Experiment : 3
Date : 16/04/2021
Author : Bonnie Simon

# Readers Writers Problem

## AIM

Write a program to implement the first readers-writers problem in python.

## THEORY

The readers-writers problem relates to an object such as a file that is shared between multiple processes. Some of these processes are readers i.e. they only want to read the data from the object and some of the processes are writers i.e. they want to write into the object.

The readers-writers problem is used to manage synchronization so that there are no problems with the object data. For example - If two readers access the object at the same time there is no problem. However if two writers or a reader and writer access the object at the same time, there may be problems.

To solve this situation, a writer should get exclusive access to an object i.e. when a writer is accessing the object, no reader or writer may access it. However, multiple readers can access the object at the same time.

This can be implemented using semaphores.

## ALGORITHM

1. Start
2. Select random number between 0 & 100
3. If random number > 50
    a. Execute Procedure Reader
4. Else if random number < 50
    a. Execute Procedure Writer
5. Stop

1. Start Procedure Reader
2. Perform Reading
3. Acquire lock
4. Access Shared data
5. Release lock
6. Stop

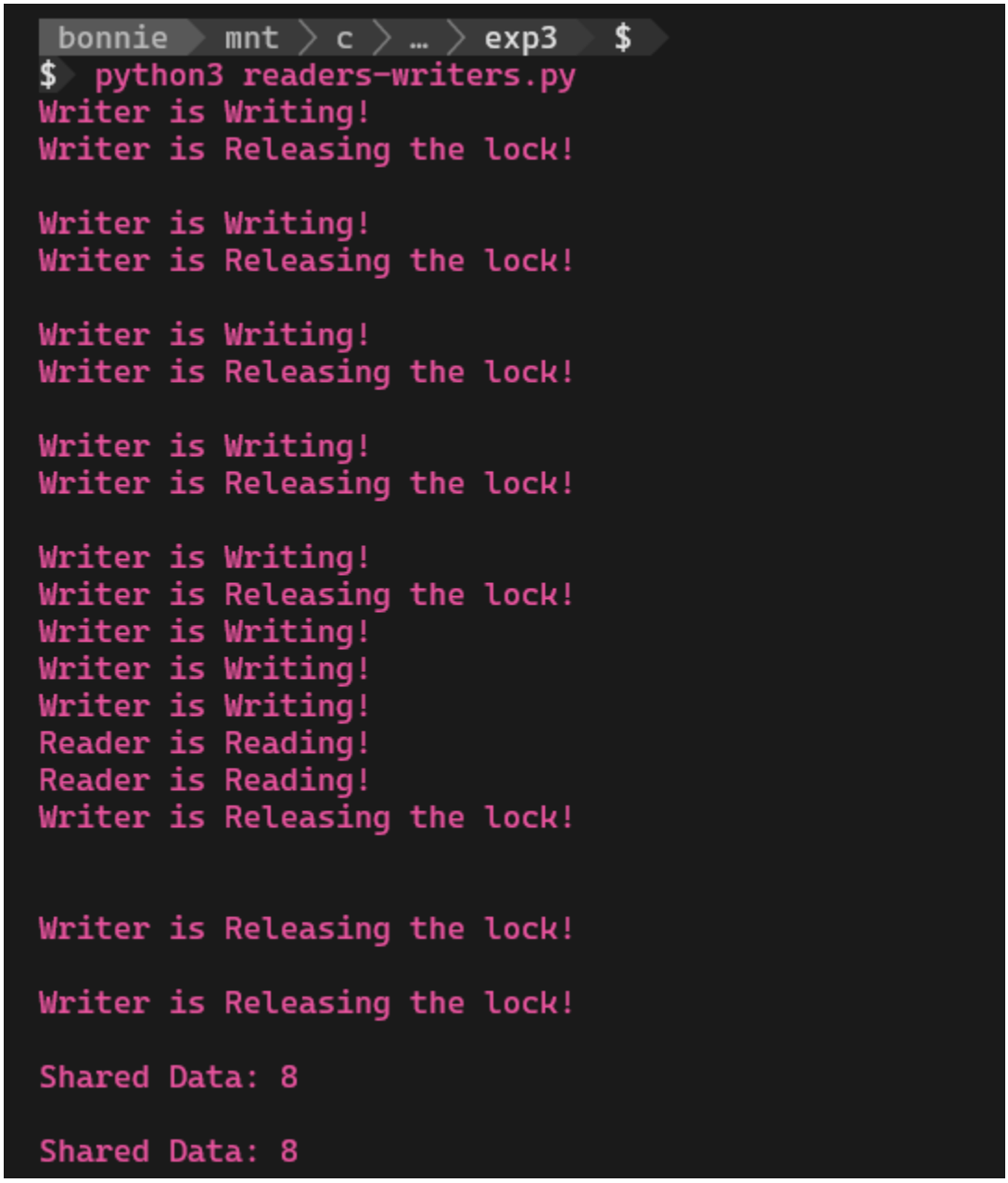
1. Start Procedure Writer
2. Perform Writing
3. Acquire Lock
4. Increment Writer count
5. Release Lock
6. Stop

# PROGRAM

```python
import threading as thread
import random

global x
x = 0
lock = thread.Lock()    #Lock for synchronising access

def Reader():
    global x
    print('Reader is Reading!')
    lock.acquire()      #Acquire the lock before Reading (mutex approach)
    print('Shared Data:', x)
    lock.release()      #Release the lock after Reading
    print()

def Writer():
    global x
    print('Writer is Writing!')
    lock.acquire()      #Acquire the lock before Writing
    x += 1              #Write on the shared memory
    print('Writer is Releasing the lock!')
    lock.release()      #Release the lock after Writing
    print()

if __name__ == '__main__':
    for i in range(0, 10):
        randomNumber = random.randint(0, 100)   #Generate a Random number between 0 to 100
        if(randomNumber > 50):
            Thread1 = thread.Thread(target = Reader)
            Thread1.start()
        else:
            Thread2 = thread.Thread(target = Writer)
            Thread2.start()

Thread1.join()
Thread2.join()
```

module threading
Thread module emulating a subset of Java's threading model.

## OUTPUT




## RESULT

The python program to implement readers writers program has been successfully executed.