

**National University of Computer and Emerging Sciences**



<Frozen Desert Facility Simulation System Call>

**Group Members**

<Ashar Zamir>…………<22K-4241>

<Anas Saleem>…………<22K-0500>

<Bilal Ahmed>…………<22K-4525>

**Supervised by**

**<Miss Anaum Hamid>**

**FAST School of Computing**

**National University of Computer and Emerging Sciences**

**[Karachi], Pakistan**

**May 2024**

**Frozen Dessert Facility Simulation System Call**

**1. Introduction**

In today’s fast-paced retail environment, managing a frozen dessert facility such as an ice cream shop demands meticulous attention to stock control and inventory management. To address this, we embarked on a project aimed at creating a simulation of an ice cream shop that leverages operating system principles. This simulation was designed to efficiently manage inventory and enhance customer service by illustrating the critical role of threads and system calls. By simulating the interactions between customers and service counters, our project highlights how operating system frameworks can prevent race conditions and ensure resource synchronization.

This project was initiated on May 1, 2024, and successfully completed on May 7, 2024. It involved the collaborative efforts of students Ashar, Anas, and Bilal from Fast University. Through this simulation, we aim to demonstrate the practical applications of operating system concepts in a real-world retail setting, thereby enhancing operational efficiency and customer satisfaction in an ice cream shop.

* 1. **Features:**

1. Customer Interaction: Describe how the code allows customers to interact with the ice cream parlor by selecting their preferred flavor and topping.

2. Concurrency: Discuss the implementation of multithreading using pthreads, allowing multiple customers to be served concurrently.

3. Mutexes: Explain the use of mutexes to ensure thread safety when accessing shared resources such as flavor and topping availability, as well as the total revenue.

4. Dynamic Pricing: Highlight the dynamic pricing of flavors and toppings, with prices stored in arrays for easy modification.

5. Error Handling: Emphasize the error handling mechanisms in place, such as informing customers when their selected flavor or topping is unavailable and prompting them to make another selection.

6. Logging: Discuss the logging feature implemented using a custom system call (`SYS\_MY\_CUSTOM\_LOG`) to record customer transactions, including their name, selected flavor and topping, and total bill.

* 1. **Technology used:**

1. C Programming Language: The code is written in the C programming language, a widely used language for system programming.
2. POSIX Threads (pthread): POSIX Threads are used for implementing multithreading in the C code.
3. VMware : Virtual machine was created on VM ware workstation 17.
4. Linux(ubuntu)
   1. **Code snippets:**

**Userspace code:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <string.h>

#include <sys/syscall.h>

#include <unistd.h>

#define SYS\_MY\_CUSTOM\_LOG 335 // Define the syscall number

#define NUM\_FLAVORS 5

#define NUM\_TOPPINGS 5

#define MAX\_CUSTOMERS 30

double pricesFlavors[NUM\_FLAVORS] = {100.00, 200.00, 150.00, 180.00, 500.00};

double pricesToppings[NUM\_TOPPINGS] = {90, 50, 70, 60, 30};

typedef struct {

char name[50];

int id;

int flavor;

int topping;

double bill;

} customer;

customer customers[MAX\_CUSTOMERS];

int flavors[NUM\_FLAVORS] = {1, 34, 18, 25, 20};

int toppings[NUM\_TOPPINGS] = {1, 34, 28, 22, 18};

double revenue = 0.0;

pthread\_mutex\_t flavorMutex = PTHREAD\_MUTEX\_INITIALIZER;

pthread\_mutex\_t toppingMutex = PTHREAD\_MUTEX\_INITIALIZER;

pthread\_mutex\_t revenueMutex = PTHREAD\_MUTEX\_INITIALIZER;

void \*counter\_thread(void \*arg);

int selectFlavor();

int selectTopping();

int main() {

pthread\_t threads[MAX\_CUSTOMERS];

int numCustomers;

printf("Welcome to The Frozen Cream\n");

printf("We serve ice creams and happiness!\n");

printf("Enter Number Of Customers [1-%d]: ", MAX\_CUSTOMERS);

scanf("%d", &numCustomers);

if (numCustomers > MAX\_CUSTOMERS || numCustomers <= 0) {

printf("Invalid Input!\n");

return 0;

}

for (int i = 0; i < numCustomers; i++) {

printf("Enter Customer %d name: ", i + 1);

scanf("%s", customers[i].name);

customers[i].id = i + 1;

customers[i].flavor = selectFlavor();

customers[i].topping = selectTopping();

}

for (int i = 0; i < numCustomers; i++) {

pthread\_create(&threads[i], NULL, counter\_thread, (void \*)&customers[i]);

}

for (int i = 0; i < numCustomers; i++) {

pthread\_join(threads[i], NULL);

}

printf("Total Revenue: RS%.2f\n", revenue);

return 0;

}

void \*counter\_thread(void \*arg) {

customer \*cust = (customer \*)arg;

double bill = 0.0;

char log\_msg[256];

printf("\nCustomer %s entered the counter\n", cust->name);

while (1) {

pthread\_mutex\_lock(&flavorMutex);

if (cust->flavor > 0 && cust->flavor <= NUM\_FLAVORS && flavors[cust->flavor - 1] > 0) {

flavors[cust->flavor - 1]--;

pthread\_mutex\_unlock(&flavorMutex);

bill += pricesFlavors[cust->flavor - 1];

printf("\nCustomer %s chose flavor %d, costing RS%.2f\n", cust->name, cust->flavor, pricesFlavors[cust->flavor - 1]);

break;

} else {

pthread\_mutex\_unlock(&flavorMutex);

printf("\nSorry %s, the selected flavor is not available. choose another:\n ",cust->name);

cust->flavor = selectFlavor();

}

}

while (1) {

pthread\_mutex\_lock(&toppingMutex);

if (cust->topping > 0 && cust->topping <= NUM\_TOPPINGS && toppings[cust->topping - 1] > 0) {

toppings[cust->topping - 1]--;

pthread\_mutex\_unlock(&toppingMutex);

bill += pricesToppings[cust->topping - 1];

printf("Customer %s added topping %d, costing RS%.2f\n", cust->name, cust->topping, pricesToppings[cust->topping - 1]);

break;

} else {

pthread\_mutex\_unlock(&toppingMutex);

printf("Sorry %s, the selected topping is not available. choose another:\n ",cust->name);

cust->topping = selectTopping();

}

}

sprintf(log\_msg, "Customer %s - Flavor: %d, Topping: %d, Total Bill: RS%.2f", cust->name, cust->flavor, cust->topping, bill);

syscall(SYS\_MY\_CUSTOM\_LOG, cust->id, log\_msg); // Use syscall to log message

pthread\_mutex\_lock(&revenueMutex);

revenue += bill;

pthread\_mutex\_unlock(&revenueMutex);

printf("Customer %s left the counter\n", cust->name);

return NULL;

}

int selectFlavor() {

int choice;

printf("Please select a flavor [1-Vanilla, 2-Chocolate, 3-Strawberry, 4-Mint, 5-Coffee]: ");

scanf("%d", &choice);

printf("Selected Flavor: %d\n", choice);

return (choice >= 1 && choice <= NUM\_FLAVORS) ? choice : -1;

}

int selectTopping() {

int choice;

printf("Please select a topping [1-Sprinkles, 2-Chocolate chips, 3-Whipped cream, 4-Caramel sauce, 5-Nuts]: ");

scanf("%d", &choice);

printf("Selected Topping: %d\n", choice);

return (choice >= 1 && choice <= NUM\_TOPPINGS) ? choice : -1;

}

**System call code:**

#include <linux/kernel.h>

#include <linux/syscalls.h>

#include <sys/sys.h>

#include <kernel.h>

asmlinkage long sys\_icecream\_log(int user\_id, const char \*user\_msg){

printk("System call initiated");

printk("\nIceCreamLog User[%d]: %s\n", user\_id, user\_msg);

return 0;

}

* 1. **Reference:**
* https://drive.google.com/file/d/1FSl59Q5Fh\_EjJ5baQnYqfd2gn20dYWdw/view?usp=sharing

**6. Conclusion:**

The Frozen Dessert Facility Simulation System Call project promises to bridge the gap between theoretical concepts taught in operating systems courses and their practical application in real-world scenarios. By simulating an ice cream shop's operations, we anticipate contributing to a deeper understanding of how operating system principles can optimize business processes and enhance customer satisfaction. This project underlines the educational value of applying theoretical knowledge to practical situations, preparing students for real-world challenges they may face in their careers.