Chandranshu Rao

CS 4348.001 - Ozbirn

2017 September

Operating Systems Concepts: Project 2 Design

I. Semaphores

A list of each semaphore, its purpose, and its initial value

1) WaitInTellerLine - this semaphore is used to protect the queue data structure that represents the line for the bank teller. Since we don't want multiple threads changing adding or removing from the queue simultaneously, we use this semaphore as a means of mutual exclusion for the queue. Initial value = 1
2) WaitInOfficerLine - this semaphore is also used to protect the queue data structure, but this is used for the other queue representing the line for the loan officer. Since we don't want multiple threads adding or removing from the queue simultaneously, we use this semaphore as a means of mutual exclusion for the queue. Initial value = 1
3) ReadyForTeller - this semaphore is used to keep Teller threads waiting until a customer thread is ready and waiting in the teller line. When the customer signals that they are ready, the teller threads will unblock and continue execution. Initial value = 0
4) ReadyForOfficer - this semaphore is used to keep the LoanOfficer thread waiting until a customer thread is ready and waiting in the loan officer line. When the customer signals that they are ready, the loan officer threads will unblock and continue execution. Initial value = 0
5) Assigned[] - this is an array of semaphores, with one semaphore for each customer. It is used so that the customer threads halt execution until they are signaled by a teller or officer that they have been assigned an employee. An array is used because each customer needs its own semaphore. Initial values = {0,0,0,0,0}
6) PerformTask[] - this is an array of semaphores, with one semaphore for each customer. It is used so that the customer threads halt execution until they are signaled by a teller or officer that their requested task has been performed. An array is used because each customer needs its own semaphore. Initial values = {0,0,0,0,0}
7) CustomerRequestTeller[] - an array of semaphores for each bank teller. This is used to halt the teller threads until the customer signals the teller that a request was made. An array is used because each teller needs its own semaphore. Initial values = {0,0}
8) CustomerRequestOfficer - a semaphore for the loan officer used to halt the officer thread until the customer signals the officer that a request was made. Initial value = 0

II. Pseudocode

Pseudocode for each class

```
Customer()
        while(visits != 3){
                enterBank(); // Customer thread is created
                getRandomTask(); // Customer is randomly assigned a task
                getRandomAmount(); // Customer is randomly assigned an amount for the task
                wait(waitIn_Line); // Use mutex so that customer can be added to queue
                waitInLine(); // Customer waits in the appropriate queue
                signal(waitIn_Line); // Done with the mutex
                signal(readyFor); // Customer signals that he/she is ready and is waiting in line
                wait(assigned[i]); // Customer waits for employee to be assigned
                requestTask(); // Customer requests task to the appropriate employee
                signal(request); // Customer signals to employee that he/she made a request
                wait(perform_task[i]); // Customer waits for task to be completed by teller/officer
                getReceipt(); // Get receipt from teller/officer
                visits ++;  // Visit again
        }

Teller()
        while (true){
                wait(readyForTeller); // Wait for a customer to be ready
                wait(waitInTellerLine); // Used as mutex so we can access queue
                pullCustomerFromLine(); // Pull a customer from line when customer is ready
                signal(waitInTellerLine); // We are done using the queue
                signal(assigned[i]); // Signal the customer they have been assigned an employee
                wait(request); // Wait for customer request
                processTask(); // Process the task, either deposit or withdrawal
                signal(perform_task[i]); // Signal to the customer that task has been done
        }

LoanOfficer()
        while (true){
                wait(readyForOfficer); // Wait for a customer to be ready
                wait(waitInOfficerLine); // Used as mutex so we can access queue
                pullCustomerFromLine(); // Pull a customer from line when customer is ready
                signal(waitInOfficerLine); // We are done using the queue
                signal(assigned[i]); // Signal the customer they have been assigned an employee
                wait(request); // Wait for customer request
                processLoan(); // Process the loan
```

```
            signal(perform_task[i]); // Signal to the customer that task has been done
        }

Main()
        createSemaphores(); // Initialize all semaphores
        createThreads(); // Create all threads (Customers, Tellers, Officers)
        startThreads(); // Run each thread
        joinThreads(); // Join threads to main
        printSummary(); // Print summary table
```