Operating Systems Lab
Fall 2024-25(L59+60)
Student Name:- Parth Suri
Registration No:- 22BDS0116
Class No. :- VL2024250102445
Faculty Name:- Rahul Srivastava

Experiment Title: Implementation of Peterson's Solution for Process Synchronization in Python

Objective: To understand and implement Peterson's solution for process synchronization using Python, ensuring mutual exclusion and avoiding race conditions in a multi-process environment.

Theory:

Peterson’s Solution is a classic algorithm used for achieving mutual exclusion in concurrent programming. It provides a solution to the critical section problem for two processes. The algorithm ensures that no two processes can enter the critical section simultaneously. Peterson’s solution satisfies the three essential conditions for a proper synchronization mechanism:

1. Mutual Exclusion: Only one process can execute in its critical section at any given time.
2. Progress: If no process is in the critical section, one of the waiting processes must be allowed to enter the critical section.
3. Bounded Waiting: There is a limit on the number of times that other processes can enter the critical section after a process has made a request to enter.

Key components of Peterson's Solution:

- Flags (boolean array): This is used by the processes to indicate if they want to enter the critical section.
- Turn (integer variable): This determines which process should be allowed to enter the critical section. Algorithm for two processes:
- flag[0] and flag[1] are used to indicate whether Process 0 and Process 1 want to enter the critical section, respectively.
- turn is a shared variable that indicates whose turn it is to enter the critical section.

Each process executes the following steps:

1. Entry Section:

o The process sets its flag to True, indicating its desire to enter the critical section.

o The process then sets the turn variable to the other process, indicating it is giving the other process a chance to enter the critical section.

o The process waits until the other process is not interested (flag[j] == False) or it is its turn (turn == i).

2. Critical Section:

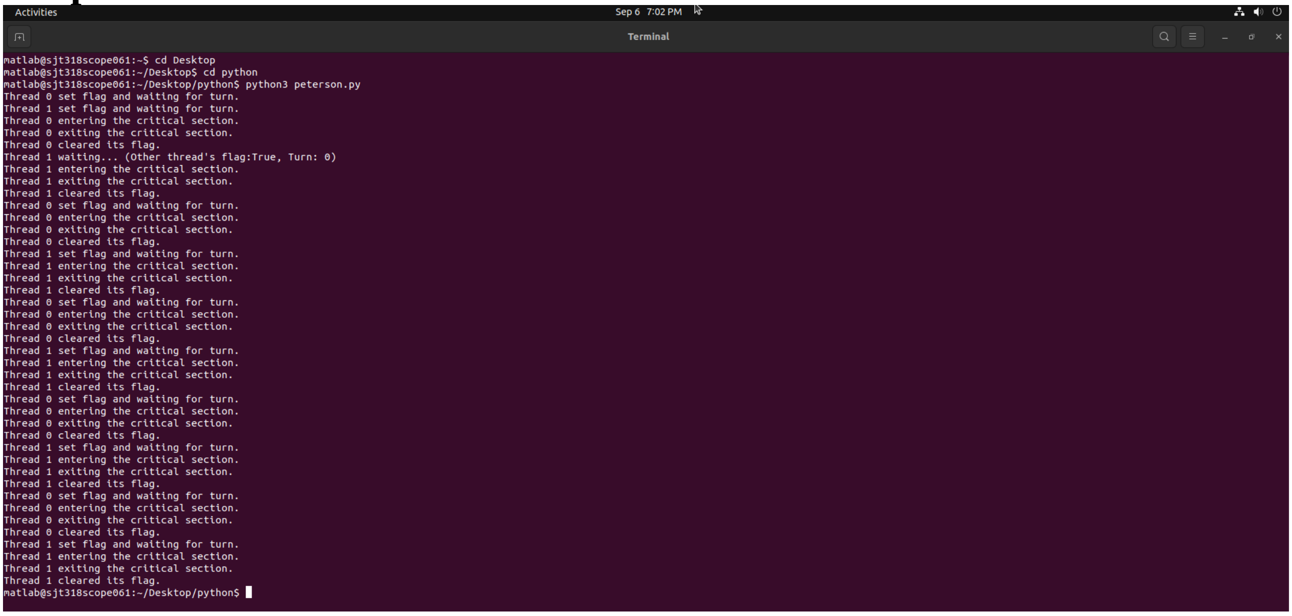
o The process enters the critical section and performs its operations.

3. Exit Section:
o The process sets its flag to False, indicating that it has exited the critical section.

Code:-

```
import threading
import time
flag = [False, False]
turn = 0
def peterson_algorithm(thread_id):
    global flag, turn
    other_thread_id = 1 - thread_id
    for _ in range(5):
        flag[thread_id] = True
        turn = other_thread_id
        print(f"Thread {thread_id} set flag and waiting for turn.")
        while flag[other_thread_id] and turn == other_thread_id:
            print(f"Thread {thread_id} waiting... (Other thread's flag: {flag[other_thread_id]}, Turn: {turn})")
            time.sleep(0.5)
        print(f"Thread {thread_id} entering the critical section.")
        print(f"Thread {thread_id} exiting the critical section.")
        flag[thread_id] = False
        print(f"Thread {thread_id} cleared its flag.")
        time.sleep(1)
thread_0 = threading.Thread(target=peterson_algorithm, args=(0,))
thread_1 = threading.Thread(target=peterson_algorithm, args=(1,))
thread_0.start()
thread_1.start()
thread_0.join()
thread_1.join()
```

Output:-



```
matlab@sjt318scope061:~$ cd Desktop
matlab@sjt318scope061:~/Desktop$ cd python
matlab@sjt318scope061:~/Desktop/python$ python3 peterson.py
Thread 0 set flag and waiting for turn.
Thread 1 set flag and waiting for turn.
Thread 0 entering the critical section.
Thread 0 exiting the critical section.
Thread 0 cleared its flag.
Thread 1 waiting... (Other thread's flag:True, Turn: 0)
Thread 1 entering the critical section.
Thread 1 exiting the critical section.
Thread 1 cleared its flag.
Thread 0 set flag and waiting for turn.
Thread 0 entering the critical section.
Thread 0 exiting the critical section.
Thread 0 cleared its flag.
Thread 1 set flag and waiting for turn.
Thread 1 entering the critical section.
Thread 1 exiting the critical section.
Thread 1 cleared its flag.
Thread 0 set flag and waiting for turn.
Thread 0 entering the critical section.
Thread 0 exiting the critical section.
Thread 0 cleared its flag.
Thread 1 set flag and waiting for turn.
Thread 1 entering the critical section.
Thread 1 exiting the critical section.
Thread 1 cleared its flag.
Thread 0 set flag and waiting for turn.
Thread 0 entering the critical section.
Thread 0 exiting the critical section.
Thread 0 cleared its flag.
Thread 1 set flag and waiting for turn.
Thread 1 entering the critical section.
Thread 1 exiting the critical section.
Thread 1 cleared its flag.
Thread 0 set flag and waiting for turn.
Thread 0 entering the critical section.
Thread 0 exiting the critical section.
Thread 0 cleared its flag.
Thread 1 set flag and waiting for turn.
Thread 1 entering the critical section.
Thread 1 exiting the critical section.
Thread 1 cleared its flag.
matlab@sjt318scope061:~/Desktop/python$ █
```