Design Document for Airline Check-in System (ACS)

Evan Strasdin - V00907185
CSC360, Assignment 2 - ACS

1. Number of Threads

Question: How many threads are you going to use?

Answer: The system will use a total of

$$\text{NUM\_CLERKS} + \text{num\_customers}$$

threads, where

$$\text{NUM\_CLERKS}$$

is the number of clerk threads (5 in this case), and

$$\text{num\_customers}$$

is the number of customer threads, which will be determined by the input file.

2. Thread Tasks

Question: Specify the task that you intend each thread to perform.

Answer: - **Customer Threads:** Each customer thread simulates the behavior of a customer arriving at the check-in system, waiting in the queue if necessary, and being served by a clerk. - **Clerk Threads:** Each clerk thread simulates the behavior of a clerk checking the queues for customers and serving them according to the priority rules (business class first, then economy class).

3. Thread Independence

Question: Do the threads work independently? Or, is there an overall "controller" thread?

Answer: The threads work independently. Each customer thread operates based on its arrival time and service requirements. Clerk threads continuously check the queues and serve customers. There is no overall "controller" thread; synchronization is managed using mutexes and condition variables.

4. Mutexes

Question: How many mutexes are you going to use? Specify the operation that each mutex will guard.

Answer: The system uses the following mutexes: - **queue_mutex:** Guards access to the business and economy queues. - **waiting_time_mutex:** Guards

access to the waiting time statistics. - **clerk_assignment_mutex:** Guards the assignment of clerks to customers and ensures that only one clerk is assigned at a time.

5. Main Thread Activity

Question: Will the main thread be idle? If not, what will it be doing?

Answer: The main thread will not be idle. It will: 1. Read customer data from the input file. 2. Initialize data structures and create customer and clerk threads. 3. Wait for all customer threads to finish. 4. Signal the termination of the program and wait for all clerk threads to finish. 5. Clean up resources and print final statistics.

6. Customer Representation

Question: How are you going to represent customers? What type of data structure will you use?

Answer: Customers are represented using a `struct customer_info` that holds their unique ID, class type, service time, arrival time, and waiting time. An array of these structures will be used to store all customer information.

7. Data Structure Concurrency

Question: How are you going to ensure that data structures in your program will not be modified concurrently?

Answer: Concurrent modification of data structures is prevented using mutexes. The `queue_mutex` ensures that only one thread can modify the queue lengths and statuses at a time. The `waiting_time_mutex` ensures that waiting time statistics are updated safely. The `clerk_assignment_mutex` ensures that clerk assignments are handled without race conditions.

8. Condition Variables

Question: How many convars are you going to use? For each convar: (a) Describe the condition that the convar will represent. (b) Which mutex is associated with the convar? Why? (c) What operation should be performed once `pthread_cond_wait()` has been unblocked and reacquired the mutex?

Answer: - **queue_convar[2]:** - **Condition:** Represents the availability of a clerk for the respective queue (business or economy). - **Associated Mutex:** `queue_mutex` for the respective queue to ensure that the condition is checked and modified safely. - **Operation:** Once `pthread_cond_wait()` is unblocked, the thread will check if a clerk is available and update the queue status and lengths accordingly.

9. Overall Algorithm

Question: Briefly sketch the overall algorithm you will use.

Answer: 1. **Main Function:** - Read customer data from the input file. - Initialize data structures and create customer and clerk threads. - Wait for all customer threads to finish. - Signal program termination and wait for all clerk threads to finish. - Clean up resources and print final statistics.

2. **Customer Thread Function:**
    - Simulate arrival time.
    - Attempt to get an available clerk.
    - If no clerk is available, enter the appropriate queue and wait.
    - Once a clerk is available, calculate waiting time and update statistics.
    - Simulate service time and mark the clerk as available after service.
3. **Clerk Thread Function:**
    - Continuously check the queues for customers.
    - Serve business class customers first, then economy class customers.
    - Signal customers in the queue when a clerk becomes available.