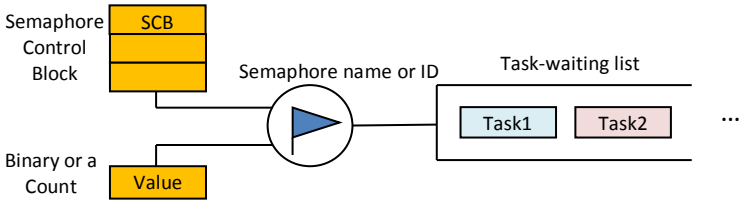
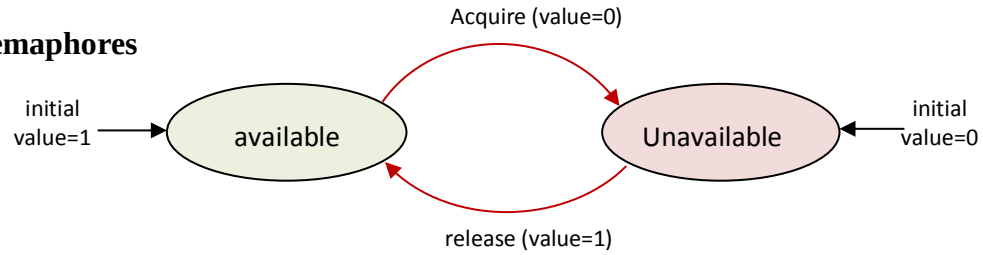
Embedded Information Systems: Addendum to the lecture of 26th of November 2019.

Semaphores

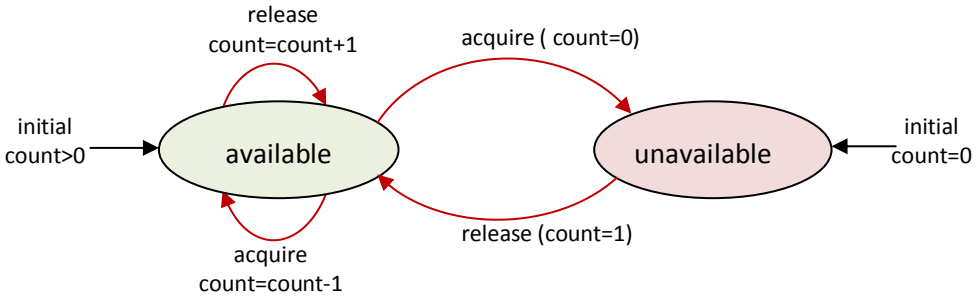


a. Binary semaphores



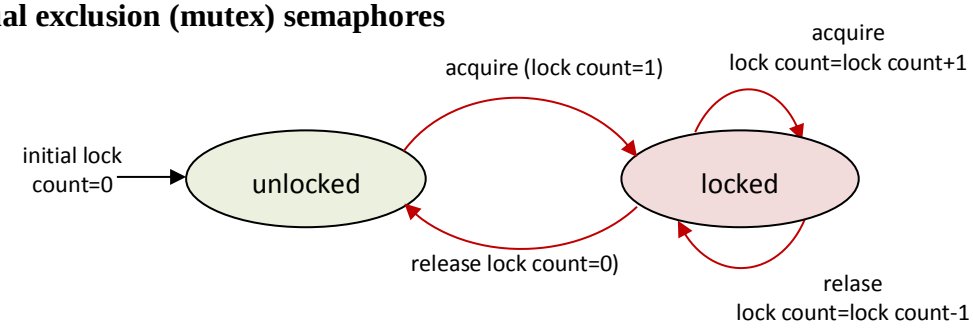
Global resource: allows any task to release it, even if the task did not initially acquire it.
When a binary semaphore is first created, it can be initialized to either available or unavailable.

b. Counting semaphores



Global resource: it can be shared by all tasks that need them. Any task can release a counting semaphore token. Each release operation increments the count by one, even if the task making this call did not acquire a token in the first place.
Some implementations might allow the count to be bounded.

c. Mutual exclusion (mutex) semaphores



Mutex ownership: it is gained when a task first locks the mutex by acquiring it. Conversely, a task loses ownership of the mutex when unlocks it by releasing it. When a task owns the mutex, it is not possible for anz other task to lock or unlock that mutex.

Semaphore operations

Create	binary, counting, mutex
Delete	ID is needed.
Acquire	synonims: take, sm_p, pend, lock
Release	synonims: give, sm_v, post, unlock:

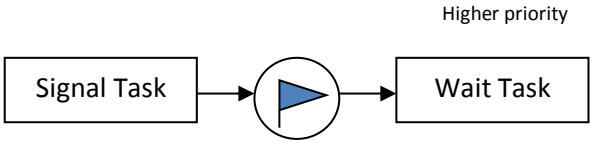
A request to acquire can perform in the following ways:

- wait forever
- wait with a timeout
- do not wait

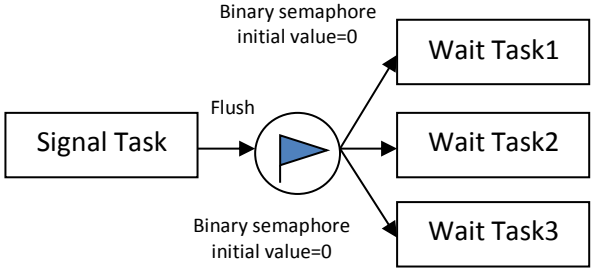
Flush Unblocks all tasks waiting on a semaphore
Show info
Show blocked tasks

Typical semaphore use>

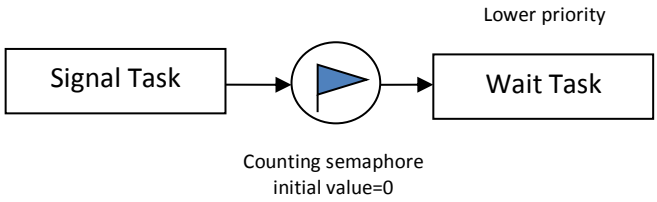
Wait-and-Signal synchronization:



Multiple-Task Wait-and-Signal synchronization:

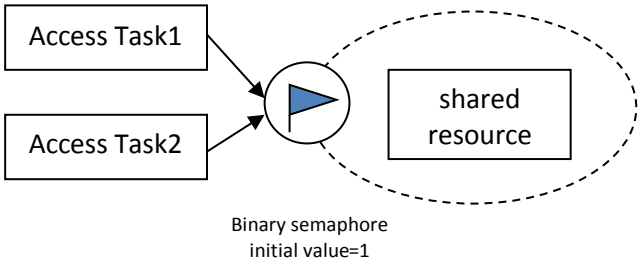


Credit-Tracking synchronization:



Typical example: signal bursts with IT request

Single Shared-Resource-Access synchronization:



Multiple Shared-Resource-Access synchronization: