Assignments

Assignment 5 - In progress

Honor Pledge Accepted
Draft - In progress
Submitted
Returned

Assignment Details

Title
Assignment 5
Due
Apr 4, 2023 11:55 PM
Number of resubmissions allowed
Unlimited
Accept Resubmission Until
Apr 6, 2023 11:55 PM
Status
Honor Pledge Accepted
Grade Scale
Points (max 100.00)
Modified by instructor
Mar 22, 2023 8:46 PM

Instructions

Assignment overview

A town has two trains and five train stations.

- Train 0 can hold a maximum of 100 passengers
- Train 1 can hold a maximum of 50 passengers

There are 500 passengers at station 0 and they all want to get to one of the other four stations.

- 50 passengers want to get to Station 1
- 100 passengers want to get to Station 2
- 250 passengers want to get to Station 3
- 100 passengers want to get to Station 4

The 500 passengers waiting at Station 0 will need to be distributed to the other four stations on one of the two trains. However, **only one train can be in a station at one time**.

For this assignment, write a C program which will simulate this activity.

Purpose

- Learn how to use multi-threading and mutual exclusion to safely update shared values
- Get experience with either
  - `pthread_mutex_init(), pthread_mutex_lock(), pthread_mutex_unlock()` and `pthread_mutex_destroy()` system functions
  - `sem_init(), sem_wait(), sem_post()` and `sem_destroy()` system functions
- Gain more experience with the C programming language from an OS perspective

Computing platform

You are welcome to develop your program on your own workstation if you wish, but you are responsible for ensuring that your program compiles and runs without error on the Gaul computing platform. Marks will be deducted if your program fails to compile, or your program runs into errors on Gaul.

- https://wiki.sci.uwo.ca/sts/computer-science/gaul

Instructions

Attached to this assignment is a tarball with the following files in it. **None of these files should be modified**:

```
Makefile                  <--- A pre-packaged Makefile. This tells you how your program should be structured
run-assignment.sh         <--- A shell script that will automatically run your program
```

Download this tarball and upload it to Gaul. Extract the tarball (`tar -xvf assignment-5.tar`). Change to the `assignment-5` directory.

You will write a program called `assignment-5.c`. This program will:

- Initialize 5 stations with requested passengers and 2 trains with capacities as defined above
- Create two threads (one for each train)
  - Starting at Station 0, attempt to enter a station. When the train has permission to enter
    - If there are passengers to pick up (Station 0), the train will pick up as many passengers as possible
    - If there are passengers to drop off and the train already has passengers
      - Deliver as many passengers as possible to Stations 1, 2, 3, and 4
    - If there are more passengers in the train to drop off but the station has no more passengers to drop off, move on to the next station
    - If there are no more passengers in the train to drop off, move back (station-by-station) to Station 0 to pick up more passengers
  - When all passengers are delivered to their stations, the thread can simply quit

Important notes and requirements:

- Picking up and unloading passengers takes a bit of time. Simulate this with the `sleep()` function. Picking up or dropping off 100 passengers takes 10 seconds. Picking up or dropping off 50 passengers takes 5 seconds. And so on. Since we have 500 passengers, the whole program should take ~50-90 seconds to run with some potential variance depending on how often the threads need to wait on one another. (You can relax this requirement to speed up your testing, but the final version should include the sleep function to help ensure some variance between Train 0 and Train 1)
- There are 6 resources that cannot be shared in this project: The five stations and `STDOUT`. If a train is in a station, the other train cannot be in the same station. Also, if a thread is printing information to the screen, the other thread cannot print to the screen. This will ensure threads do not print out on top of one another and creates a smooth print out to the screen. As long as the train is in a station, it should also have full control of `STDOUT`. This means that, for example, train 0 is ready to enter station 0 and train 1 is ready to enter station 1, but only one of the threads gets to print to the screen, the other has to wait until it can get control of `STDOUT`.

Output

Executing `./assignment-5` should produce the following output:

```
 Train 0 ENTERS Station 0
        Station 0 has 500 passengers to pick up
        Picking up passengers...
        Train 0 is at Station 0 and has 100/100 passengers
        Station 0 has 400 passengers to pick up
 Train 0 LEAVES Station 0
 Train 0 ENTERS Station 1
        Station 1 has 50 passengers to drop off
        Unloading passengers...
        Train 0 is at Station 1 and has 50/100 passengers
        Station 1 has 0 passengers to drop off
 Train 0 LEAVES Station 1
...
 Train 1 ENTERS Station 2
        Station 2 has 0 passengers to drop off
        Train 1 is at Station 2 and has 0/50 passengers
        Station 2 has 0 passengers to drop off
 Train 1 LEAVES Station 2
 Train 1 ENTERS Station 1
        Station 1 has 0 passengers to drop off
        Train 1 is at Station 1 and has 0/50 passengers
        Station 1 has 0 passengers to drop off
 Train 1 LEAVES Station 1
...
 Train 0 ENTERS Station 3
        Station 3 has 0 passengers to drop off
        Train 0 is at Station 3 and has 50/100 passengers
        Station 3 has 0 passengers to drop off
 Train 0 LEAVES Station 3
 Train 0 ENTERS Station 4
        Station 4 has 50 passengers to drop off
        Unloading passengers...
        Train 0 is at Station 4 and has 0/100 passengers
        Station 4 has 0 passengers to drop off
 Train 0 LEAVES Station 4
 Train 1 ENTERS Station 0
        Station 0 has 0 passengers to drop off
        Train 1 is at Station 0 and has 0/50 passengers
        Station 0 has 0 passengers to drop off
 Train 1 LEAVES Station 0
```

and `./run-assignment.sh 5` should produce the following output:

```
Assignment 5 STARTED - Dow Mon  ## ##:##:## AM/PM EST 2023

Cleaning environment
----------------------------
rm -f assignment-5

Checking environment
----------------------------
af36ed4145c2285730afa41687618ffe  ./run-assignment.sh
<unique id>  assignment-5.c
Makefile: OK

Building environment
----------------------------
make all
make[1]: Entering directory '/home/wbeldman/3305/Projects/Assignment 5'
gcc -o assignment-5 assignment-5.c -Wall -Wpedantic -Wextra -std=gnu17
make[1]: Leaving directory '/home/wbeldman/3305/Projects/Assignment 5'

Assignment 5
----------------------------
 Train 0 ENTERS Station 0
        Station 0 has 500 passengers to pick up
        Picking up passengers...
        Train 0 is at Station 0 and has 100/100 passengers
        Station 0 has 400 passengers to pick up
 Train 0 LEAVES Station 0
 Train 0 ENTERS Station 1
        Station 1 has 50 passengers to drop off
        Unloading passengers...
        Train 0 is at Station 1 and has 50/100 passengers
        Station 1 has 0 passengers to drop off
 Train 0 LEAVES Station 1
...
 Train 1 ENTERS Station 2
        Station 2 has 0 passengers to drop off
        Train 1 is at Station 2 and has 0/50 passengers
        Station 2 has 0 passengers to drop off
 Train 1 LEAVES Station 2
 Train 1 ENTERS Station 1
        Station 1 has 0 passengers to drop off
        Train 1 is at Station 1 and has 0/50 passengers
        Station 1 has 0 passengers to drop off
 Train 1 LEAVES Station 1
...
 Train 0 ENTERS Station 3
        Station 3 has 0 passengers to drop off
        Train 0 is at Station 3 and has 50/100 passengers
        Station 3 has 0 passengers to drop off
 Train 0 LEAVES Station 3
 Train 0 ENTERS Station 4
        Station 4 has 50 passengers to drop off
        Unloading passengers...
        Train 0 is at Station 4 and has 0/100 passengers
        Station 4 has 0 passengers to drop off
 Train 0 LEAVES Station 4

real    0m50.004s
user    0m0.001s
sys     0m0.001s

Cleaning environment
```

```
----------------------------
rm -f assignment-5

Assignment 5 COMPLETED - Dow Mon ## ##:##:## AM/PM EST 2023
```

Helpful hints

- You can represent the passenger count as +500 in station 0, -50 in station 1, -100 in station 2, etc. You are done when all 5 stations have a count of 0
  - However you choose to represent passengers, be careful as train 0 may detect that there are no passengers left to pick up because all remaining passengers are "in transit" on train 1. So there are situations where there is nothing for one thread to do but the whole task is not done yet.
- Every pick up and drop off requires some thinking. Usually the train can pick up as many passengers as it can hold and drop off as many passengers as it holds. However, watch out for situations like 50 passengers left on station 0 but train 0 can hold as many as 100 passengers. Of course, only load 50 passengers into train 0, not 100. Same for drop offs: The train may have 100 passengers but only 50 passengers are left to drop off. Drop off 50 passengers, not 100.

Submitting

When you are finished your assignment, follow these steps

1. From inside the assignment-5 directory, run the following command (make sure `run-assignment.sh` is executable): `script -c './run-assignment.sh 5' assignment-5.out`
   Your directory should now contain the following files:

```
assignment-5.c    <--- Your program
assignment-5.out  <--- The output produced by running the script command above
Makefile          <--- A pre-packaged Makefile. This tells you how your program should be structured
run-assignment.sh <--- A shell script that will automatically run your program and put the results in assignment-5.out
```

2. Assuming the command was successful, run the follow command to get out of the `assignment-5` directory: `cd ..`
3. Package your assignment into a tarball: `tar -cvf assignment-5.tar assignment-5`
4. Verify the contents of your tarball (`tar -tvf assignment-5.tar`) (`du -sh assignment-5.tar`). **If your tarball is 10kb in size** you have an empty tarball and you made an error on this step. Make sure you are properly creating your tarball with the right files in it.
5. Use an SFTP program to download the tarball and then upload it to OWL.

Additional resources for assignment

- Assignment-5.tar (10 KB; Mar 5, 2023 12:47 pm)

Grading Rubric

Preview Rubric

Submission

Attachments

No attachments yet

Select a file from computer Choose File No file chosen

Proceed Preview Save Draft Cancel

Don't forget to save or proceed!