Summary of Threads and Semaphores

Semaphore is an integer used for signaling among processes, only three operations may be performed on a semaphore, which are: initialize, decrement, and increment. The thread provides the mechanism for running a task, we can launch multiple threads from a program concurrently. The threads can be executed simultaneously in multiprocessor system.

In this project I learn the coordination of multiple threads using semaphores, which is used to simulate a post office with postal workers, customers and types of service. The idea behind this small project is to use Java threads and Java semaphore (java.util.concurrent.Semaphore). The purpose of this project is to use thread and semaphores to model customer and employee behavior with 50 customers threads and three postal worker threads. The postal workers and customers are threads that each act independently of one another to achieve realistic simulation. For the customer thread, the customer will wait until there is an open spot in postal worker, signified by semaphore “max\_capcity.acquire”. Once one become available the customer take that spot. Once the postal worker has accepted the customer, the customer has not gained control over a shared variable for the customer’s information. This prevents other customers from overwriting this data for other postal workers to use until the customer’s postal worker has read that information. Then the customer simply waits on the postal worker to complete the service. For postal worker thread, has less required actions than that of a customer because they are not involved with obtaining access to the line. As a result the semaphore of importance to the postal worker is the “cust\_ready” variable that notifies the customers that the postal worker is available for interaction. After this there is a reflection of the exchange from the Customer.run() method where the postal worker and the customer are exchanging data while protecting it from other postal workers. This passed information contains the type of service to be rendered by the postal worker. Once the service is complete the customer is released from awaiting the service and a new customer is waited upon.

The function of each of the threads are as followed:

Customer:

1. 50 customers visit the Post Office (1 thread per customer up to 50), all created initially.
2. Only 10 customers can be inside the Post Office at a time.
3. Each customer upon creation is randomly assigned one of the following tasks:
   1. buy stamps
   2. mail a letter
   3. mail a package
4. Times for each task are defined in the task table.

Postal Worker:

1. 3 created initially, one thread each.
2. Serves next customer in line.
3. Service time varies depending on customer task.

The project was implemented in Java language with Unix Operating System platform. The project was tested on school unix server. The project was implemented in the following steps:

1. The design and pseudo code for the project was designed
2. The functions containing the tasks to be performed by the various types of threads were designed
3. 50 customer threads, 3 postal worker threads were created
4. Semaphores are created to maintain concurrency of threads
5. Print each thread when it is created and when it is joined
6. The customer threads prints customer actions including buy stamps, mail letter, mail package. The postal worker threads prints postal worker action in loop

In this project, I speed time reading about knowledge of threads and semaphores. I also learn how to use Java to implement threads and semaphores. After these stage, I began to design the project and code the project in stages. Also I check the output at each stage to ensure that the project was working properly up to that stage. During the project, I have some difficulty in maintain the synchronization and the design of semaphore of different thread. I have to debug the code very carefully because of the concurrent threads. The result is expected as discussed in the document, customers are concurrently executing.