IEEEtran reports template
Mälardalen University - M.Sc.Eng Robotics Reports

Pontus Svensson*,
School of Innovation, Design and Engineering, M.Sc.Eng Robotics
Mälardalens University, Västerås, Sweden
Email: psn19003@student.mdu.se*

I. Introduction

This report covers the newly introduced problems that occur with using a random amount of producers and consumers. The report will follow the structure according to chapter 5 in Lab 6: FREERTOS SEMAPHORES AND QUEUES.

- Describe the newly introduced problems with using a random amount of producers and consumers.
- Describe how the producer/consumer problems are fixed using semaphores.

II. Results

A. Initial fix for the producers / consumer problem using semaphores

The initial problems that occurred was deadlock and race conditions. The race conditions was fixed by introducing a binary semaphore which the producer / consumer tries to take whenever it should write something to the buffer. This ensures that no producer / consumer can read or write to the buffer if another producer / consumer already acquired the binary semaphore and thus no values can be overwritten or read when it is not supposed to.

Deadlock avoidance was achieved by introducing two counting semaphores. A counting semaphore that keeps track of the available slots in the shared buffer (semaphore-empty) and A counting semaphore that keeps track if the shared buffer is full or not (semaphore-full). Whenever the producer / consumer task is executing, they will take or give the corresponding semaphore, either semaphore-empty or semaphore-full. If the producer cannot take semaphore-empty. Then the producer will wait until a consumer has increased the value of semaphore-empty.

The same works for the consumer, but it tries to take semaphore-full. If the consumer can take semaphore-full, then the count for semaphore-full will decrease by 1 indicating that it is going to remove one byte from the shared buffer. After the consumer has removed the byte, it will give semaphore-empty, increasing the value of the count by 1.

B. Newly introduced problems with using a random amount of producers and consumers

After creating a random amount of producers and consumers, the program executed like it had previously done. No indications of race conditions or deadlocks could be found. The issues that could be determined are, fairness of execution time, debugging complexity, and resource management.

Fairness includes the execution time for each of the producers / consumers, more explicit is that the execution time could not be established for each of the tasks, and some tasks could get more execution time and others less. I.e. the scheduling of these task was not set such that each got the same amount of execution time. This is not a problem causing any noticeable errors for this assignment, but for larger programs optimization of the scheduling is needed.

When trying to debug the program, the randomly introduced producers / consumers makes the program behave differently each run, making any errors occurring difficult to reproduce and thus increasing the complexity.

C. Describe how the producer / consumer problems are fixed, with a random amount of producers / consumers

The issues found in the program does not make the program throw any errors and still works as intended, and with the suggested optimizations the complexity of the program increases, and thus it is deemed not necessary to fix for this assignment.