Operating Systems

**CS4348**

**Project #2: Threads and Semaphores**

**Design of the project –**

1. **Semaphore Details, Purpose and initial value.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Semaphore Details** | | | |
| **No** | **Semaphore Name** | **Initial Value** | **Description** |
| 1 | finished [50] | 0 | Array of 50 semaphores with initial value as 0, this is to ensure the synchronization between the customer threads. Once postal work thread processes the request made by particular customer then only that customer should exit. |
| 2 | customerEntry | 10 | This is to allow 10 customer threads at a time in postal office. That’s why initial value is set to 10, after that each release function on the semaphore will allow one customer in the postal office. |
| 3 | postalWorker | 3 | This semaphore represents 3 postal workers. |
| 4 | mutex1 | 1 | This semaphore is used to allow only one customer thread through the critical section where customer ID is saved in global variable. |
| 5 | mutex2 | 0 | This semaphore is used to allow only one postal worker thread so that this thread will copy value from global variables related to customer ID and Action and set it into local variable so that each thread can have its own value for these two parameters. |
| 6 | mutex3 | 0 | This semaphore is used to allow only one customer thread through the critical section where postal worker ID copied from global variable to local variable so that each customer thread will have its assigned workers ID |
| 7 | mutex4 | 1 | This semaphore is used to allow only one postal worker thread so that this thread will save value to global variables related to postal worker ID so that customer thread can access this value and save into its own local variable. |
| 8 | postalWorkerAssigned | 0 | This semaphore is used to allow customer thread only when customer is assigned to its respective postal worker. Therefore, customer thread will wait for postal worker thread to get ready to serve it. |
| 9 | customerRequested | 0 | This semaphore is used to allow postal worker thread to wait for customer thread to request an action such as mailing a letter, mailing a package or buying a stamp. |
| 10 | scale | 1 | This semaphore is used for synchronizing the scale resource. Only one postal worker thread is allowed to enter into the critical section to use scale. |

1. **Pseudo-code for each component –**
2. **Post office main()**

|  |
| --- |
| //Post office simulator  int customer\_max\_capacity = 50;  int postal\_worker\_max\_capacity = 3;  Semaphore finished [50] = {0};  Semaphore customerEntry = 10, postalWorker = 3;  Semaphore mutex1 = 1, mutex2 = 0, mutex3=0,mutex4=1;  Semaphore postalWorkerAssigned = 0, customerRequested = 0, scale = 1;  int customerID, postalWorkerID, custActionID; |

1. **Customer()**

|  |
| --- |
| void customer(){  int custNum, custActionNum, postalWorkerNum;  wait(customerEntry);  enterPostOffice();  wait(mutex1);  customerID = custNum;  custActionID = custActionNum;  signal(mutex2);  wait(mutex3);  postalWorkerNum = postalWorkerID;  signal(mutex4);  wait(postalWorkerAssigned);  customerRequestTask(custNum, custActionNum, postalWorkerNum);  signal(customerRequested);  wait(finished[custNum]);  customerTaskCompletionNotify(custNum, custActionNum);  signal(customerEntry);  } |

1. **PostalWorker()**

|  |
| --- |
| Void PostalWorker(){  int custID, postalWrkrID, custActionNum;  wait(postalWorker);  wait(mutex2);  custID = customerID;  custActionNum = custActionID;  wait(mutex4);  postalWorkerID = postalWrkrID;  signal(mutex3);  signal(mutex1);  assignCustomerToWorker();  signal(postalWorkerAssigned);  wait(customerRequested);  workOnCustomerTask (custID, custActionNum, postalWrkrID);  notifyOnFinishedTask();  signal(finished[custNum]);  signal(postalWorker);  } |

1. **customerTaskCompletionNotify()**

|  |
| --- |
| void customerTaskCompletionNotify(){  switch(action){  case 0: finishedBuyingStamps();  case 1: finishedMailingLetter();  case 2: finishedMailingPackage();  }  } |

1. **customerRequestTask()**

|  |
| --- |
| Void customerRequestTask(){  switch(action){  case 0: requestTaskForBuyingStamps();  case 1: requestTaskForMailingLetter();  case 2: requestTaskForMailingPackage();  }  } |

1. **workOnCustomerTask()**

|  |
| --- |
| Void workOnCustomerTask(){  switch(action){  case 0: workOnStampsTask(1000);  case 1: workOnMailingLetter(1500);  case 2: wait(scale);  useScale();  workOnMailingPackageUsingScales(2000);  releaseScale();  signal(scale);  }  } |