```

def results(n):
    print("The Factorial of",n,"is:",factorial(n))

t1=Thread(target=results,args=(5,))
t2=Thread(target=results,args=(9,))
t1.start()
t2.start()
```

Output:

```
The Factorial of 5 is: 120
The Factorial of 9 is: 362880
```

In the above program instead of RLock if we use normal Lock then the thread will be blocked.

Difference between Lock and RLock:

table

Lock:

1. Lock object can be acquired by only one thread at a time.Even owner thread also cannot acquire multiple times.
2. Not suitable to execute recursive functions and nested access calls
3. In this case Lock object will takes care only Locked or unlocked and it never takes care about owner thread and recursion level.

RLock:

1. RLock object can be acquired by only one thread at a time, but owner thread can acquire same lock object multiple times.
2. Best suitable to execute recursive functions and nested access calls
3. In this case RLock object will takes care whether Locked or unlocked and owner thread information, recursiion level.

Synchronization by using Semaphore:

In the case of Lock and RLock,at a time only one thread is allowed to execute.

Sometimes our requirement is at a time a particular number of threads are allowed to access(like at a time 10 memebers are allowed to access database server,4 members are allowed to access Network connection etc).To handle this requirement we cannot use Lock and RLock concepts and we should go for Semaphore concept.

Semaphore can be used to limit the access to the shared resources with limited capacity.

Semaphore is advanced Synchronization Mechanism.