COMP304 PROJECT – 2

Air Traffic Control Simulation

Team: “nolodiga”

Members:

Tuna Çimen – 80418

Mehmet Tuna Günenç – 79932

Introduction:

We utilized a virtual machine running Ubuntu Linux to test our implementation thoroughly. To facilitate this process, we securely transferred the code to the Linux machine using SSH. This approach allowed us to efficiently conduct our testing in a controlled environment.

We implemented an air traffic control simulator with one runway and 2 seconds of runway control time.

HOW TO RUN: You can run using "./main <seconds> <probability>" or you can recompile from scratch.

You can see the example log file Example_Log.txt for n=60 and probability 0.5 (60 seconds and 0.5 probability). We did not screenshots since it is too long and does not fit the screen.

" ./main 60 0.5 "

Part 1:

Planes are implemented as threads and have their control function. Inside this function every plane with a certain probability requests landing if flying and departing if they are landed. Since there is a mutex lock called comm_lock only one plane at a time can contact the tower for landing or departing. There is another lock for the runway that allows only one plane to use the runway at a time. Another thread is responsible for counting the seconds passed. For fuel efficiency, the initial implementation always favored landing ones and only allowed took-offs after 3 planes landed. However, this caused starvation since sometimes planes could not take off because they were waiting for some plane to land but no plane was in the landing queue. In part 2, we solved it.

Part 2:

To avoid take-off starvation, added a maximum waiting time of 15 seconds so after some plane waits to take off more than 15 seconds the starve mutex lock is unlocked so planes that wait for taking off can take off. After any takeoff starve_lock is locked so only after either 3 planes landed or 15 seconds passed since waiting, it can be unlocked.

Part 3:

For every request, as well as for every landing and departure, we log the entry meticulously. At the end of the specified n seconds, these logged entries are compiled and printed in a straightforward format. This process is clearly demonstrated in the example log file, which showcases how the entries are organized and recorded for review.

References

Comp304 PS Slides and Codes

https://www.geeksforgeeks.org/

https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html

https://stackoverflow.com/

https://www.geeksforgeeks.org/mutex-lock-for-linux-thread-synchronization/

https://hpc-tutorials.llnl.gov/posix/