Assignment - 7

Aim : Thread synchronisation using lock mechanism

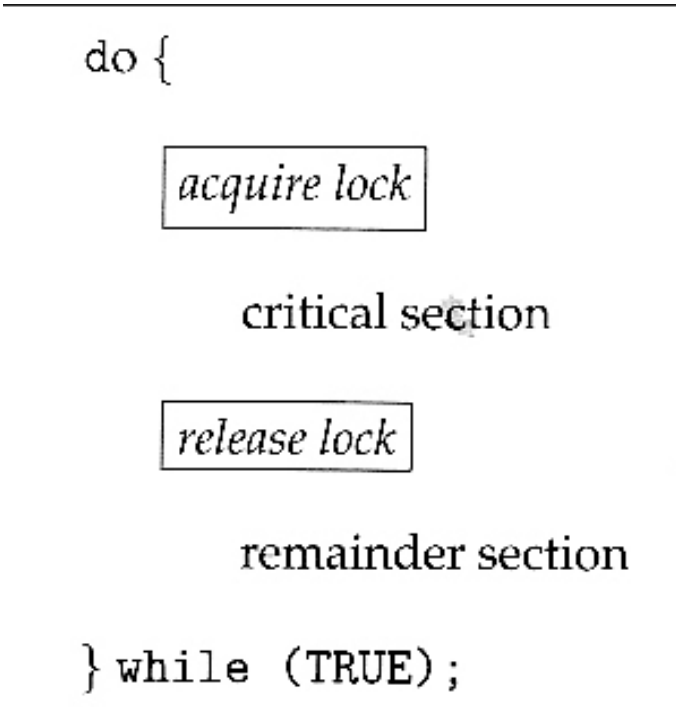
Theory :

Lock Variable :
In an operating system (OS), a lock is a synchronization method that prevents multiple threads from accessing a resource at the same time. Locks can be advisory or mandatory. Advisory locks require each thread to cooperate in gaining and releasing locks. Mandatory locks prevent other threads from accessing a resource, and issue an exception if this occurs.

Locks have two operations:

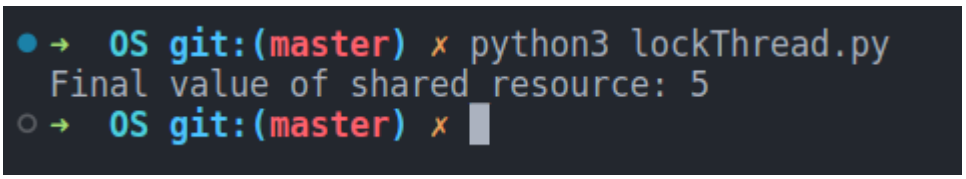
- Acquire: Allows a thread to take ownership of a lock
- Release: Relinquishes ownership of the lock, allowing another thread to take ownership of it

A lock variable is a software mechanism that provides a simple synchronization mechanism for processes. It's implemented in user mode, so it doesn't require support from the operating system. A lock variable can have two values: 0 or 1. A value of 0 means that the critical section is vacant, while a value of 1 means that it's occupied.



Program & Output :

```
import threading

shared_resource = 5

lock = threading.Lock()

def increment():
   global shared_resource
   for _ in range(1000):
       lock.acquire()
       try:
           # critical section
           shared_resource = shared_resource + 1
       finally:
           lock.release()

def decrement():
   global shared_resource
   for _ in range(1000):
       lock.acquire()
       try:
           # critical section
           shared_resource = shared_resource - 1
       finally:
           lock.release()

thread1 = threading.Thread(target=increment)
thread2 = threading.Thread(target=decrement)

thread1.start()
thread2.start()

thread1.join()
thread2.join()


print("Final value of shared_resource:", shared_resource)
```



Conclusion : Here in this experiment, we studied lock variable mechanism and implemented it in python using threading.