COMP304 Project 2

Hakan Ayral

Spring 2024

Alperen Kars (akars20)

This project includes two files: **pthread\_sleep.c** (given), **air\_traffic\_control.cpp**

The makefile is specified as **air\_traffic\_control**. It can also be created by using the following command:

g++ -o air\_traffic\_control air\_traffic\_control.cpp pthread\_sleep.c -pthread

The makefile can be run by using the following command for a 60 seconds of simulation with n=5 and probabiliy of arrival of a plane p=0.5.

./air\_traffic\_control -s 60 -p 0.5 --seed 1234 -n 5

Files

**pthread\_sleep.c**

No changes are introduced to pthread\_sleep.c. It serves as an auxiliary function to air\_traffic\_control.cpp and helps to sleep threads without using the built-in sleep function.

**air\_traffic\_control.cpp**

This file is designed according to the structure given in the handout. There exist two queues for landing and departing queues, named accordingly.

It holds two mutexes:

**queueMutex:** this mutex is used to protect access to queues by different threads, so that only one thread can access to shared data.

**runwayCond:** this mutex works in accordance with condition variable. It waits for the condition of the planes to be true, for instance waiting for a plane or changing its status.

Functions in air\_traffic\_control.cpp

**void logPlane:** it is simply used to write to the log file according to the given format in the handout.

**void\* planeThread:** this function creates a thread for a specific plane, locks the queue mutex accordingly and then adds that plane to relevant queue by checking its status. Then this function sends a signal to **towerThread** about a change in the plane’s status and unlocks the mutex again so that another thread can access and modify it.

**void\* towerThread:** this function is mainly the tower version of planeThread, it creates a thread for the tower. The main difference between this and previous function is that this function runs until one condition requests it to break.

It first locks the queue mutex, then checks if stop condition is true and both landing and departing queues are empty (this choice is made deliberately to be able to track the status of the planes from the logfile. If a thread is cancelled before it has been handled, it might not be written to logfile as required). If they are not empty, it waits until a new plane arrives by changing the mutexes.

In case of a new plane, it first records the arrival time and determines the next plane to be processed. However, this raw solution may cause a starvation for some planes, as described below.

Explanation of Starvation: Starvation for landing planes can occur if the air traffic control algorithm always prioritizes departing planes, or vice versa. If the landing planes are always put at the back of the queue or never get the runway when needed, they can be indefinitely waited.

Two conditions are added here to solve starvation problem. First of them checks if one of the planes is waiting for a time longer than a predetermined **max\_wait\_time** (for this project, it is 10), it is serviced first. The second of these conditions switches between departing and landing queues to serve the planes in both of them fairly. Then the mutexes are unlocked again and the system is waited for 2t times by using **pthread\_sleep** function from the given file.

**void\* snapshotThread**: this function is created particularly for Keeping Logs part, and it simply outputs the planes that are on the ground and on the air at every second, after a given command line argument n seconds. This also handles the locking and unlocking of the mutexes accordingly.

**int main:** The main task of this function is to parse the command line arguments, open the log file, generate the random seed for the calculation of the probability of a plane thread creation, create the plane and tower threads and assign planes their properties such as status, ID, and time. It lastly runs the simulation for a given -s seconds and waits for all threads to finish, then closes the log file.