Page No.:
Date:
YOUVA

Q-21

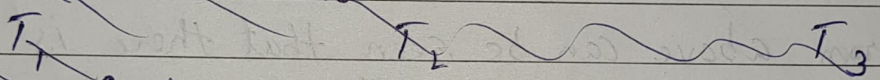
As per code the destination and source vectors for each of 3 threads are.

Thread-1
v_dest → V1
v_src → V2

Thread-2
v_dest → V2
v_src → V3

Thread-3
v_dest → V3
v_src → V1

Define execution as below.



pth_mutex_lock(&v_dest→lock)

Let execution order be like for vector_add() by all threads

① pthread_mutex_lock(&v_dst→lock) // by T_1 so V1 lock
↓ interrupt & swith to T_2

② pthread_mutex_lock(&v_dst→lock) // by T_2 so V2 lock
↓ interrupt & switch to T_3

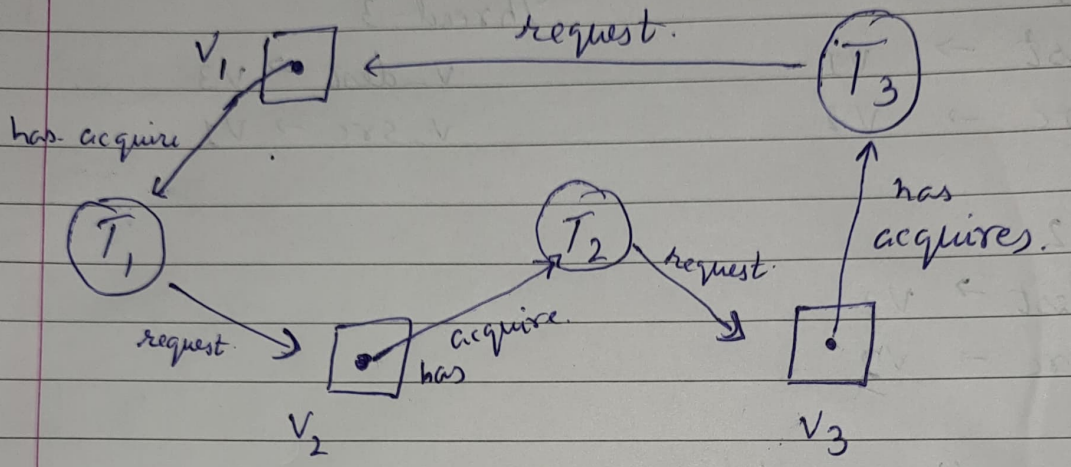
③ pthread_mutex_lock(&v_dst→lock) // by T_3 so V_3 lock

If the above order of execution is done then all V_1, V_2, V_3 as destination are locked by T_1, T_2, T_3 threads.
Now in order to lock any V_1, V_2, V_3 no thread can acquire lock as its already taken by other thread.
So this situation results in deadlock.

The Resource Allocation Graph is defined as.

T_1, T_2, T_3 => threads.

V_1, V_2, V_3 => Resources (with only one instance)



Here from above can be seen that there is the circular wait in resource allocation graph which shows existance of deadlock in the code.