CS-329 Operating Systems

Submitted to: Miss Urooj Ainuddin
Dated: 24th July, 2022

Batch 2019

COMPLEX ENGINEERING PROBLEM

The Barbershop Problem

Submitted By:

CS-19302 Muhammad Jan

CS-19303 Musfirah Fayyaz

CS-19304 Faseeh U Rehman

SYNCHRONIZATION

Synchronization means coordinating the execution of processes such that no two processes access the same shared resources and data. Synchronization should be there in an operating system. It helps to maintain the consistency of data by using variables so that only one process can make changes to the shared memory at a time.

CLASSICAL SYNCHRONIZATION PEOBLEM

The Barbershop Problem

FCFS (First Come First Serve) ALGORITHM

First come first serve (FCFS) scheduling algorithm has been used. It is Non Pre-emptive algorithm. It simply schedules the jobs according to their arrival time on first come first serve basis.

CODE:

```
import threading    # import threading module
from time import sleep as sleeep    # import sleep function from time module

# This Program works on First Come First Serve (FCFS) basis

#Semaphores
customers = threading.Semaphore(0)    # semaphore for customers
barbers= threading.Semaphore(0)     # semaphore for barber
mutex= threading.Semaphore(1)    # semaphore for mutual exclusion

sleep=0   # For ensuring, if barber sleeping or not; Binary variable(0 or 1): 1 for sleeping & 0 for not sleepi
waiting=0 # Number of customers waiting to get their hair cut done
total=0   # Total number of customers(including the one getting hair cut too)
waiting_lst=[]  # contains the name of customer sitting on waiting chair
customer_on_barbers_chair="" # contains the name of customer currently getting hair cut
```

```
def barber(): # barber will cut hair of customer
    global total, waiting, sleep , waiting_lst, customer_on_barbers_chair # global variables
    while True:
        print("Barber is looking for more customers , total waiting customers are : ", waiting, "\n")

        if (len(waiting_lst)>0): # if there is someone on waiting chair
            print("waiting customers are : " , waiting_lst)

        if (waiting==0 and total==0): #if no customers are in the shop, barber will go to sleep
            print("(: , no customers are in the shop, Barber is going to sleep")
            sleep=1
        customers.acquire() # semwait
        mutex.acquire()    # semwait

        if(waiting>0):  # if customer waiting
            waiting-=1
        barbers.release()  # semsignal
        mutex.release()    # semsignal
        sleeep(1) # wait for 1 sec
        cut_hair() # barber cut hair
        sleeep(4)  # wait for 4 sec
        print(f"Barber is done with the hair cutting of {customer_on_barbers_chair}\n")

        if(len(waiting_lst)>0 and  customer_on_barbers_chair in waiting_lst): # if customer on waiting chair
            waiting_lst.remove(customer_on_barbers_chair)  # remove customer from list who is done with service
        customer_on_barbers_chair = ""
        total -=1
```

```
def customer(name): # customer will enter the shop and get hair cut
    global total, waiting, sleep , waiting_lst, customer_on_barbers_chair , chairs # global variables
    mutex.acquire() # semwait

    if (waiting < chairs or total==0): # if chairs are available or no customer in shop

        if (total==0):
            total+=1
            print(f"Customer: {name} has entered in barber shop \n")
        else:
            total+=1
            waiting+=1
            waiting_lst.append(name) # add name of customer in waiting list
            print(f"{name} has entered waiting room , total waiting customers :", waiting,"\n")
        customers.release()  # semwait
        mutex.release()      # semwait
        barbers.acquire()    # semsignal

        if (sleep==1): # barber is sleeping
            print(f"{name} is waking up the barber because he is sleeping\n")
            sleep=0 # barber has awaken
        customer_on_barbers_chair=name
        get_hair_cut(name) # customer get hair cut
    else: # no chair is available means shop is full
        mutex.release() # semsignal
        balk(name)   # leave the shop
```

```
def get_hair_cut(name):   # customer will call barber to get hair cut
    print(f"Customers {name} want Hair Cut\n")
    return

def cut_hair():  # barber will cut hair of customer
    print(f"Barber is cutting hairs of {customer_on_barbers_chair} \n")
    return

def balk(name):  # if no seats available, customer will leave the shop
    print(f"{name} is trying to enter waiting room...")
    print("But barber is busy and no seats are available")
    print(f"{name} is leaving the shop...")

print("\n Made by : \n Muhammad Jan (CS-19302) \n Musfirah Fayyaz (CS-19303) \n Faseeh U Rehman (CS-19304) \n")
print("<"+"="*50,"Welcome to Barber Shop","="*50+">","\n") # welcome message

barber_thread= threading.Thread(name="Barber" , target=barber) # create thread for barber
barber_thread.start() # start thread for barber
cust_threads=[] # list of customer threads
customer_lst=[] # list of customer names
```

5

```
while True:
    global chairs # number of chairs in shop
    if (len(cust_threads)): # if there is customer threads
        for t in cust_threads: # join all customer threads
            t.join()
    if(waiting==0 and len(customer_on_barbers_chair)==0):

        # if no customers are in the shop, barber will go to sleep
        # and if no customer is on barbers chair, User will be asked to run program again

        print("Select options \n 1 ---> Start Barber Shop \n 2 ---> Exit \n")
        start=int(input("Enter your choice : ")) # user will be asked to run program again

        if(start==0): # if user wants to run program again
            break # break the loop
        no_of_customers= int(input("How many customer are there ? ")) # number of customers in shop
        for i in range(no_of_customers):
            temp_customer="Customer "+str(i) # create random customer name
            customer_lst.append(temp_customer) # add customer name in list
        chairs= int(input("How many chairs are there in the shop (Excluding Barber's Chair) ? ")) #chairsin sho

        if (no_of_customers==0): # if no customers in shop
            print("(: , no customers are in the shop, Barber is still sleeping (Barber not gets disturbed)")
        else: # if there are customers in shop
            print("Settling down customers...")
        for index,cust in enumerate(customer_lst[:no_of_customers]): # creating thread for each customer in sho
            sleeep(1)
            customer_thread=threading.Thread(name="Customer", target=customer, args=[f'{cust}']) #thread of cus
            customer_thread.start() # start thread for customer
            cust_threads.append(customer_thread) # add customer thread in list of customer threads
```

TEST CASES:

1st CASE:

Customers : **4**

waiting chairs = **2**

Condition:

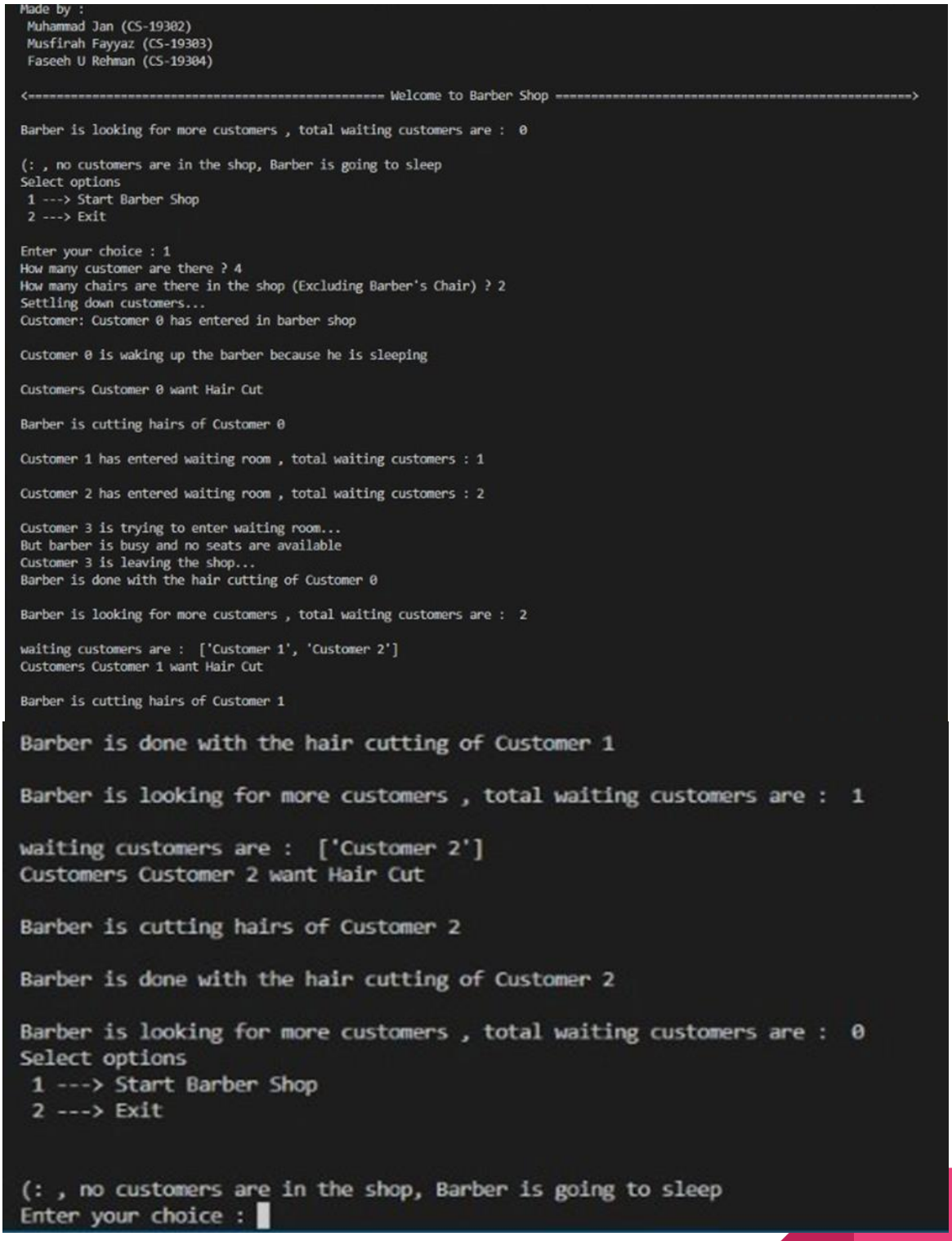
waiting chairs < customers

Expected Output:

3 customers will settle down, 1 customer will wake the barber up and will sit on his chair and 2 customers will wait on waiting chairs. 1 customer will call the Balk method because no seats are available. (He will leaves the shop)

CASE PASSED!

Actual Output:



TEST CASES:

2nd CASE:

Customers : **1**

waiting chairs = **1**

Condition:

waiting chairs = customers

Expected Output:

A customer will enter and will wake the barber up, sits on his chair, get the haircut done, thus no waiting chairs utilized in this case. (Here is 1 waiting chair but it is not utilized.)

CASE PASSED!

Actual Output:

```
 Made by :
 Muhammad Jan (CS-19302)
 Musfirah Fayyaz (CS-19303)
 Faseeh U Rehman (CS-19304)

<========================================== Welcome to Barber Shop ==========================================>

Barber is looking for more customers , total waiting customers are :  0
Select options
 1 ---> Start Barber Shop
 2 ---> Exit


(: , no customers are in the shop, Barber is going to sleep
Enter your choice : 1
How many customer are there ? 1
How many chairs are there in the shop (Excluding Barber's Chair) ? 1
Settling down customers...
Customer: Customer 0 has entered in barber shop

Customer 0 is waking up the barber because he is sleeping

Customers Customer 0 want Hair Cut

Barber is cutting hairs of Customer 0

Barber is done with the hair cutting of Customer 0

Barber is looking for more customers , total waiting customers are :  0

(: , no customers are in the shop, Barber is going to sleep
Select options
 1 ---> Start Barber Shop
 2 ---> Exit

Enter your choice :
```

TEST CASES:

3rd CASE:

Customers : **1**

waiting chairs : **0**

Condition:

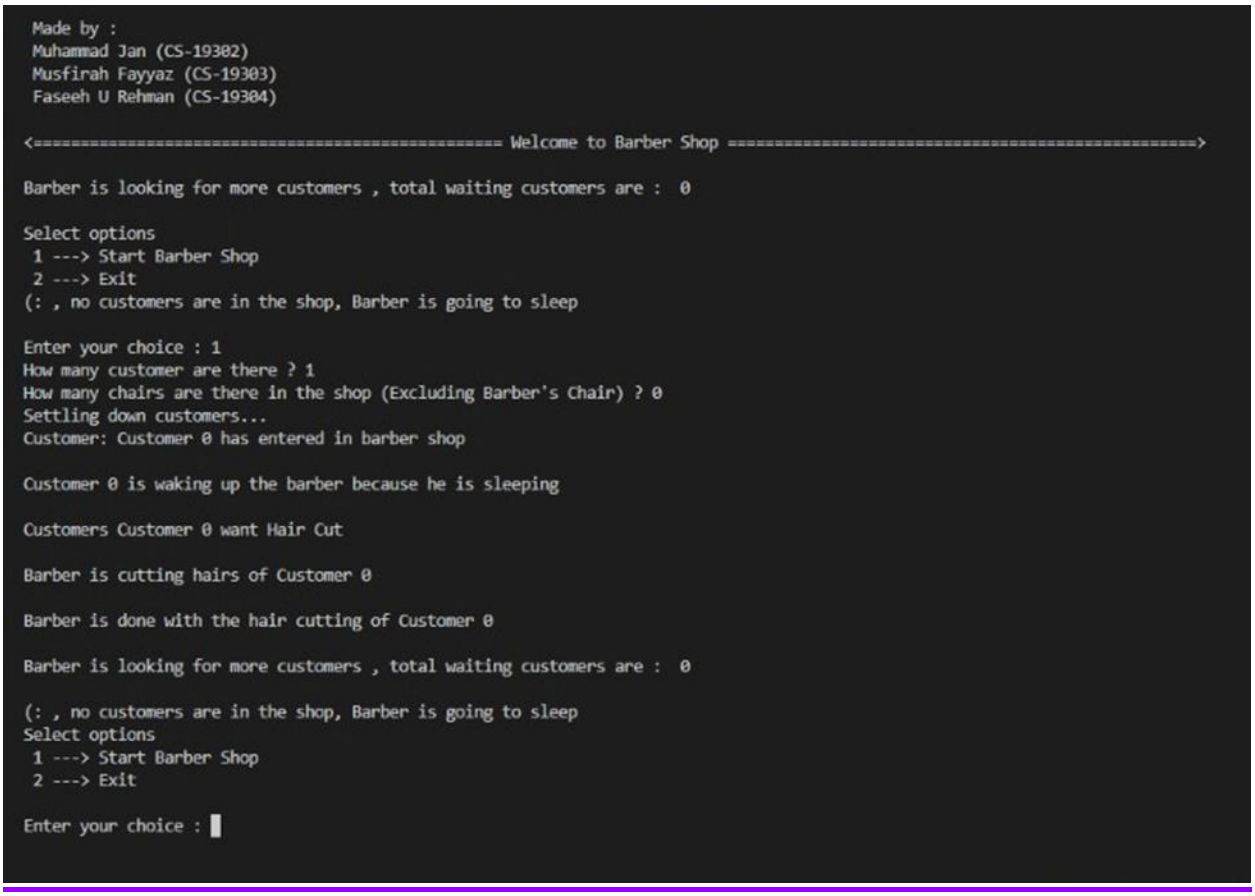
waiting chairs < customers

Expected Output:

A customer will come, wake the barber up, sit on his chair, get the hair cut done. Since, there are no waiting seats therefore no waiting seat is utilized in this case.

CASE PASSED!

Actual Output:



TEST CASES:

4th CASE:

Customers : **3**

waiting chairs = **6**

Condition:

waiting chairs > customers

Expected Output:

Out of 6 waiting chairs only two will be utilized. All the 3 customers will successfully get the services and no one will balk away.

CASE PASSED!

Actual Output:

```
 Made by :
 Muhammad Jan (CS-19302)
 Musfirah Fayyaz (CS-19303)
 Faseeh U Rehman (CS-19304)

<============================================== Welcome to Barber Shop ==============================================>

Barber is looking for more customers , total waiting customers are :  0

Select options
 1 ---> Start Barber Shop
 2 ---> Exit

(: , no customers are in the shop, Barber is going to sleep
Enter your choice : 1
How many customer are there ? 3
How many chairs are there in the shop (Excluding Barber's Chair) ? 6
Settling down customers...
Customer: Customer 0 has entered in barber shop

Customer 0 is waking up the barber because he is sleeping

Customers Customer 0 want Hair Cut

Barber is cutting hairs of Customer 0
Customer 1 has entered waiting room , total waiting customers : 1


Customer 2 has entered waiting room , total waiting customers : 2

Barber is done with the hair cutting of Customer 0

Barber is looking for more customers , total waiting customers are :  2

waiting customers are :  ['Customer 1', 'Customer 2']
Customers Customer 1 want Hair Cut

Barber is cutting hairs of Customer 1
```

```
Barber is done with the hair cutting of Customer 1

Barber is looking for more customers , total waiting customers are :  1

waiting customers are :  ['Customer 2']
Customers Customer 2 want Hair Cut

Barber is cutting hairs of Customer 2

Barber is done with the hair cutting of Customer 2

Barber is looking for more customers , total waiting customers are :  0

(: , no customers are in the shop, Barber is going to sleep
Select options
 1 ---> Start Barber Shop
 2 ---> Exit

Enter your choice :
```

5th CASE:

Customers : **0**

waiting chairs : **3**

Condition:

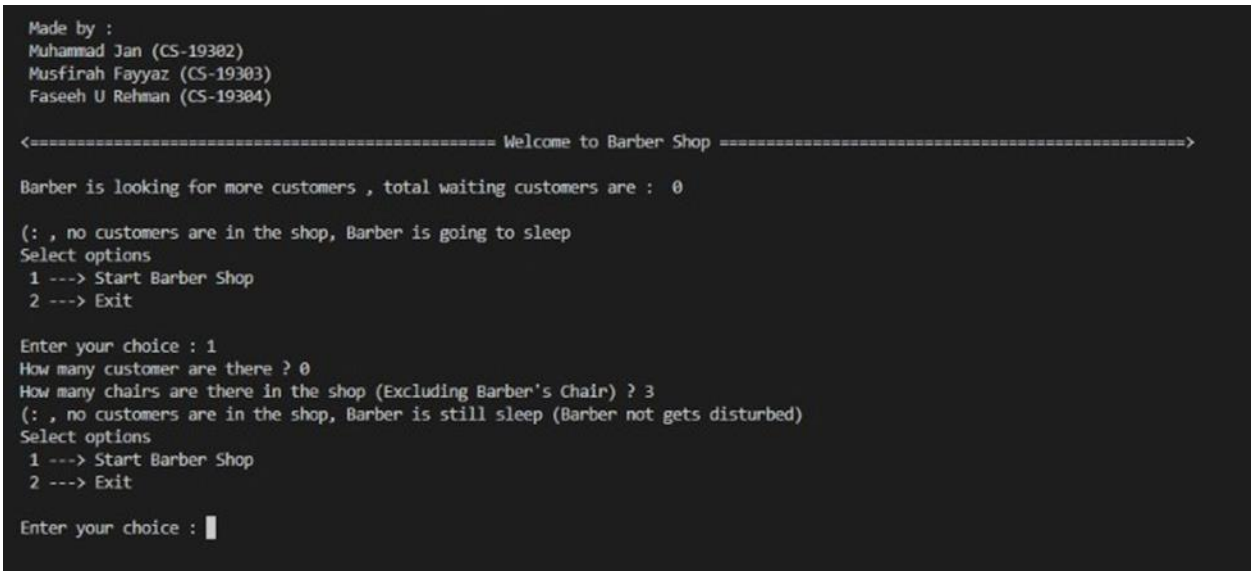
waiting chairs > customers

Expected Output:

Since, no customer enters the shop therefore the barber will remain in the sleep mode and won't get disturbed by any customer.

CASE PASSED!

Actual Output:



Conclusion:

In this kind of problem, it’s a good way to observe User Level Threads (ULTs) on a system. It helps us to observe deadlocks. About, how to overcome these deadlocks and also we learnt how to handle it.