

**Usman Institute of Technology**

**Department of Computer Science Fall 2022**

Name: Muhammad Waleed

Roll no: 20B-115-SE

Course: Operating Systems (CS312)

Course Instructor: Ma’am Shabina Mushtaq

Date: 22-Dec-2022

# Lab Tasks:

1. Write a python program that demonstrates the synchronization of Consumer producer Bounded Buffer Problem using semaphores.

import threading,os

try:

    from rich.console import Console

    from rich.table import Table

except ImportError:

    os.system("pip install rich")

    from rich.console import Console

    from rich.table import Table

console = Console()

table = Table(show\_header=True, header\_style="bold magenta")

buf = []

empty = threading.Semaphore(5)

full = threading.Semaphore(0)

mutex = threading.Lock()

table.add\_column("Name", style="dim", width=12)

table.add\_column("Full", style="dim", width=12)

table.add\_column("Empty", style="dim", width=12)

def producer(name):

    empty.acquire()

    mutex.acquire() # added

    print("Before name: {} Full: {} Empty: {}".format(name,full.\_value,empty.\_value))

    print("Producer is producing")

    mutex.release() # added

    full.release()

    print("After name: {} Full: {} Empty: {}".format(name,full.\_value,empty.\_value))

    table.add\_row(name, str(full.\_value), str(empty.\_value))

def consumer(name):

    full.acquire()

    mutex.acquire() # added

    print("Before name: {} Full: {} Empty: {}".format(name,full.\_value,empty.\_value))

    print("Consumer is consuming")

    mutex.release() # added

    empty.release()

    print("After name: {} Full: {} Empty: {}".format(name,full.\_value,empty.\_value))

    table.add\_row(name, str(full.\_value), str(empty.\_value))

threads=[]

threads.append(threading.Thread(target=consumer,args=("c1",)))

threads.append(threading.Thread(target=producer,args=("p1",)))

threads.append(threading.Thread(target=producer,args=("p2",)))

threads.append(threading.Thread(target=producer,args=("p3",)))

threads.append(threading.Thread(target=consumer,args=("c2",)))

threads.append(threading.Thread(target=producer,args=("p4",)))

threads.append(threading.Thread(target=producer,args=("p5",)))

threads.append(threading.Thread(target=producer,args=("p6",)))

threads.append(threading.Thread(target=producer,args=("p7",)))

for thread in threads:

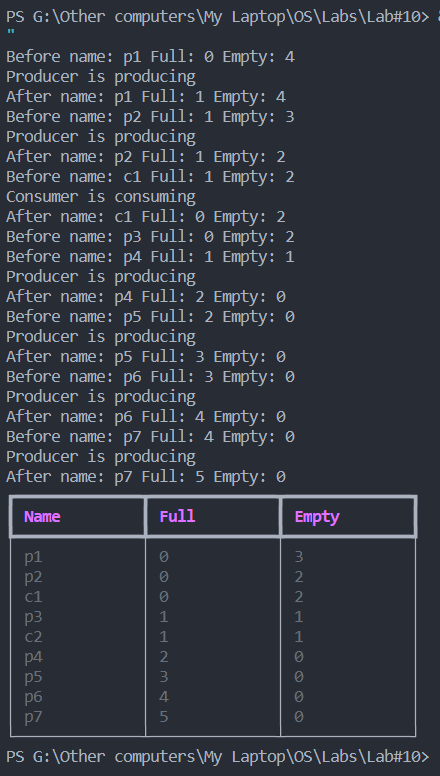
    thread.start()

for thread in threads:

    thread.join()

console.print(table)

Output:



1. Write a python program that demonstrates the synchronization of Readers and Writer Problem using semaphores.

import threading,os,time

readcount = 0

mutex = threading.Lock()

wrt = threading.Lock()

def reader():

    global readcount

    print("Reader arrived")

    mutex.acquire()

    readcount += 1

    if readcount == 1:

        wrt.acquire()

    mutex.release()

    print("Reader is reading")

    mutex.acquire()

    readcount -= 1

    if readcount == 0:

        wrt.release()

    mutex.release()

    time.sleep(2)

def writer():

    print("Writer arrived")

    wrt.acquire()

    print("Writer is writing")

    wrt.release()

    time.sleep(1)

writer = threading.Thread(target=writer)

reader1 = threading.Thread(target=reader)

reader2 = threading.Thread(target=reader)

reader3 = threading.Thread(target=reader)

writer.start()

reader1.start()

reader2.start()

reader3.start()

writer.join()

reader1.join()

reader2.join()

reader3.join()

Output:

Text

Description automatically generated