CmpE 322 - Operating Systems Project#3 - Simulation of a Fun Fair Payment System

Alperen Dağı

2019400138

This is a C program that simulates the operations of ticket vending machines and customers.

The program defines two constants: NUM_MACHINES and NUM_COMPANIES. NUM_MACHINES specifies the number of ticket vending machines in the simulation, and NUM_COMPANIES specifies the number of companies that customers may pay to.

The program also defines a global array of strings called companies that contains the names of the companies, and a global variable called num_customers that stores the number of customers in the simulation. The output_file variable is a global variable that stores a pointer to the output file for the program.

The program defines two structs: Machine and Customer.

The Machine struct represents a ticket vending machine and has the following fields:

- company_id: The ID of the company associated with the machine.
- balance: The amount of money currently in the machine.
- mutex: A mutex to synchronize access to the machine's data.
- machine_id: The ID of the machine.
- currentCustomer: The ID of the current customer being serviced by the machine.
- total_customer: The total number of customers that will be serviced by the machine.

The Customer struct represents a customer and has the following fields:

- company_id: The ID of the company the customer is paying to.
- amount: The amount of money the customer is paying.
- sleep_time: The amount of time the customer will sleep before using the machine.
- machine_id: The ID of the machine the customer will use.
- customer_id: The ID of the customer.

The program also defines a struct called BankAccount which represents a bank account and has the following fields:

- name: The name of the account holder.
- balance: The balance of the account.
- mutex: A mutex to synchronize access to the account data.

The program has two global arrays: machines and bank_accounts. machines is an array of 10 Machine structs, and bank_accounts is an array of 5 BankAccount structs.

The program has three functions: customer_thread_func, machine_thread_func, and main.

- **customer_thread_func** is a function that simulates the actions of a customer. It takes a void pointer as an argument and returns a void pointer. The function first casts the void pointer to a Customer pointer and assigns it to a local variable called customer. It then sleeps for the customer's sleep_time, which is specified in the Customer struct. After waking up, the function acquires the mutex associated with the machine specified in the machine_id field of the Customer struct. It then updates the balance and company_id fields of the Machine struct to reflect the customer's payment.

- **machine_thread_func** is a function that simulates the actions of a ticket vending machine. It takes a void pointer as an argument and returns a void pointer. The function first casts the void pointer to a Machine pointer and assigns it to a local variable called machine.
  The function then enters a loop that continues until all customers have been serviced by the machine. If the machine's company_id field is not set to -1, it indicates that the machine has a customer waiting to be serviced.
  The function retrieves the BankAccount struct associated with the company from the bank_accounts array, and acquires the mutex associated with the account. It then updates the account balance and the machine's balance to reflect the customer's payment. It releases the mutex associated with the account and the machine, and decrements the total_customer field of the Machine struct to keep track of the number of customers that have been serviced.

- **main** is the main function of the program. It first checks that an input file was provided as a command line argument. If not, it prints an error message and returns 1.

It then opens the input file and reads the number of customers from the first line. It then creates num_customers Customer structs and reads the data for each customer from the input file.

It then initializes the machines array and the bank_accounts array. It also creates 10 threads, one for each machine, and passes each thread a pointer to the corresponding Machine struct as an argument.

Finally, it creates num_customers threads, one for each customer, and passes each thread a pointer to the corresponding Customer struct as an argument. It then waits for all threads to complete and closes the output file.