COMP 304 Project 2: Spacecraft Control with Pthreads

Yakup Enes Güven

Part I

In this part, I created 2 different fields for the launch pads with the help of threads and mutexes. I did this thanks to mutex, thereads, and queues. I aimed to make it easier by adding only duration and limit in the queue.c file. I first created a log file to store all the data. In main I initialize all mutex, threads and queues first. I then used them in appropriate functions. I used the same pseudo code given in ps for the job parts. In the controlTower section, I first made the controls, and then I locked and unlocked the threds accordingly. Again, I used the given pseudocode for PadA and PadB. I used PrintQueue to write the desired queue to the log file. I used 'WriteLog' to write the desired evolutions to the log file. I used QueuePrinter to write all transactions to terminal.

```
comp304@comp304-VirtualBox:~/Desktop/project2$ gcc -pthread -o project_2 project_2.c
comp304@comp304-VirtualBox:~/Desktop/project2$ ./project_2
At 30 sec landing: empty
At 30 sec launch: empty
At 30 sec assembly: empty
At 30 sec padA: 827
At 30 sec padB: 570
At 32 sec landing: empty
At 32 sec launch: empty
At 32 sec assembly: empty
At 32 sec padA: 123
At 32 sec padB: 570
At 34 sec landing: empty
At 34 sec launch: empty
At 34 sec assembly: empty
At 34 sec padA: empty
At 34 sec padB: 570
At 36 sec landing: empty
At 36 sec launch: empty
At 36 sec assembly: empty
At 36 sec padA: 635
At 36 sec padB: 570
At 38 sec landing: empty
At 38 sec launch: 54
At 38 sec assembly: empty
At 38 sec padA: 635 26
At 38 sec padB: 570
At 40 sec landing: empty
At 40 sec launch: 54
At 40 sec assembly: empty
At 40 sec padA: 26 687
At 40 sec padB: 570
^C
```

Part II

In this part, I used the same code of the first part, but I made some changes so that there would be no starvation. Since a task that is thrown to the end according to the task load causes

starvation, I tried to deal with it by giving priority to controlTower if anyone is in this situation.

```
comp304@comp304-VirtualBox:~/Desktop/project2$ gcc -pthread -o project_2_part2 project_2_part2.c
comp304@comp304-VirtualBox:~/Desktop/project2$ ./project_2_part2
At 30 sec landing: 418 747 173 268 181 512 583 965 712 658
^C
```

Part III

In this part, I used a code like the first batch code. However, I have prioritized 2 different situations here. For the A and B pads, I adjusted and created them according to my emergency situation. The working style is almost like part1, but it only considers the emergency situation.

```
comp304@comp304-VirtualBox:~/Desktop/project2$ gcc -pthread -o project_2_part3 project_2_part3.c
comp304@comp304-VirtualBox:~/Desktop/project2$ ./project_2_part3
At 30 sec landing: empty
At 30 sec launch: empty
At 30 sec assembly: empty
At 30 sec padA: 137
At 30 sec padB: 570
At 32 sec landing: empty
At 32 sec launch: empty
At 32 sec assembly: empty
At 32 sec padA: 78
At 32 sec padB: empty
At 34 sec landing: empty
At 34 sec launch: empty
At 34 sec assembly: empty
At 34 sec padA: 78
At 34 sec padB: 245
At 36 sec landing: empty
At 36 sec launch: empty
At 36 sec assembly: empty
At 36 sec padA: 272
At 36 sec padB: empty
At 38 sec landing: empty
At 38 sec launch: empty
At 38 sec assembly: empty
At 38 sec padA: 272
At 38 sec padB: empty
At 40 sec landing: empty
At 40 sec launch: empty
At 40 sec assembly: empty
At 40 sec padA: 547
At 40 sec padB: empty
At 42 sec landing: empty
At 42 sec launch: empty
At 42 sec assembly: empty
At 42 sec padA: 547
At 42 sec padB: empty
At 44 sec landing: empty
At 44 sec launch: empty
At 44 sec assembly: empty
At 44 sec padA: empty
At 44 sec padB: empty
^C
comp304@comp304-VirtualBox:~/Desktop/project2$
```

Git link : https://github.com/yguven17/project2