

CSE320 : System Fundamentals II

Assignment 7

DUE: 29-Nov-2024

Mastering Advanced C Programing Concepts - Semaphores

Overview/Goals

The objective of this assignment is to reinforce your understanding of advanced C programming concepts. You will work on a solution to the 'sleeping barber' problem.

The assignment will give you experience with:

1. Understanding the semaphore APIs and semaphore usage.
2. Understanding how to avoid deadlocks and resolve race conditions that may result in work being 'discontinued' because of event timing.

Problem Overview

The sleeping barber problem is a well-known challenge in inter-process communication and synchronization, highlighting the complexities that arise in systems with multiple operating system processes. This problem was initially introduced by computer science pioneer Edsger Dijkstra in 1965, using it to emphasize the potential redundancy of general semaphores ([https:// en.wikipedia.org/wiki/Sleeping_barber_problem](https://en.wikipedia.org/wiki/Sleeping_barber_problem))

The scenario involves a hypothetical barbershop with one barber, a single barber chair, and a waiting room containing n chairs (where n may be 0) for waiting customers. The rules governing this scenario are as follows:

- If no customers are present, the barber falls asleep in the chair.
- A customer must wake the barber if they find him asleep.
- If a customer arrives while the barber is working, the customer leaves if all chairs are occupied; otherwise, they sit in an empty chair if available.
- When the barber completes a haircut, they check the waiting room for any waiting customers and fall asleep if none are found.

Two main complications arise in this scenario. Firstly, there is a risk of a race condition, where the barber may sleep while a customer waits for a haircut, leading to potential issues during the various actions involved. Secondly, problems may occur when two customers arrive simultaneously, both attempting to sit in the single available chair.

In the case of the multiple sleeping barbers problem, additional complexity arises in coordinating multiple barbers among waiting customers.

Various solutions exist for the sleeping barber problem, all of which require the use of a mutex to ensure that only one participant can change state at a time. The barber acquires the room status mutex before checking for customers and releases it when transitioning to sleep or haircut. Customers acquire the mutex before entering the shop, release it upon sitting in a waiting room or barber chair, and acquire it again when leaving the shop due to a lack of available seats. Additionally, semaphores are needed to indicate the system's state, such as storing the number of people in the waiting room.

The implementation of a solution provided (assignment7.c) may lead to the potential starvation of a customer or a deadlock problem.

Tasks

1. Download assignment7.c
2. Make your own Makefile to compile
3. Fix any starvation or deadlock problems
4. Any enhancement is allowed.
5. Properly handle any error cases and print error messages in the error cases.
6. Add comments on any of your modifications. Describe why you added them in details.

Implementation

Your solution must address any potential deadlong or starvation possibilities in the code.

Reflection

- Document your code thoroughly with comments explaining each section.
- Prepare a report PDF document summarizing everything such as each step or functions that you implemented, challenges faced, and what you've learned during this assignment.


Submission Instructions

Deliverable Files: Completed assignment7.c, your Makefile, and your brief report packaged in a single zip or tgz (gzipped tar) file.

Important: Remove any debug output before submitting. This can make it difficult to judge the accuracy of the code against the expected results.

Make sure you test the code in an Ubuntu environment. That is where I will test the code and if it does not build, has warnings on the build or crashes, some points will be lost.

Please follow this procedure for submission:

1. Place the deliverable files into a folder by themselves. The folder's name should be CSE320_HW7_<yourname>_<yourid>. So if your name is Alice Kim and your id is 12345678, the folder should be named 'CSE320_HW7_AliceKim_12345678'
2. Compress the folder and submit the zip file.
 - a. On Windows, do this by clicking the right-mouse button while hovering over the folder. Select 'Send to -> Compressed (zipped) folder'. The compressed folder will have the same name with a .zip extension. You will upload that file to the Brightspace.
 - b. On Mac, move the mouse over the folder then right-click (or for single button mouse, use Control-click) and select **Compress**. There should now be a file with the same name and a .zip extension. You will upload that file to the Brightspace.
3. Navigate to the course Brightspace site. Click **Assignments** in the top navbar menu. Look under the category 'Assignments'. Click **Assignment7**.
 - a. Scroll down and under **Submit Assignment**, click the **Add a File** button.
 - b. Click **My Computer** (first item in list).
 - c. Find the zip file and drag it to the 'Upload' area of the presented dialog box.
 - d. Click the **Add** button in the dialog.
 - e. You may write comments in the comment box at the bottom.
 - f. Click **Submit**.  Be sure to do this so I can retrieve the submission!

Grading Criteria

The assignment will be graded for the following items:

Any errors in the compile reduce the grade by 50%.

Any warnings in the compile (but the code builds completely) reduces the grade by 30%.

If code crashes due to issues with improper use of data types, pointers, C language features, or the System API calls, this will reduce the grade by 30%.

If code does not crash but gives incorrect results or results that do not match the assignment specification, it will reduce the grade by 20%

If code is not well commented with reasonable variable names and consistent spacing and indentation, the grade may be reduced by up to 5%.

If the Assignment submission does not follow all directives (debut output is removed, includes required files, etc., is packaging (name of zip or tgz file.)...) the assignment may lose up to 5%.

Quality and completeness of report is worth 10% of the total points.