**PRODUCER CONSUMER PROBLEM**

**Paras Raj Pahari**

**PROBLEM DEFINITION:**

Producer Consumer problem is one of the most common problem to demonstrate the concurrent programming. It consists of fix sized buffer and two thread classes – producer and consumer.

Producer job is to place the data in the buffer location of size 10, while consumer job is to consume the data from the respective buffer location where producer placed it.

Producer depends on consumer to empty the buffer location, so it can place the data in the buffer whereas, consumer depends on producer to place the data, so it can consume the data from the respective buffer location.

1. **Solution:**

Producer should check if buffer is full or not. If it is full, it should release the thread and wait for some random time (in between 10 to 100 milliseconds) for consumer to consume the data. Once the consumer make the space available in the buffer location, it notifies the producer that the buffer location is available for new data to be produced. Then producer can wake up and acquire the thread, so it can start placing the data in the buffer space available.

Whereas, Consumer should check if buffer is empty or not. If there’s no data available in buffer location for consuming, then consumer should release the thread and waits random time (in between for 10 to 100 milliseconds) for producer to place that it in the respective location. Once the producer places the data in the buffer, producer notifies the consumer so that consumer can wake up and consume the data.

The main thing is to formulate a way for communication between two threads – Producer and Consumer, so that they can communicate each other of their status while producing and consuming from the same buffer.

1. **IMPLEMENTATION DETAILS:**

Programming Language: Python 2.7

IDE USED: PYCHARM

OS Environment: LINUX/WINDOWS

It can as well be executed from the terminal in Linux or command line from Windows.

**$** python pc.py

1. **Explanation:**

Shared Buffer Size: 10

Shared Buffer Data Structure: Dictionary with Key and Value pair, where key represent the index and value represent the data in that index

Random Wait Time for Producer and Consumer Thread: 10 to 100 milliseconds

Program Executes for 15 seconds

**Producer Thread Flow**

* Acquires the thread
* Checks whether buffer has reached the maximum size
* If Buffer space is available
* Produces the number item (0-14) in sequential manner in the respective sequential buffer location
* Update the Console and file ‘producer.txt’ with the log
* Notifies and Releases the thread and wait for some random time in between 10 to 100 milliseconds

Else

* Notifies the Thread
* Releases the Thread
* Waits for consumer for some random times in between 10 to 100 milliseconds
* Once the buffer has reached the maximum size, initialize the buffer again to the initial location, so it can start producing from the beginning. Here also, it notifies and releases the thread for consumer.

**Consumer Thread Flow**

* Acquires the thread
* Checks whether it has reached the end location of buffer
* Checks if the buffer location has data produced or not
* If buffer location has data available, it consumes the data, updates the console and ‘consumer.txt’ with log; else it notifies and releases the thread, wait for some random time in between 10 to 100 milliseconds for producer to produce the data in the buffer location
* Once the buffer has reached the end of the buffer location, it loops to the beginning of the buffer again to consume from the beginning, and increases the count of buffer loop. Once it performs the operation, it notifies and releases the thread, and sleeps for some random time in between 10 to 100 milliseconds

1. **Result:**

Result is obtain as desired.

Both producer and consumer produced and consumed data randomly from the shared buffer. Random time in between two consecutive threads execution was not greater than 100 milliseconds. And whole program executed for only 15 seconds and terminated successfully. Producer produced the sequential data in the respective sequential buffer location in random manner, whereas, consumer consumed respective sequential data produced by producer from the respective sequential buffer location in random manner.

Logs are updated respectively in the ‘producer.txt’ and ‘consumer.txt’ file.

1. **References:**
2. High Level Threading Interface Python Module: <https://docs.python.org/2/library/threading.html>
3. Random Python Module : <https://docs.python.org/2/library/random.html>
4. Time access and conversions in python: <https://docs.python.org/2/library/time.html>

1. Data Structure in Python: <https://docs.python.org/3/tutorial/datastructures.html>
2. Condition Objects in Python: <https://docs.python.org/2.4/lib/condition-objects.html>
3. Python Multi-Threading Tutorial: <https://www.tutorialspoint.com/python/python_multithreading.htm>