Experiment – 8

Reader - Writers Problem

The reader - writers problem is a process synchronization problem which relates to a data or a file that is shared b/w more than one processes at a time. Among these various process, some are readers - which can only read the data, some are writers - that can both read and write the data.

The reader - written problem is used for managing synchronization b/w these readers and writers such that no inconcistency is genrated

To do so, a critical section is marked inside the code which can be accessed by only one of the two types of process - readers or writers. This is accomplished using semaphores. Semaphore is a simple variable that can be shared between threads. Semaphores are of two types, and any one of them can be used to solve the reader - writers problem

1. Binary Semaphore
2. Counting Semaphore

Using the value of semaphores, two functions are called throughout the code to manage the critical section control -

1. wait() - decrements semaphore value
2. signal() - increments semaphore value

ALGORITHM

1. If the semaphore value is 0, both reader and writer can enter the critical section (not together)
2. If reader enters the critical section the value of semaphore is increased by 1
3. If writer enters the critical section, the value of semaphore is decreased by 1
4. If the value of semaphore is positive then writer cannot enter the critical section
5. If the value of semaphore is negative then reader cannot enter the critical section
6. When reader exits the critical section the value of semaphore is decreased by 1
7. When writer exits the critical section the value of semaphore is increased by 1.

Code

```
import time
import threading
from datetime import datetime


def timestamp(data, typ, process, i="/"):
    typ = "Reader " if typ == "r" else "Writter"
    print(f'{datetime.now().
strftime("%X")} : {typ} {process} : Iteration {i} : {data}')


class Problem:
    def __init__(
        self,
        readers,
        readerStart,
        readerWait,
        readerEnd,
        readerRepeat,
        writters,
        writerData,
        writterStart,
        writterWait,
        wriiterEnd,
        writterRepeat,
    ):
        """
        Reader Writter Problem

        readers         - int   - number of readers
        readerStart     - list  - waiting time before reader starts
        readerWait      - list  - waiting time before reading data
        readerEnd       - list  - waiting time after reading data
        readerRepeat    - list  - number of time reader reades data

        writters        - int   - number of writters
        writerData      - list  - data to be written by writter
        writterStart    - list  - waiting time before writter starts writting
        writterWait     - list  - waiting time before writting
        wriiterEnd      - list  - waiting time after writting
        writterRepeat   - list  - number of times writter writes

        val             - list  - value to be written/read
```

```
        rdCount         - list  - number of writters currently in critical sectio
n
        """

        self.readers = readers
        self.readerStart = readerStart
        self.readerWait = readerWait
        self.readerEnd = readerEnd
        self.readerRepeat = readerRepeat

        self.writters = writters
        self.writerData = writerData
        self.writterStart = writterStart
        self.writterWait = writterWait
        self.wriiterEnd = wriiterEnd
        self.writterRepeat = writterRepeat

        self.val = -1
        self.rdCount = 0
        self.start()

    def start(self):
        # Writters
        for i in range(self.writters):
            threading.Thread(
                target=self.writter,
                args=(
                    i + 1,
                    self.writterStart[i],
                    self.writterWait[i],
                    self.wriiterEnd[i],
                    self.writterRepeat[i],
                    self.writerData[i],
                ),
            ).start()

        # Readers
        for i in range(self.readers):
```

```
            threading . Thread(
                target=self . reader,
                args=(
                    i + 1,
                    self . readerStart[i],
                    self . readerStart[i],
                    self . readerEnd[i],
                    self . readerRepeat[i],
                ),
            ) . start()

    def wait(self):
        time . sleep(0 . 001)

    def reader(self, idx, start, wait, end, repeat):
        time . sleep(start)
        timestamp("started", "r", idx)
        for i in range(repeat):
            waitOne = 0
            while self . rdCount < 0:
                if waitOne:
                    waitOne = 1
                    timestamp(f"waiting", "r", idx, i)
                self . wait()

            self . rdCount += 1
            time . sleep(wait)
            timestamp(f"< {self . val} >", "r", idx, i)
            self . rdCount -= 1

            time . sleep(end)

    def writter(self, idx, start, wait, end, repeat, data):
        time . sleep(start)
        timestamp("started", "w", idx)
        for i in range(repeat):
            waitOne = 0
            while self . rdCount > 0:
                if waitOne:
                    waitOne = 1
```

```
                    timestamp(f"waiting", "w", idx, i)
                self.wait()

            self.rdCount -= 1
            time.sleep(wait)
            self.val = data
            timestamp(f"< {self.val} >", "w", idx, i)
            self.rdCount += 1
            time.sleep(end)


def linput(data):
    return list(map(int, input(data).split()))
def iinput(data):
    return int(input(data))
readers = iinput("Humber of readers : ")
readerStart = linput("Reader start time : ")
readerWait = linput("Reader Wait time : ")
readerEnd = linput("Reader End time : ")
readerRepeat = linput("Reader Repeats : ")
writters = iinput("Humber of writters : ")
writerData = linput("Writter data : ")
writterStart = linput("Writter start time : ")
writterWait = linput("Writter wait time : ")
wriiterEnd = linput("Writter end time : ")
writterRepeat = linput("Writter repeats : ")

print("Deekshant Wadhwa\n01296303118\n")

problem = Problem(
    readers,
    readerStart,
    readerWait,
    readerEnd,
    readerRepeat,
    writters,
    writerData,
    writterStart,
    writterWait,
    wriiterEnd,
    writterRepeat,
)
```

Output

```
PS D:\Drive\Sem 6\OS\lab> python -u "d:\Drive\Sem 6\OS\lab\readerWritter

Number of readers : 4
Reader start time : 1 2 3 4
Reader Wait time : 2 1 3 1
Reader End time : 1 2 3 1
Reader Repeats : 3 2 4 1
Number of writters : 2
Writter data : 2 3
Writter start time : 0 0
Writter wait time : 1 0
Writter end time : 2 1
Writter repeats : 5 4

Deekshant Wadhwa
01296303118

12:52:33 : Writter 1 : Iteration / : started
12:52:33 : Writter 2 : Iteration / : started
12:52:33 : Writter 2 : Iteration 0 : < 3 >
12:52:34 : Reader  1 : Iteration / : started
12:52:34 : Writter 2 : Iteration 1 : < 3 >
12:52:34 : Writter 1 : Iteration 0 : < 2 >
12:52:35 : Reader  2 : Iteration / : started
12:52:35 : Reader  1 : Iteration 0 : < 2 >
12:52:36 : Reader  3 : Iteration / : started
12:52:37 : Reader  4 : Iteration / : started
12:52:37 : Reader  2 : Iteration 0 : < 2 >
12:52:37 : Reader  1 : Iteration 1 : < 2 >
12:52:39 : Reader  3 : Iteration 0 : < 2 >
12:52:39 : Reader  1 : Iteration 2 : < 2 >
12:52:41 : Reader  4 : Iteration 0 : < 2 >
12:52:41 : Reader  2 : Iteration 1 : < 2 >
12:52:41 : Writter 2 : Iteration 2 : < 3 >
12:52:42 : Writter 1 : Iteration 1 : < 2 >
12:52:42 : Writter 2 : Iteration 3 : < 3 >
12:52:45 : Reader  3 : Iteration 1 : < 3 >
12:52:46 : Writter 1 : Iteration 2 : < 2 >
12:52:51 : Reader  3 : Iteration 2 : < 2 >
12:52:52 : Writter 1 : Iteration 3 : < 2 >
12:52:57 : Reader  3 : Iteration 3 : < 2 >
12:52:58 : Writter 1 : Iteration 4 : < 2 >
```