Batch: C2	Roll No.: 16010121110
Experiment No. 01	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

TITLE: Implementation of Process synchronization algorithms using semaphore - producer consumer problem , reader-writers problem

---

AIM: Implementation of Process synchronization algorithms using semaphore - producer consumer problem, reader-writers problem

---

Expected Outcome of Experiment:

CO 3. To understand the concepts of process synchronization and deadlock.

Books/ Journals/ Websites referred:

1. **Silberschatz A., Galvin P., Gagne G. "Operating Systems Principles", Willey Eight edition.**
2. **Achyut S. Godbole , Atul Kahate "Operating Systems" McGraw Hill Third Edition.**
3. **William Stallings, "Operating System Internal & Design Principles", Pearson.**
4. **Andrew S. Tanenbaum, "Modern Operating System", Prentice Hall.**

Pre Lab/ Prior Concepts:

Knowledge of Concurrency, Mutual Exclusion, Synchronization, Deadlock, Starvation.

Description of the chosen process synchronization algorithm:

The Producer collects the data items one at the time and places the items into a queue for processing later. The Consumer then takes one item at the time from the queue, process it, and saves the data for further analysis. This decoupled design makes easy to parallelize the whole process as desired.

Implementation details:

```
# function to simulate consumer producer

import time

max_len = 10

def produce(n,mutex,pointer):

for i in range(10):

if(pointer.value<max_len):

if (mutex.value==0):

mutex.value = 1 #flop

pointer.value+=1

n[pointer.value]=1

print("I am producing at",pointer.value )
```

```
for i in n:

print(i,end=",")

print()

mutex.value = 0 #flop

else:

print("Producer cannot produce at full array")

time.sleep(0.2)



def consume(n,mutex,pointer):

for i in range(10):

if(pointer.value>=0):

if (mutex.value==0):

mutex.value = 1 #flip

n[pointer.value]=0
```

```
print("I am consuming at" ,pointer.value)

for i in n:

print(i,end=",")

print()

pointer.value-=1



mutex.value = 0 #flip

else:

print("Consumer cannot consume from empty array")

time.sleep(0.5)



if __name__ == '__main__': # dont remove this line
code wont run in windows.

import multiprocessing
```

```
# Define a shared variable

n = multiprocessing.Array('i', max_len)

mutex = multiprocessing.Value('i',0)

pointer = multiprocessing.Value('i',-1)




# Create two processes that run the increment function

process1 =
multiprocessing.Process(target=produce,args=(n,mutex,p
ointer,))

process2 =
multiprocessing.Process(target=consume,args=(n,mutex,p
ointer,))



# Start both processes

process2.start()
```

```
process1.start()

process2.join()

process1.join()
```

```
I am producing at 4
1,1,1,1,1,0,0,0,0,0,
I am producing at 5
1,1,1,1,1,1,0,0,0,0,
I am consuming at 5
1,1,1,1,1,0,0,0,0,0,
I am producing at 5
1,1,1,1,1,1,0,0,0,0,
I am producing at 6
1,1,1,1,1,1,1,0,0,0,
I am consuming at 6
1,1,1,1,1,1,0,0,0,0,
I am consuming at 5
1,1,1,1,1,0,0,0,0,0,
I am consuming at 4
1,1,1,1,0,0,0,0,0,0,
I am consuming at 3
1,1,1,0,0,0,0,0,0,0,
I am consuming at 2
1,1,0,0,0,0,0,0,0,0,
I am consuming at 1
1,0,0,0,0,0,0,0,0,0,
```

READER WRITER

```
# function to simulate reader writer

import time

max_len = 10

def writer(n,mutex,pointer):

for i in range(10):

if(pointer.value<max_len):

if (mutex.value==0):

mutex.value = 1 #flop

pointer.value+=1

n[pointer.value]=1

print("I am writing at",pointer.value )

for i in n:

print(i,end=",")
```

```
print()

mutex.value = 0 #flop



time.sleep(0.2)



def reader(n,mutex,readerno):

for i in range(10):

if (mutex.value==0):

mutex.value = 1 #flip

print("Reader no ", readerno,"reads ",end="")

for i in n:

print(i,end=",")

print()
```

```
mutex.value = 0 #flip

time.sleep(0.3+readerno*0.2)



if __name__ == '__main__': # dont remove this line
code wont run in windows.

import multiprocessing



# Define a shared variable

n = multiprocessing.Array('i', max_len)

mutex = multiprocessing.Value('i',0)

pointer = multiprocessing.Value('i',-1)




# Create two processes that run the increment function
```

```
process1 =
multiprocessing.Process(target=writer,args=(n,mutex,po
inter,))

process2 =
multiprocessing.Process(target=reader,args=(n,mutex,1,
))

process3 =
multiprocessing.Process(target=reader,args=(n,mutex,2,
))



# Start both processes

process1.start()

process2.start()

process3.start()



process1.join()

process2.join()
```

```
process3.join()
```

K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

```
Reader no  2 reads 0,0,0,0,0,0,0,0,0,0,
I am writing at 0
1,0,0,0,0,0,0,0,0,0,
I am writing at 1
1,1,0,0,0,0,0,0,0,0,
Reader no  1 reads 1,1,0,0,0,0,0,0,0,0,
I am writing at 2
1,1,1,0,0,0,0,0,0,0,
Reader no  2 reads 1,1,1,0,0,0,0,0,0,0,
I am writing at 3
1,1,1,1,0,0,0,0,0,0,
Reader no  1 reads 1,1,1,1,0,0,0,0,0,0,
I am writing at 4
1,1,1,1,1,0,0,0,0,0,
I am writing at 5
1,1,1,1,1,1,0,0,0,0,
Reader no  2 reads 1,1,1,1,1,1,0,0,0,0,
I am writing at 6
1,1,1,1,1,1,1,0,0,0,
Reader no  1 reads 1,1,1,1,1,1,1,0,0,0,
I am writing at 7
1,1,1,1,1,1,1,1,0,0,
I am writing at 8
1,1,1,1,1,1,1,1,1,0,
Reader no  1 reads 1,1,1,1,1,1,1,1,1,0,
Reader no  2 reads 1,1,1,1,1,1,1,1,1,0,
Reader no  1 reads 1,1,1,1,1,1,1,1,1,0,
Reader no  2 reads 1,1,1,1,1,1,1,1,1,0,
Reader no  1 reads 1,1,1,1,1,1,1,1,1,0,
```

Conclusion: Thus we have implemented the processor consumer problem in Python. We also have implemented the reader-writer problem. These strategies are used to enforce process synchronization.

Post Lab Objective Questions

1) A semaphore is a shared integer variable
   a) That can't drop below zero
   b) That can't be more than 1
   c) That can't drop below one

Ans: a)

2) Mutual exclusion can be provided by the
   a) Mute locks
   b) Binary semaphores
   c) Both a and b
   d) None of these

Ans: c)

3) A monitor is a module that encapsulates
   a) Shared data structures
   b) Procedures that operate on shared data structure
   c) Synchronization between concurrent procedure invocation
   d) All of the above

Ans: d)

4) To enable a process to wait within the monitor
   a) A condition variable must be declared as condition
   b) Condition Variables must be used as Boolean objects
   c) Semaphore must be used
   d) All of the above

Ans: d)

Date: 30 Oct 2023 **Signature of faculty in-charge**