****

**Course Name:**

Operating System And System Programming Lab

**Train Synchronization Using Semaphores**

**Semester:** IV

**Batch:** B8

**Submitted By:**

Arjun Goyal – 17103294

Rhythm Gupta – 17103310

Sankalp Biswal – 17103314

Pranav Sachdeva – 17103318

**PROBLEM STATEMENT:-**

There are ‘n’ no. of trains on the train route. There are three types of trains: local, mail and special. Special train has the highest priority, local train has the middle priority and mail train has the lowest priority. Every train has a specific route from its source to destination which is taken in the form of track sequences on which the train will go to reach its destination. There can be only one train on a track at a time. So the problem is to synchronize the trains on the basis of their track routes and their priorities.

|  |  |
| --- | --- |
| **Train Type** | **Priority** |
| Special | 1 |
| Local | 2 |
| Mail | 3 |

**PROBLEM SOLVING METHODOLOGY**

All the ‘n’ trains are ‘n’ no of threads running concurrently. And there is one semaphore associated with each train. Each train has its own thread and its track route. There is one more thread which starts running after all the trains are started from their first tracks. This thread keeps running until all the trains reaches their destinations. This thread checks for the trains which are waiting for a track and are having the highest priority. It then signals these trains to go for their next tracks if they are available.

**Method Used:-** We have used semaphores and threads to solve the problems and create synchronization between the processes.

**Semaphores**

They are simple locks which allow multiple processes to build up correctly and efficiently using certain instances of some shared resources. They are of two types– binary and counting semaphores. Binary semaphores are used when there is only one instance of the shared resource and counting semaphores are used in case of shared resources having more than one instance.

A semaphore is as an object with an integer value in C and can be manipulate with two routines sem\_wait() and sem\_post().

The sem\_wait() function decrements the semaphore value or waits for the value to be greater than zero. The sem\_post() increments it.

int sem\_init(sem\_t \**sem*, int *pshared*, unsigned int *value*);

#include <semaphore.h>

sem\_t *sem*;

int *pshared*;

int *ret*;

int *value*;

/\* initialize a private semaphore \*/

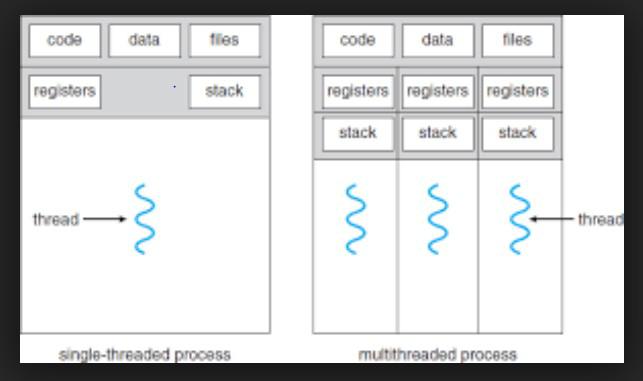
*pshared* = 0;

*value* = 1;

*ret* = sem\_init(&*sem*, *pshared*, *value*);

**Threads**

A thread is a single sequence stream within in a process. They are light weight process. In a process, threads allow multiple executions of streams.



**Mutexes**

A mutex provides mutual exclusion. In computer programming, a mutex is a

program allows multiple program threads to share the same resource, such as file

access, but not simultaneously.

**Practical Application:-**



This is an example of a complex railway track network (Delhi Metro Route Map).

It is required to synchronize or schedule the trains properly for a safe travelling experience.

Train synchronization is necessary to avoid clashes and to ensure the proper arrival and departure of trains.

We use the same route map to give inputs to our system.

**Screenshot of the output with sample input:**

