

Solution #1

Ubuntu 64-bit - VMware Workstation

File Edit View VM Tabs Help

Library Type here to search

Activities Terminal Oct 26 19:42

cse4001@ubuntu:~/fall2020-synchronization-azim-shaf\$./cse4001_sync 1

```
87 // nsWriter *nsWriter ( void *threadid)
88 void *nsWriter ( void *threadid)
89 {
90     // Thread number
91     int x = (long)threadID;
92
93     while(1) {
94         sleep(3); // Slow the threads
95         turnstile.wait();
96         roomEmpty.wait();
97         printf("Writer %d : writing\n", x);
98         Rhythmbox_fflush(stdout);
99         turnstile.signal();
100        roomEmpty.signal();
101    }
102
103    //pthread_exit(0);
104
105 /* No-Starve Reader Function */
106 /**
107 void *nsReader ( void *threadid)
108 {
109     // Thread number
110     int x = (long)threadID;
111
112     while(1) {
113         turnstile.wait();
114         turnstile.signal();
115         readswitch.wait();
116         readers += 1;
117         if (readers == 1) {
118             roomEmpty.wait();
119         }
120         readswitch.signal();
121         readers -= 1;
122     }
123 }
```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

Save

sync.cpp

sync.cpp

C++ Tab Width: 8 Ln 195, Col 10 INS

10:42 PM 10/26/2020

No-Starve Reader-Writer: This solution works by allowing the readers to lock the room using a lightswitch, but aren't allowed to stay indefinitely since the writers control a turnstile semaphore that limits how many readers can enter. Locks were implemented using semaphores and counters rather than creating a class.

Solution #2

Ubuntu 64-bit - VMware Workstation

File Edit View VM Tabs Help |

Library Type here to search... Home Activities Terminal Oct 26 19:47

Activities Terminal Oct 26 19:47

Open main.cpp cse4001@ubuntu:~/fall2020-synchronization-azim-shafiq\$./cse4001_sync_2 sync.cpp Save

sync.cpp

```
87 */  
88 void *nsWriter ( void *thread )  
89 {  
90     // Thread number  
91     int x = (long)threadID;  
92     while(1) {  
93         sleep(3); // Slow the threads  
94         turnstile.wait();  
95         roomEmpty.wait();  
96         printf("Writer %d: writing\n", x);  
97         fflush(stdout);  
98         turnstile.signal();  
99         roomEmpty.signal();  
100    }  
101    /*  
102     No-Starve Reader function  
103     */  
104    void *nsReader ( void *thread )  
105 {  
106     // Thread number  
107     int x = (long)threadID;  
108     while(1) {  
109         turnstile.wait();  
110         turnstile.signal();  
111         readSwitch.wait();  
112         readers += 1;  
113         if (readers == 1) {  
114             roomEmpty.wait();  
115             readSwitch.signal();  
116             writer.signal();  
117         }  
118         turnstile.wait();  
119         turnstile.signal();  
120         readSwitch.signal();  
121         writers -= 1;  
122         if (writers == 0) {  
123             roomEmpty.signal();  
124         }  
125     }  
126 }
```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

C++ Tab Width: 8 Ln 195, Col 10 INS

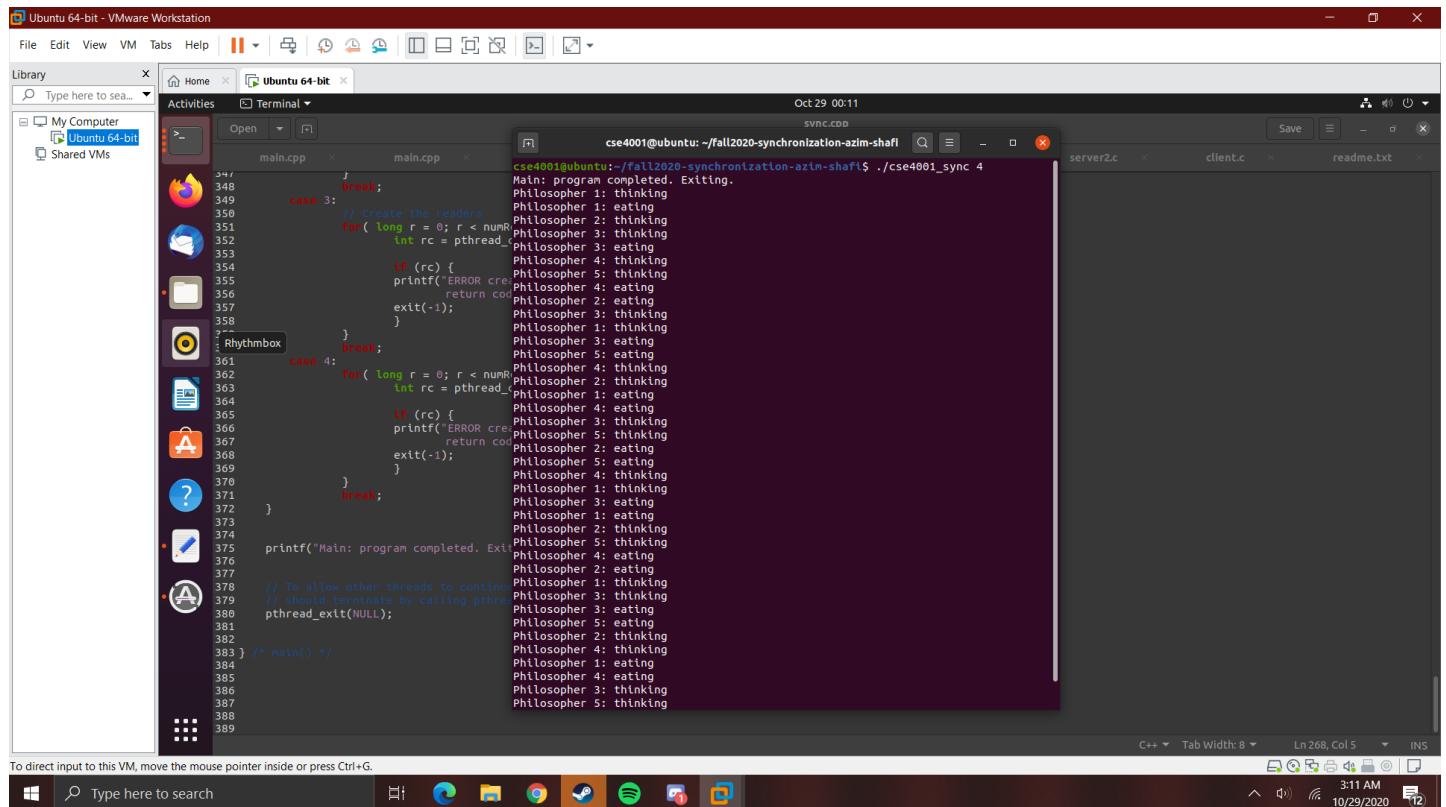
Type here to search

Writer Priority Reader-Writer: This solution works by allowing both readers and writers to lock the room, but writers are given priority since writers must enter and leave one-by-one, but readers enter one-by-one then exit en masse. The `readswitch` implementation used in the first solution was mirrored to create the `writeswitch` here.

Solution #3

Dining Philosophers #1: This solution solves deadlock by ensuring that only four can be seated at the table to ensure a situation in which all aren't waiting for their second fork. An array of semaphores and modulo statements were used as suggested by the book, but due to trouble implementing the provided semaphore class as an array, five semaphores and a switch statement were used originally, but a discussion in class cleared up that issue.

Solution #4



```
cse4001@ubuntu:~/fall2020-synchronization-azim-shafi$ ./cse4001_sync 4
Main: program completed. Exiting.
Philosopher 1: thinking
Philosopher 1: eating
Philosopher 2: thinking
Philosopher 2: eating
Philosopher 3: thinking
Philosopher 3: eating
Philosopher 4: thinking
Philosopher 4: eating
Philosopher 5: thinking
Philosopher 5: eating
Philosopher 1: thinking
Philosopher 1: eating
Philosopher 2: thinking
Philosopher 2: eating
Philosopher 3: thinking
Philosopher 3: eating
Philosopher 4: thinking
Philosopher 4: eating
Philosopher 5: thinking
Philosopher 5: eating
Philosopher 1: thinking
Philosopher 1: eating
Philosopher 2: thinking
Philosopher 2: eating
Philosopher 3: thinking
Philosopher 3: eating
Philosopher 4: thinking
Philosopher 4: eating
Philosopher 5: thinking
Philosopher 5: eating
Philosopher 1: thinking
Philosopher 1: eating
Philosopher 2: thinking
Philosopher 2: eating
Philosopher 3: thinking
Philosopher 3: eating
Philosopher 4: thinking
Philosopher 4: eating
Philosopher 5: thinking
Philosopher 5: eating
Philosopher 1: thinking
Philosopher 1: eating
Philosopher 2: thinking
Philosopher 2: eating
Philosopher 3: thinking
Philosopher 3: eating
Philosopher 4: thinking
Philosopher 4: eating
Philosopher 5: thinking
Philosopher 5: eating
Philosopher 1: thinking
Philosopher 1: eating
Philosopher 2: thinking
Philosopher 2: eating
Philosopher 3: thinking
Philosopher 3: eating
Philosopher 4: thinking
Philosopher 4: eating
Philosopher 5: thinking
Philosopher 5: eating
Philosopher 1: thinking
Philosopher 1: eating
Philosopher 2: thinking
Philosopher 2: eating
Philosopher 3: thinking
Philosopher 3: eating
Philosopher 4: thinking
Philosopher 4: eating
Philosopher 5: thinking
Philosopher 5: eating
Philosopher 1: thinking
Philosopher 1: eating
Philosopher 2: thinking
Philosopher 2: eating
Philosopher 3: thinking
Philosopher 3: eating
Philosopher 4: thinking
Philosopher 4: eating
Philosopher 5: thinking
Philosopher 5: eating
Philosopher 1: thinking
Philosopher 1: eating
Philosopher 2: thinking
Philosopher 2: eating
Philosopher 3: thinking
Philosopher 3: eating
Philosopher 4: thinking
Philosopher 4: eating
Philosopher 5: thinking
Philosopher 5: eating
Philosopher 1: thinking
Philosopher 1: eating
Philosopher 2: thinking
Philosopher 2: eating
Philosopher 3: thinking
Philosopher 3: eating
Philosopher 4: thinking
Philosopher 4: eating
Philosopher 5: thinking
Philosopher 5: eating
Main: program completed. Exiting.
```

Dining Philosophers #2: This solution solves deadlock by ensuring at least one leftie and rightie are at the table. This was done by assigning odd philosophers to grab the left fork first, while even philosophers grab the right fork first.