Slide 1

Look carefully at program in file example3.c. In this program, three tasks are created with a global (shared) integer variable x and a semaphore, created first as a mutex.

Slide 2

Function executed in first task tries every second to take the semaphore. If it succeeds within 500 ms, it puts x equal to 1 and displays a message indicating this operation. Otherwise, the displayed message indicates failure of taking mutex. Note that this is a very simple example of a critical section protected by a mutex. Tasks 2 and 3 are similar, but put x to 2 and 3 respectively.

Slide 3

When first executed with task 3 having higher priority, followed by 2, then 1, the three tasks operate each second according to their priority levels.

Slide 4

Here, one of the tasks does not give the mutex at the end of its critical section. As seen, execution stops with an error as a task that takes a mutex must give it back.

Slide 5

Here we changed the semaphore from mutex to binary. Since the binary semaphore initial value is 0, no task will be able to take the semaphore. Since we specified a time out period, the higher priority task will stop waiting for the semaphore, change the value of x, and gives the semaphore. From this point, operation continues normally as when a mutex was used. This would not occur if timeout period was portMAX_DELAY. Try now to expect what happens if task 3 does not give the semaphore after its critical section.

Slide 6

Execution will not stop as in the case of the mutex. However, tasks will repeatedly time out after waiting to take the semaphore that was not given.