University of Saint Thomas, Maputo, Mozambique  
Computer Science 3rd year  
Parallel Computing   
Lecturer: Lars Lamos

Assignment 1

Supermarket cashier Parallelism Documentation

Group Members : Erick Wasonga , Kainza Scovia, Daniel Darsamo

Contents

[Documentation of Supermarket Cashier code showcasing parallelism 2](#_Toc169105476)

[Code Structure 2](#_Toc169105477)

[Headers and Definitions 2](#_Toc169105478)

[Data Structures 3](#_Toc169105479)

[Helper Function 3](#_Toc169105480)

[Checkout Function 3](#_Toc169105481)

[Utility Functions 4](#_Toc169105482)

[Main function 6](#_Toc169105483)

[Execution flow 7](#_Toc169105484)

[Key Points 7](#_Toc169105485)

[Output example 8](#_Toc169105486)

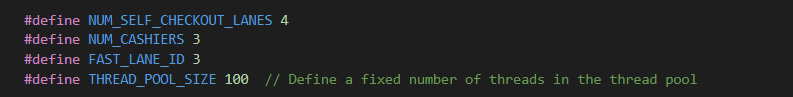
# Documentation of Supermarket Cashier code showcasing parallelism

This program simulates a supermarket checkout process with a combination of self-checkout lanes and cashier lanes. It uses a fixed-size thread pool to manage concurrent customer checkouts. The main features of the program include:

* Four self-checkout lanes.
* Three cashier lanes.
* A fast lane for customers with 15 items or less.
* Dynamic customer distribution across lanes.
* Semaphore-based synchronization to manage thread pool and lane access.

## Code Structure

### Constants

* **NUM\_SELF\_CHECKOUT\_LANES:** The number of self-checkout lanes (4).
* **NUM\_CASHIERS:** The number of cashier lanes (3).
* **FAST\_LANE\_ID:** The ID for the fast lane (3).
* **THREAD\_POOL\_SIZE:** The number of threads in the thread pool (100).

## Data Structures

### CustomerInfo

Holds information about each customer:

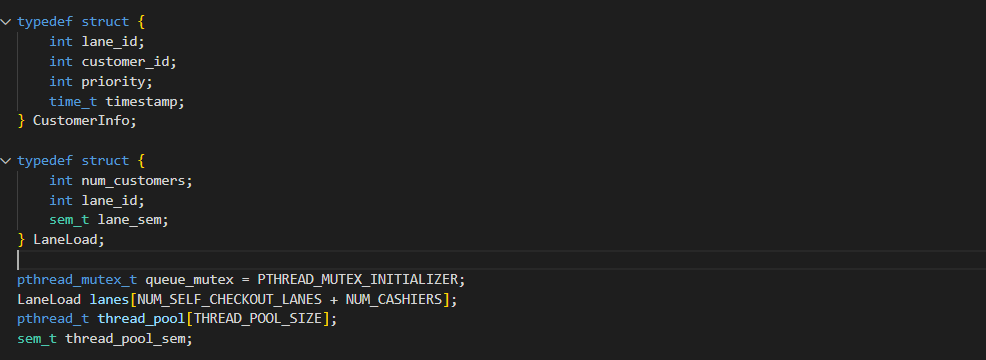
* **int lane\_id:** The ID of the lane the customer is using.
* **int customer\_id:** The unique ID of the customer.
* **int priority:** The priority of the customer (unused in current implementation).
* **time\_t timestamp:** The timestamp of the customer's checkout start time.

### LaneLoad

Holds information about each lane's load:

* **int num\_customers:** The number of customers processed in the lane.
* **int lane\_id:** The ID of the lane.
* **sem\_t lane\_sem:** A semaphore to control access to the lane.

## Global Variables

* **pthread\_mutex\_t queue\_mutex:** A mutex to control access to the customer queue.
* **LaneLoad lanes [NUM\_SELF\_CHECKOUT\_LANES + NUM\_CASHIERS]:** An array of lane loads.
* **pthread\_t thread\_pool[THREAD\_POOL\_SIZE]:** An array of threads forming the thread pool.
* **sem\_t thread\_pool\_sem:** A semaphore to manage the availability of threads in the pool.

## Functions

### get\_current\_time

* Populates the provided buffer with the current time formatted as YYYY-MM-DD HH:MM: SS.

### checkout

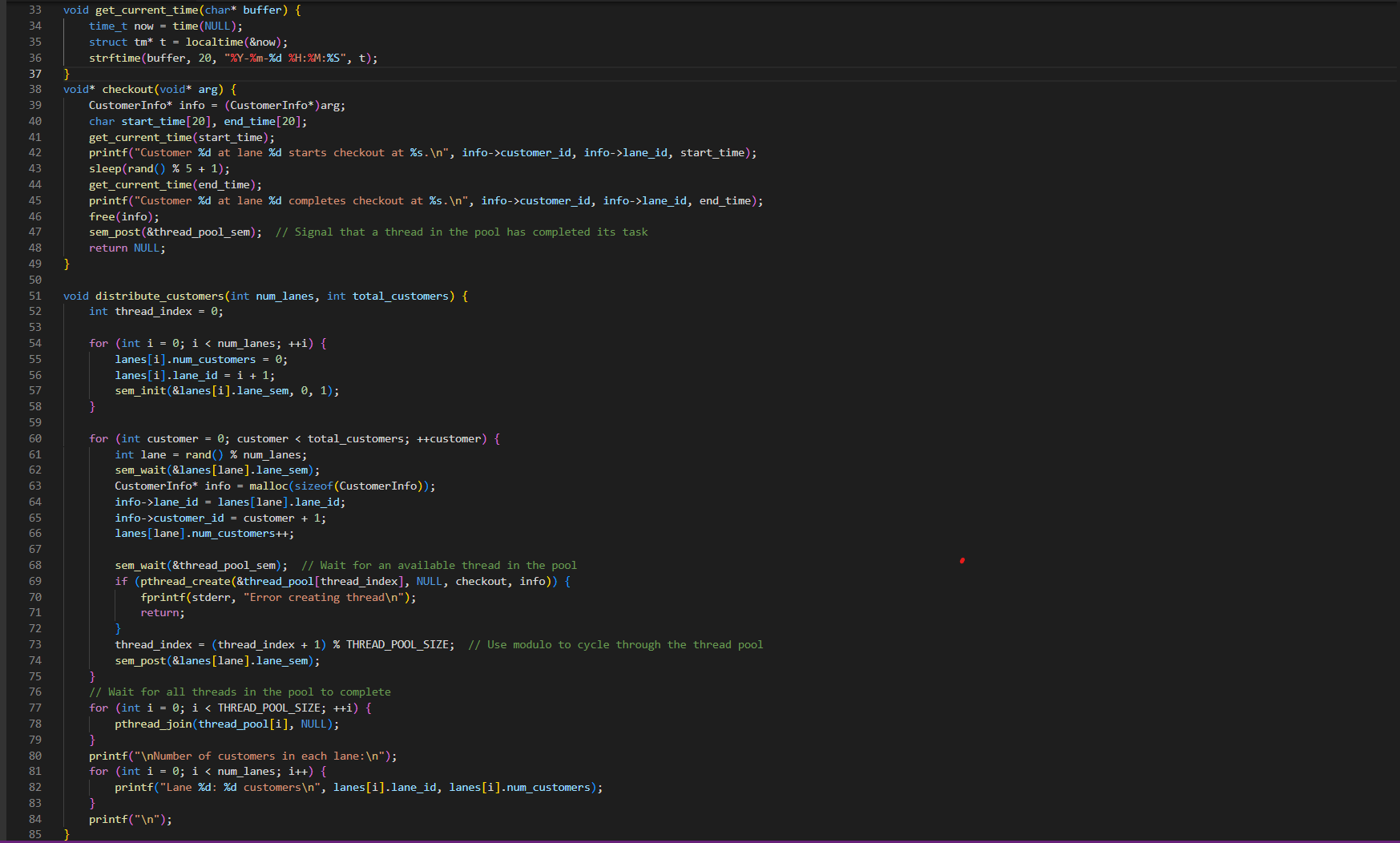
* Processes the checkout for a customer.
* Prints start and end times for the checkout process.
* Frees the allocated memory for CustomerInfo.
* Signals that a thread in the pool has completed its task using sem\_post.

### distribute\_customers

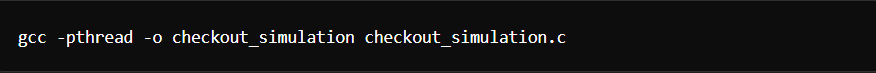
* Initializes lanes and semaphores.
* Distributes customers randomly across the available lanes.
* Creates threads from the pool to handle each customer's checkout.
* Waits for all threads to complete.
* Prints the number of customers processed in each lane.

## Main Function

* Initializes random number generator and semaphore for the thread pool.
* Prompts the user for the number of customers for self-checkout lanes, the fast lane, and normal lanes.
* Calls distribute\_customers to process customers for self-checkout lanes and cashier lanes.
* Destroys the semaphore for the thread pool.



# Usage

To compile the program, use a C compiler such as gcc with the following command:

To run the compiled program, use the following command:



Follow the prompts to enter the number of customers for each lane type. The program will output the start and end times for each customer's checkout and the total number of customers processed in each lane.

After running the program and entering the number of customers for each lane type, the output will detail the start and end times for each customer's checkout, followed by the total number of customers processed in each lane. For example:

