
Step	**test data**	**Actual result**	**Status**
1	Customers = 4 , Seats = 10	As Expected	Pass
2	Customers = 5 , Seats = 5	As Expected	Pass
3	Customers = 10, Seats = 4	As Expected	Pass
post conditions:	There is only 1 Barber.		

OUTPUTS:

A. OUTPUT 1

```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: C:¥Users¥ce¥Desktop¥OS_CEA_barber.py ==================
Enter the number of customers for a day : 4
Enter the number of seats available for sitting : 10
**************************************************
            BARBER SHOP IS OPENING
**************************************************
**************************************************
           CUSTOMER STARTS ARRIVING
**************************************************
No customers in the waiting area, barber is sleeping :)
Customer 1 arrived, sitting in the waiting room
Customer wakes up barber
Customer 1's turn arrives, jumps towards barber's room and sits on the barber chair '
Barber started cutting hair of customer 1
Customer 2 arrived, sitting in the waiting room
Customer 3 arrived, sitting in the waiting room
Customer 4 arrived, sitting in the waiting room
Barber finished cutting hair of customer 1
Customer 2's turn arrives, jumps towards barber's room and sits on the barber chair '
Barber started cutting hair of customer 2
Barber finished cutting hair of customer 2
Customer 3's turn arrives, jumps towards barber's room and sits on the barber chair '
Barber started cutting hair of customer 3
Barber finished cutting hair of customer 3
Customer 4's turn arrives, jumps towards barber's room and sits on the barber chair '
Barber started cutting hair of customer 4
Barber finished cutting hair of customer 4
No customers in the waiting area, barber is sleeping :)
**************************************************
            BARBER SHOP IS CLOSED
**************************************************
>>>
```

B. OUTPUT 2

```
================== RESTART: C:¥Users¥ce¥Desktop¥OS_CEA_barber.py ================
Enter the number of customers for a day : 5
Enter the number of seats available for sitting : 5
**************************************************
            BARBER SHOP IS OPENING
**************************************************
**************************************************
           CUSTOMER STARTS ARRIVING
**************************************************
No customers in the waiting area, barber is sleeping :)
Customer 1 arrived, sitting in the waiting room
Customer wakes up barber
Customer 1's turn arrives, jumps towards barber's room and sits on the barber chair '
Barber started cutting hair of customer 1
Customer 2 arrived, sitting in the waiting room
Customer 3 arrived, sitting in the waiting room
Customer 4 arrived, sitting in the waiting room
Barber finished cutting hair of customer 1
Customer 2's turn arrives, jumps towards barber's room and sits on the barber chair '
Barber started cutting hair of customer 2
Customer 5 arrived, sitting in the waiting room
Barber finished cutting hair of customer 2
Customer 3's turn arrives, jumps towards barber's room and sits on the barber chair '
Barber started cutting hair of customer 3
Barber finished cutting hair of customer 3
Customer 4's turn arrives, jumps towards barber's room and sits on the barber chair '
Barber started cutting hair of customer 4
Barber finished cutting hair of customer 4
Customer 5's turn arrives, jumps towards barber's room and sits on the barber chair '
Barber started cutting hair of customer 5
Barber finished cutting hair of customer 5
No customers in the waiting area, barber is sleeping :)
**************************************************
            BARBER SHOP IS CLOSED
**************************************************
>>>
```

C. OUTPUT 3

```
========== RESTART: C:¥Users¥ce¥Desktop¥OS_CEA_barber.py ==========
Enter the number of customers for a day : 10
Enter the number of seats available for sitting : 4
**************************************************
            BARBER SHOP IS OPENING
**************************************************
**************************************************
           CUSTOMER STARTS ARRIVING
**************************************************
No customers in the waiting area, barber is sleeping :)
Customer 1 arrived, sitting in the waiting room
Customer wakes up barber
Customer 1's turn arrives, jumps towards barber's room and sits on the barber chair '
Barber started cutting hair of customer 1
Customer 2 arrived, sitting in the waiting room
Customer 3 arrived, sitting in the waiting room
Customer 4 arrived, sitting in the waiting room
Customer 5 arrived, sitting in the waiting room
Waiting room is full, 6 is leaving
Waiting room is full, 7 is leaving
Barber finished cutting hair of customer 1
Customer 2's turn arrives, jumps towards barber's room and sits on the barber chair '
Barber started cutting hair of customer 2
Customer 8 arrived, sitting in the waiting room
Barber finished cutting hair of customer 2
Customer 3's turn arrives, jumps towards barber's room and sits on the barber chair '
Barber started cutting hair of customer 3
Customer 9 arrived, sitting in the waiting room
Barber finished cutting hair of customer 3
Customer 4's turn arrives, jumps towards barber's room and sits on the barber chair '
Barber started cutting hair of customer 4
Customer 10 arrived, sitting in the waiting room
Barber finished cutting hair of customer 4
Customer 5's turn arrives, jumps towards barber's room and sits on the barber chair '
Barber started cutting hair of customer 5
Barber finished cutting hair of customer 5
Customer 8's turn arrives, jumps towards barber's room and sits on the barber chair '
Barber started cutting hair of customer 8
Barber finished cutting hair of customer 8
Customer 9's turn arrives, jumps towards barber's room and sits on the barber chair '
Barber started cutting hair of customer 9
Barber finished cutting hair of customer 9
Customer 10's turn arrives, jumps towards barber's room and sits on the barber chair '
Barber started cutting hair of customer 10
Barber finished cutting hair of customer 10
No customers in the waiting area, barber is sleeping :)
**************************************************
            BARBER SHOP IS CLOSED
**************************************************
>>>
```