LE19.1

Bookmark this page

LE19.1.1: Going Bananas

0.0/1.0 point (ungraded)

The Chiquita Fruit Company has acquired a fancy multi-process machine that they want to use to print the word BANANA. The machine has two types of processes – called BA and NA – that can coordinate their execution via shared semaphores which respond to the standard signal(S) and wait(S) procedures. Since we need twice as many NAs as BAs, there's **one BA process and two NA processes** running on the machine. Assume that execution may switch between any of the three processes at any point in time.

Their summer intern wrote many versions of the BA and NA code, then made a list of the first six characters they printed (any subsequent printed characters were not recorded) when run on the machine:

1. Select all possible strings that the following code could produce.

```
Process BA

  Loop1:
     print("B")
     print("A")
     goto Loop1
```

```
Process NA

  Loop2:
     print("N")
     print("A")
     goto Loop2
```

- [ ] BANANA
- [ ] ANANAB
- [ ] BNNAAA
- [ ] BANNAA
- [ ] BANBAN
- [ ] NABANA

2. Select all possible strings that the following code could produce.

```
Semaphore S = 0; Semaphore T = 0;    // Shared semaphores
```

```
Process BA

  Loop1:
     print("B")
     print("A")
     signal(S)
     signal(S)
     wait(T)
     wait(T)
     goto Loop1
```

```
Process NA

  Loop2:
     wait(S)
     print("N")
     print("A")
     signal(T)
     goto Loop2
```

- [ ] BANANA
- [ ] ANANAB
- [ ] BNNAAA
- [ ] BANNAA

Calculator

- [ ] BANBAN
- [ ] NABANA

3. A clever 6.004 student observes that interchanging two lines in the Process BA code of part (B) will ensure that only BANANA will be printed. Select the **two** commands from the loop of process BA that should be interchanged. Assume the "NA" process is untouched.

- [ ] print("B")
- [ ] print("A")
- [ ] signal(S)
- [ ] signal(S)
- [ ] wait(T)
- [ ] wait(T)
- [ ] goto Loop1

Submit

LE19.1.2: Precedence Constraints

0.0/1.0 point (ungraded)
The following pair of processes share a common variable X:

```
Process A                         Process B
    int Y;                            int Z;
A1: Y = X * 2;                    B1: Z = X + 1;
A2: X = Y;                        B2: X = Z;
```

X is set to 5 before either process begins execution. As usual, statements within a process are executed sequentially, but statements in process A may execute in any order with respect to statements in process B. There are four possible values for X. Here are the possible ways in which statements from A and B can be interleaved:

```
A1 A2 B1 B2: X = 11
A1 B1 A2 B2: X = 6
A1 B1 B2 A2: X = 10
B1 A1 B2 A2: X = 10
B1 A1 A2 B2: X = 6
B1 B2 A1 A2: X = 12
```

1. The programs are modified as follows to use a shared binary semaphore T:

```
semaphore T = 1;   // shared semaphore

Process A                         Process B
    int Y;                            int Z;
    wait(T)                           wait(T)
A1: Y = X * 2;                    B1: Z = X + 1;
A2: X = Y;                        B2: X = Z;
    signal(T)                         signal(T)
```

Calculator

Select all the possible values of X after both processes finish executing:

- [ ] 6
- [ ] 10
- [ ] 11 ✔
- [ ] 12 ✔

Explanation
The addition of the wait(T) command before executing either process A or B, and initializing semaphore T to 1, guarantees that only one of the processes can run at a time, their execution cannot be interleaved. The second process only gets to run after the first one completes and signals T once again. So either the execution order is A1 A2 B1 B2 which produces X = 11, or the execution order is B1 B2 A1 A2 which produces X = 12.

2. Using two semaphores, S and T, please specify the initial values of semaphores S and T, and add wait(S), wait(T), signal(S), and signal(T) statements as needed to modify the original procedures so that the only **possible value of X is 10**. To be considered correct, the solution must not introduce any unnecessary precedence constraints, and must execute all four pieces of code A1, A2, B1, and B2. *Note there are two execution orders that result in X = 10.* Indicate the initial value of any semaphores you use.

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. **Assume that semaphore T is required to execute A2.** Note that some drop downs may contain a subset of all the possibilities in order to ensure a single correct answer to the problem.

// Semaphore initial values

semaphore S = [ ] Answer: 0 ;

semaphore T = [ ] Answer: 0 ;

// Process A		// Process B	
int Y;		int Z;	
Select an option	Answer: None	Select an option	Answer: None
Select an option	Answer: None	Select an option	Answer: None
A1: Y = X * 2;		B1: Z = X + 1;	
Select an option	Answer: signal(S)	Select an option	Answer: wait(S)
Select an option	Answer: wait(T)	Select an option	Answer: None
A2: X = Y;		B2: X = Z;	

Calculator

Select an option Answer: None Select an option Answer: signal(T)

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

Explanation
The two code sequences that produce X = 10 are A1 B1 B2 A2 and B1 A1 B2 A2. Another way of thinking about it is that A1 and B1 must run 1st and 2nd, in either order, B2 must be run third, and finally A2 must run last. To allow A1 and B1 to be run first in either order, we don't put any wait statements before A1 and B1. Since A1 and B1 are each exactly one instruction long, there is no interleaving that can happen between A1 and B1. We want B2 to run next but not before A1 has completed, so we signal(S) after A1 completes and wait(s) after B1. Once the B process gets semaphore S, it is guaranteed that both A1 and B1 have run. Next, we use semaphore T to ensure that A2 runs last.

Submit

Answers are displayed within the problem

Discussion

Topic: 19. Concurrency and Synchronization / LE19.1

Hide Discussion

Add a Post

Show all posts by recent activity

Should X be set to 5 before the start of Problem 19.1.2 (B)?

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

Calculator

Connect

[Idea Hub]
[Contact Us]
[Help Center]
[Security]
[Media Kit]

Calculator