MITx 6.004.3x
Computation Structures 3: Computer Organization

Help

Course Progress Dates Discussion

Course / 19. Concurrency and Synchronization / Lecture Videos (36:48)

Previous Next

LE19.3

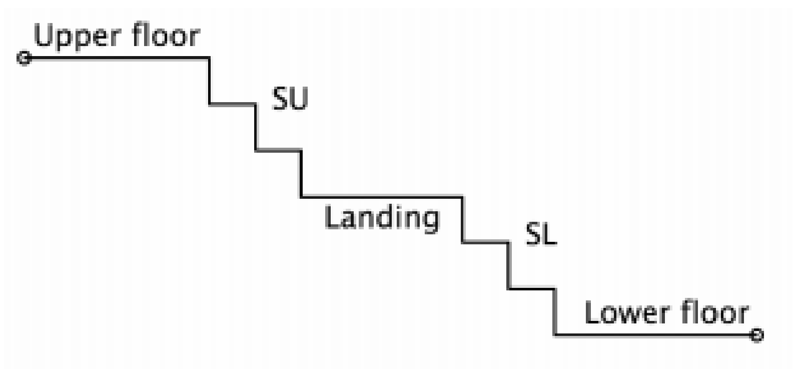
Bookmark this page

Calculator

LE19.3.1: Stairs!

0.0/1.0 point (ungraded)

The MIT Safety Office is worried about congestion on stairs and has decided to implement a semaphore-based trafficcontrol system. Most connections between floors have two flights of stairs with an intermediate landing (see figure). The constraints the Safety Office wishes to enforce are



- Only 1 person at a time on each flight of stairs
- A maximum of 3 persons on a landing
- As few traffic constraints as possible
- No deadlock (a particular concern if there's bidirectional travel)

Assume stair traffic is unidirectional: once on a flight of stairs, people continue up or down until they've reached their destination floor (no backing up!), although they may pause at the landing.

There are three semaphores: they control the upper flight of stairs (SU), the landing (L), and the lower flight of stairs (SL). **Please provide appropriate initial values for these semaphores and add the necessary wait() and signal() calls to the Down() and Up() procedures below.** Note that the Down() and Up() routines will be executed by many students simultaneously and the semaphores are the only way their code has of interacting with other instances of the Down() and Up() routines. Your code must avoid deadlock and enforce the stair and landing occupancy constraints. **Hint:** You may find it easier to first implement a solution where only 1 person at time is in-between floors (but be careful of deadlock here too!).

For each drop down, select the missing line of code. If a particular code region only requires one command, then select that command for the first drop down and select None for the second drop down. If no commands are needed in a region then select None for both answers.

// Semaphores shared by all students, provide initial values

semaphore SU = Answer: 1 ;

semaphore SL = Answer: 1 ;

semaphore L = Answer: 3 ;

// code for going downstairs		// code for going upstairs	
Down() {		Up() {	
Select an option	Answer: wait(L)	Select an option	Answer: wait(L)
Select an option	Answer: wait(SU)	Select an option	Answer: wait(SL)
Enter SU;		Enter SL;	
Select an option	Answer: None	Select an option	Answer: None
Select an option	Answer: None	Select an option	Answer: None

Calculator

Select an option Answer: None Select an option Answer: None

Exit SU / enter landing;	Exit SL / enter landing;
Select an option Answer: signal(SU)	Select an option Answer: signal(SL)
Select an option Answer: wait(SL)	Select an option Answer: wait(SU)
Exit landing / enter SL;	Exit landing / enter SU;
Select an option Answer: signal(L)	Select an option Answer: signal(L)
Select an option Answer: None	Select an option Answer: None
Exit SL;	Exit SU;
Select an option Answer: signal(SL)	Select an option Answer: signal(SU)
Select an option Answer: None	Select an option Answer: None
}	}

Explanation
We are told that only 1 person can be on either the upper stairs or the lower stairs, while up to 3 can be on the landing. To achieve this we initialize our semaphores to SU = 1, SL = 1, and L = 3.
In order to avoid deadlock, we must first request the L semaphore in both the Down() and Up() routines. If we don't do this, then imagine a situation where there are already 3 students on the landing and no students on either staircase. It would be possible for two new students to grab the SU and SL semaphores and then get stuck waiting for the L sempahore. Unfortunately, at the same time, anyone trying to leave the landing would also need to request an SU or SL semaphore but none would be available and we would enter a deadlock situation. To avoid this, we first ensure that we have room for us on the landing, then we request the semaphore that we need for the particular staircase we are about to enter.
Upon exiting a staircase, we always signal the semaphore for that staircase so that someone else can make use of it. We then remain on the landing until we are able to grab the semaphore for the second half of our staircase journey. Upon entering the second staircase, and thus exiting the landing, we can free the landing semaphore. Finally, when we exit the second staircase, we free the semaphore that was required for that staircase.

Submit

Answers are displayed within the problem

Discussion

Hide Discussion

Topic: 19. Concurrency and Synchronization / LE19.3

Add a Post

Show all posts by recent activity

There are no posts in this topic yet.

✖

Calculator

< Previous Next >

edX

About
Affiliates
edX for Business
Open edX
Careers
News

Legal

Terms of Service & Honor Code
Privacy Policy
Accessibility Policy
Trademark Policy
Sitemap
Cookie Policy
Your Privacy Choices

Connect

Idea Hub
Contact Us
Help Center
Security
Media Kit

GET IT ON Google play
Download on the App Store

Calculator