****

**National University of Computer & Emerging Sciences (FAST-NU)**

**Ice Cream Shop Problem.**

**Submitted to:**

**Miss Safia**

**By:**

**Project Team**

**Ajay Kumar 18K-1136**

**Muhammad Ali Khan 18K-0337**

**Hafiz Sameed 18K-0323**

**Submission Date**

**[ June 20th, 2020 ]**

**ICE CREAM SHOP PROBLEM**

1. **DESCRIPTION:**

Ice cream Shop problem is a project in operating systems that creates a simulation of an ice cream shop where multiple customers are entertained with the prevention of flavor condition.

1. **HOW WE STARTED:**

First, we made (or tried to make) a system call that printed a string and an ID of the customer as soon as someone left the store or if we run out of the inventory etc. Then, we made the code in C language that used threads and semaphores and we also handled occurrence of deadlock.

1. **PROBLEMS FACED:**

We Faced problem in the making of the system call. We were getting the same error over and over again, that we have an undefined reference to our function.

1. **THE ACTUAL WORKING OF THE PROJECT (Methodology):**

The working of the project is as follow:

Every Customer is given an id and then the customers enter the ticket counter, after getting there tickets checked, they enter the flavor counter, after choosing their flavors, they enter the topping counter, and after choosing their topping they enter the payment counter, and after they pay their bill, they exit.

1. **EXPLANATION OF SOURCE CODE:**

We have implemented it in complete C language. We used Multi-threading and semaphores. Firstly, we initialize the prices of flavors and topping, number of tickets, flavor, and toppings, semaphores and the prototype of the function.

In main Function, we asked for the number of customers, then we give each customer their own specific ID, then we initialize each semaphore of all the counters, flavors, and toppings. Then we create thread array for number of customers and for each customer we call the function ice-cream.

In Ice-cream Function, we use sem\_wait which means that the customer has entered the ticket counter and is waiting, then we check if we have run out tickets (if tickets <= 0)  
then we leave the shop as we have run out of tickets, else we decrement and accept the ticket and exit the counter with sem\_post. Then we enter the flavor counter with the same semaphore function and now flavor condition may occur here but it will be handled through local thread variable because our condition is (flavor[0] == 0 && flavor[1] == 0 && flavor[2] == 0) and this will only work if we run out of all flavors but what if we run out of only one flavor? For this we create a variable which will be incremented inside a chosen flavor and inside the condition (flavor[0] > 0) and if we have run out of that specific flavor then the variable will remain 0 and if the variable is 0, then the customer will leave the shop as that specific flavor has finished. After leaving the flavor counter, the customer enters the topping counter and then the payment counter where the bill is appended in the total revenue generated by the shop.

Then after returning to the main functions, all the threads are joined and we print out the number of customers, total revenue, and the number of tickets remaining (if there are any). Then finally we destroy all the semaphores.

1. **PROJECT ASPECTS IN CURRENT/FUTURE TECHNOLOGY:**

None.

1. **Project configurations and Code**

*CODE:*

|  |  |
| --- | --- |
|  | #include <stdio.h>  #include <stdlib.h> |
|  | #include <unistd.h> |
|  | #include <pthread.h> |
|  | #include <semaphore.h> |
|  | #include <string.h> |
|  | #include <sys/syscall.h> |
|  | #include <linux/kernel.h> |
|  |  |
|  |  |
|  | #define priceF1 1.05 |
|  | #define priceF2 2.00 |
|  | #define priceF3 1.67 |
|  | #define priceT1 0.8 |
|  | #define priceT2 0.5 |
|  |  |
|  |  |
|  | int ticket = 30, \_flavors[3] = {15, 15, 15}, \_toppings[2] = {20, 20}; |
|  | double revenue = 0.0; |
|  |  |
|  | sem\_t tc, fc, f1, f2, f3, toc, t1, t2, pc; |
|  |  |
|  |  |
|  | void \*iceCream(void \*\_id); |
|  | int main() |
|  | { |
|  | int nc; |
|  |  |
|  | printf("\n\nEnter The Number Of Customers [1-%d]: ", ticket); |
|  | scanf("%d", &nc); |
|  | printf("\n\n"); |
|  |  |
|  | If (nc > ticket || nc <= 0) |
|  | { |
|  | Printf ("\n\nInvalid Input!\n\n"); |
|  | return 0; |
|  | } |
|  |  |
|  | int id[nc]; |
|  |  |
|  | For (int i=0; i<nc; i++) |
|  | { |
|  | id[i] = i+100; |
|  | } |
|  |  |
|  | sem\_init (&tc, 0, 1); |
|  | sem\_init (&fc, 0, 3); |
|  | sem\_init (&f1, 0, 1); |
|  | sem\_init (&f2, 0, 1); |
|  | sem\_init (&f3, 0, 1); |
|  | sem\_init (&toc, 0, 2); |
|  | sem\_init (&t1, 0, 1); |
|  | sem\_init (&t2, 0, 1); |
|  | sem\_init (&pc, 0, 1); |
|  | pthread\_t \_customer[nc]; |
|  |  |
|  | For (int i=0; i<nc; i++) |
|  | { |
|  | pthread\_create(&\_customer[i], 0, &iceCream, (void\*) &id[i]); |
|  | } |
|  |  |
|  | For (int i=0; i<nc; i++) |
|  | { |
|  | pthread\_join(\_customer[i], NULL); |
|  | } |
|  | Printf ("Number Of Customers: %d", nc); |
|  | Printf ("\nRevenue Generated: $ %f", revenue); |
|  | Printf ("\nTickets Remaining: %d\n\n", ticket); |
|  |  |
|  | sem\_destroy(&tc); |
|  | sem\_destroy(&fc); |
|  | sem\_destroy(&f1); |
|  | sem\_destroy(&f2); |
|  | sem\_destroy(&f3); |
|  | sem\_destroy(&toc); |
|  | sem\_destroy(&t1); |
|  | sem\_destroy(&t2); |
|  | sem\_destroy(&pc); |
|  | return 0; |
|  | } |
|  | void \*iceCream(void \*id) |
|  | { |
|  | int ID = \*(int\*) id, check = 0; |
|  | double bill = 0.0; |
|  |  |
|  | // ticket counter - ENTER |
|  | sem\_wait(&tc); |
|  |  |
|  | If (ticket <= 0) |
|  | { |
|  | printf("\nCustomer[%d]: Leaving Shop. [REASON]: Tickets Finished\n", ID); |
|  | //syscall(348,"Leaving Shop. [REASON]: Tickets Finished\n", ID); |
|  | Sleep (1); |
|  | pthread\_exit (NULL); |
|  | } |
|  |  |
|  | ticket--; |
|  | printf("Customer[%d] Got Ticket.\n", ID); |
|  | //syscall(348,"Got Ticket.\n", ID); |
|  |  |
|  | sem\_post(&tc); |
|  | Sleep (2); |
|  |  |
|  | sem\_wait(&fc); |
|  |  |
|  | if(\_flavors[0] <= 0 && \_flavors[1] <= 0 && \_flavors[2] <= 0) |
|  | {  Printf ("Leaving Shop. [REASON]: Flavors Finished\n") |
|  | //syscall(348,"Leaving Shop. [REASON]: Flavors Finished\n", ID); |
|  | sleep(1); |
|  | pthread\_exit(NULL); |
|  | } |
|  | Else |
|  | { |
|  | sem\_wait(&f1); |
|  |  |
|  | if(\_flavors[0] > 0) |
|  | { |
|  | \_flavors[0]--; |
|  | check++; |
|  | bill = bill + priceF1; |
|  | printf("\nCustomer[%d]: Got Flavor[0].\n", ID); |
|  | //syscall(348,"Got Flavor[0].\n", ID); |
|  | sleep(1); |
|  | } |
|  |  |
|  | sem\_post(&f1); |
|  |  |
|  | sem\_wait(&f2); |
|  |  |
|  | if(\_flavors[1] > 0) |
|  | { |
|  | \_flavors[1]--; |
|  | check++; |
|  | bill = bill + priceF2; |
|  | printf("Customer[%d]: Got Flavor[1].\n", ID); |
|  | //syscall(348,"Got Flavor[1].\n", ID); |
|  | sleep(1); |
|  | } |
|  |  |
|  | sem\_post(&f2); |
|  |  |
|  | sem\_wait(&f3); |
|  |  |
|  | if(\_flavors[2] > 0) |
|  | { |
|  | \_flavors[2]--; |
|  | check++; |
|  | bill = bill + priceF3; |
|  | printf("Customer[%d]: Got Flavor[2].\n", ID); |
|  | //syscall(348,"Got Flavor[2].\n", ID); |
|  | sleep(1); |
|  | } |
|  |  |
|  | sem\_post(&f3); |
|  |  |
|  | if(check== 0) |
|  | { |
|  | printf("\nCustomer[%d]: Leaving Shop. [REASON]: Flavors Finished\n", ID); |
|  | //syscall(348,"Leaving Shop. [REASON]: Flavors Finished\n", ID); |
|  | sleep(1); |
|  | pthread\_exit(NULL); |
|  | } |
|  | } |
|  |  |
|  | printf("\nCustomer[%d]: Got Flavor(s). Leaving Flavor Counter\n", ID); |
|  | //syscall(348,"Got Flavor(s). Leaving Flavor Counter\n", ID); |
|  | //sleep(2); |
|  |  |
|  | sem\_post(&fc); |
|  |  |
|  | sem\_wait(&toc); |
|  |  |
|  | sem\_wait(&t1); |
|  |  |
|  | if(\_toppings[0] > 0) |
|  | { |
|  | \_toppings[0]--; |
|  | bill = bill + priceT1; |
|  | } |
|  |  |
|  | sem\_post(&t1); |
|  |  |
|  | sem\_wait(&t2); |
|  |  |
|  | if(\_toppings[1] > 0) |
|  | { |
|  | \_toppings[1]--; |
|  | bill = bill + priceT2; |
|  | } |
|  |  |
|  | sem\_post(&t2); |
|  |  |
|  | //printf("\nCustomer[%d]: Leaving Topping Counter.\n", ID); |
|  | syscall(348,"Leaving Topping Counter.\n", ID); |
|  |  |
|  | sem\_post(&toc); |
|  |  |
|  | sleep(2); |
|  |  |
|  | sem\_wait(&pc); |
|  |  |
|  | revenue = revenue + bill; |
|  |  |
|  | printf("\nCustomer[%d]: Billed: $ %f.\n", ID, bill); |
|  | char a[100]; |
|  | sprintf(a,"Billed: $ %f.\n",bill); |
|  | //syscall(348,a, ID); |
|  |  |
|  | sem\_post(&pc); |
|  |  |
|  | sleep(2); |
|  |  |
|  | printf("\nCustomer[%d]: Leaving Ice-Cream Shop.\n\n", ID); |
|  | //syscall(348,"Leaving Ice-Cream Shop.\n\n", ID); |
|  |  |
|  | return NULL; |
|  | } |