Comp 304 Project II Report

Mislina Akça - 80006 & Duru Tandoğan - 79479

PART I

The simulation is compiled with the commands:

- gcc pthread_sleep.c -o pthread_sleep
- ./pthread_sleep -s 60 -p 0.5 -n 20

The corresponding arguments were passed into it, since it was suggested in the project description file, therefore we will demonstrate our example outputs from there.

First, we have a Plane structure containing id, landing/departure type and a request time taken from the initialization of the structure. These structures are created in a function called plane(). This function initializes planes with unique ids, random land or departure types and assigns creation times and puts the plane into the landing or departure queues according to their type.

We have a method for Control Tower which handles all cases in itself. This method takes the simulation time as it's parameter and takes the starting time for each call. It runs the simulation until the current time exceeds the simulating time which enables the use of the real-time.
It contains a while loop with the wait pthread. Until the runway is empty and there are no landing planes or departure planes waiting in the line, it will be stuck in this loop securing the entry to the runway.

If the runaway is available with no planes, entry to the runaway will be available. The source code first checks the priority conditions created for Part II, which will be explained later.

For this part, landing planes will take priority. If there are landing planes existing, checked by the landingCounter, we will execute the landing queue by taking the first plane entered, and granting the lock for the runaway for this plane. The thread mutex will be unlocked for the landing, and the plane will be granted with 2t seconds to land. After landing the thread mutex will be locked and the runway will be modified to be available for the usage of other planes. After all these steps, the current time will be taken and it will be subtracted from the start time. Moreover, request time of the plane will be also substracted from the result of the previous equations. Both results are logged to the file plane & tower logs. This log can be seen with the command:

- cat planes.log
- cat towers.log

The same process will be executed for the departing planes, if no planes are landing.

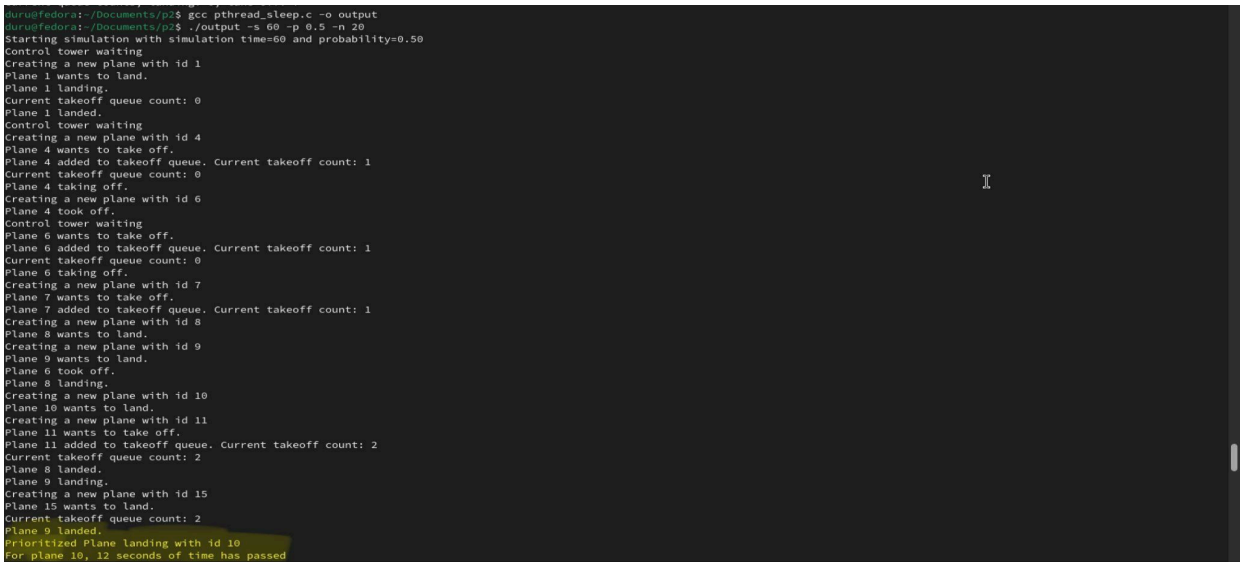
This logical structure overall enables only one plane at the runway once and also favors landing planes over departure planes.

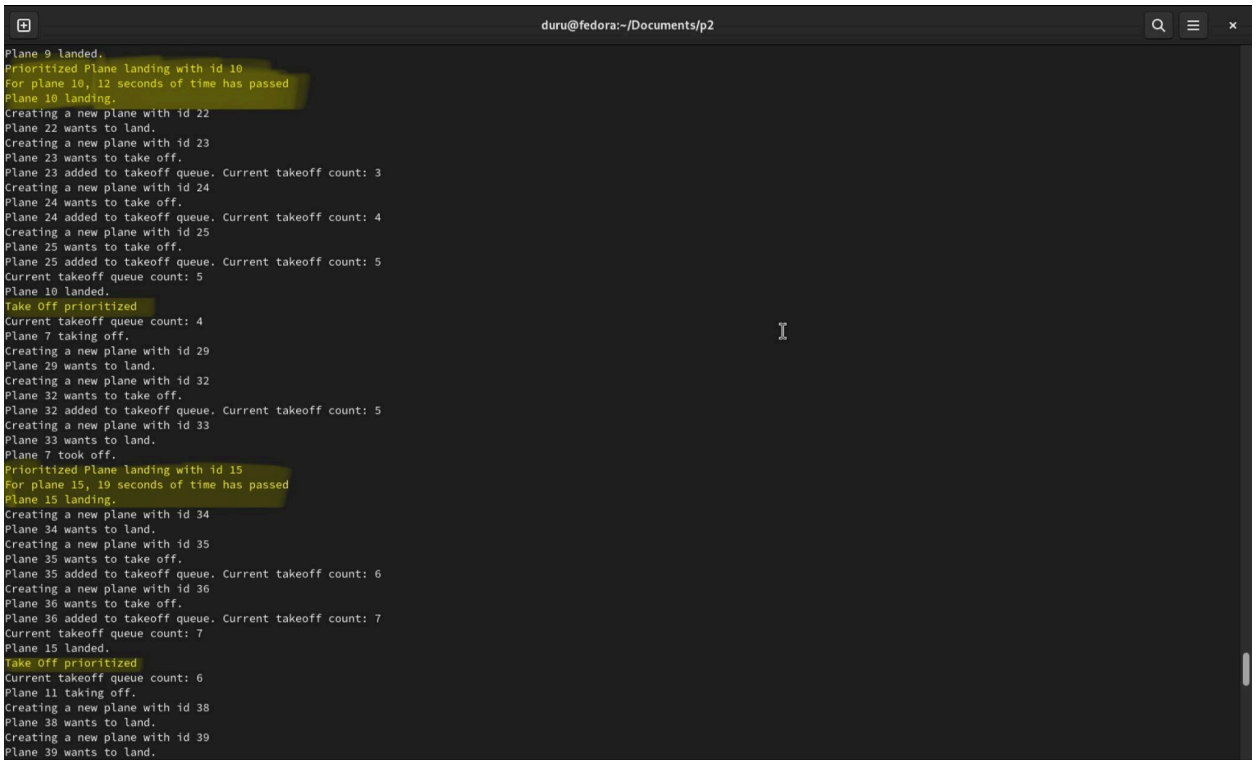
The main function:

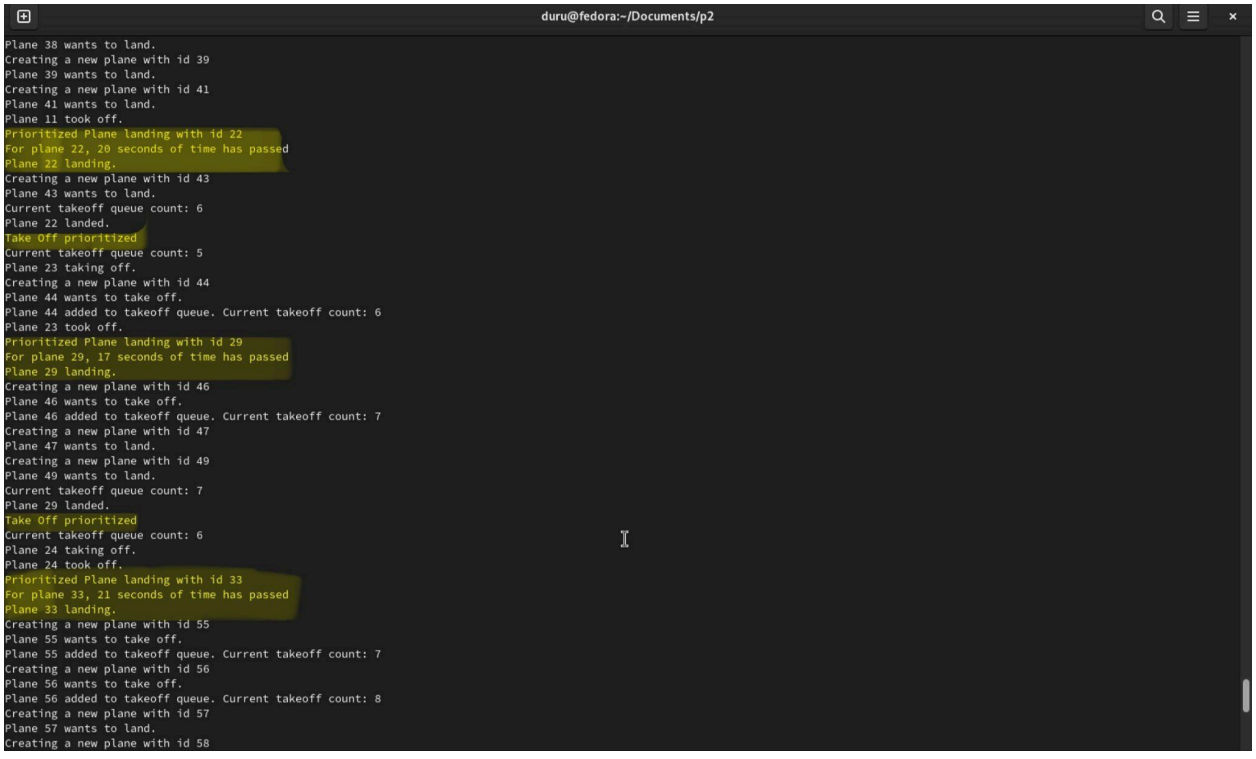
In the main method, we first get the time of the day to initialize the simulation start time. The method first compares the arguments to get the simulation time(-s), probability of plane creation(-p) and the interval of taking the snapshots of the environment(-n). All of these have their initial values if the user does not provide them.
Main method first notifies the user that the simulation is starting with a print statement and it initializes the starting time of the simulation.
It opens log files to keep the planeLog and the towerLog. After opening the log files, main method initializes the mutex lock and condition that we will use as airway and condition to land on in our project.
Main method has a loop that iterates for every second in the simulation time. For each second i, it generates a plane based on the probability variable. It calls the plane method and based on its isLanding attribute, adds it to landing or takeOff queues.
Main method calls pthread_join() method for making the tower thread wait for others. Since Control Tower is the first and main thread, it should start before every other thread and execute until all other threads are finished.
At the end, main method closes the log files and returns 0 for success.

The outputs:

Since the output generations were left to us, we utilized comprehensive print statements at each step of the simulation for easy understanding. We also have two different output files written for the plane and the control tower states at the specified intervals.




duru@fedora:~/Documents/p2
Plane 9 landed.
Prioritized Plane landing with id 10
For plane 10, 12 seconds of time has passed
Plane 10 landing.
Creating a new plane with id 22
Plane 22 wants to land.
Creating a new plane with id 23
Plane 23 wants to take off.
Plane 23 added to takeoff queue. Current takeoff count: 3
Creating a new plane with id 24
Plane 24 wants to take off.
Plane 24 added to takeoff queue. Current takeoff count: 4
Creating a new plane with id 25
Plane 25 wants to take off.
Plane 25 added to takeoff queue. Current takeoff count: 5
Current takeoff queue count: 5
Plane 10 landed.
Take Off prioritized
Current takeoff queue count: 4
Plane 7 taking off.
Creating a new plane with id 29
Plane 29 wants to land.
Creating a new plane with id 32
Plane 32 wants to take off.
Plane 32 added to takeoff queue. Current takeoff count: 5
Creating a new plane with id 33
Plane 33 wants to land.
Plane 7 took off.
Prioritized Plane landing with id 15
For plane 15, 19 seconds of time has passed
Plane 15 landing.
Creating a new plane with id 34
Plane 34 wants to land.
Creating a new plane with id 35
Plane 35 wants to take off.
Plane 35 added to takeoff queue. Current takeoff count: 6
Creating a new plane with id 36
Plane 36 wants to take off.
Plane 36 added to takeoff queue. Current takeoff count: 7
Current takeoff queue count: 7
Plane 15 landed.
Take Off prioritized
Current takeoff queue count: 6
Plane 11 taking off.
Creating a new plane with id 38
Plane 38 wants to land.
Creating a new plane with id 39
Plane 39 wants to land.


duru@fedora:~/Documents/p2
Plane 38 wants to land.
Creating a new plane with id 39
Plane 39 wants to land.
Creating a new plane with id 41
Plane 41 wants to land.
Plane 11 took off.
Prioritized Plane landing with id 22
For plane 22, 20 seconds of time has passed
Plane 22 landing.
Creating a new plane with id 43
Plane 43 wants to land.
Current takeoff queue count: 6
Plane 22 landed.
Take Off prioritized
Current takeoff queue count: 5
Plane 23 taking off.
Creating a new plane with id 44
Plane 44 wants to take off.
Plane 44 added to takeoff queue. Current takeoff count: 6
Plane 23 took off.
Prioritized Plane landing with id 29
For plane 29, 17 seconds of time has passed
Plane 29 landing.
Creating a new plane with id 46
Plane 46 wants to take off.
Plane 46 added to takeoff queue. Current takeoff count: 7
Creating a new plane with id 47
Plane 47 wants to land.
Creating a new plane with id 49
Plane 49 wants to land.
Current takeoff queue count: 7
Plane 29 landed.
Take Off prioritized
Current takeoff queue count: 6
Plane 24 taking off.
Plane 24 took off.
Prioritized Plane landing with id 33
For plane 33, 21 seconds of time has passed
Plane 33 landing.
Creating a new plane with id 55
Plane 55 wants to take off.
Plane 55 added to takeoff queue. Current takeoff count: 7
Creating a new plane with id 56
Plane 56 wants to take off.
Plane 56 added to takeoff queue. Current takeoff count: 8
Creating a new plane with id 57
Plane 57 wants to land.
Creating a new plane with id 58

PART II

For this part, we created an algorithm to prevent the starvation that will be caused by the action of favoring landing planes over the departure ones. Here, it is additionally stated the case of waiting planes for the take off action. (In the case of 5 or more planes waiting to take off, they should be also prioritized.)

This condition will create starvation for the planes in the air. Our solution utilizes the waiting time of each plane. If the difference between the current time and the first plane's request time for the landing action exceeds a determined threshold, this plane will be assigned the first priority. The threshold was chosen after an excessive amount of debugging, which we believe was the optimal amount of time. This algorithm will also make sure to take the oldest plane waiting due to the structure of the queues, and will effectively handle the starvation of the landing processes.

This priority is checked at the start of the simulation and if the criteria are matched, the exact same execution process with the previous part's landing action will take place inside.

Another priority was the case of the takeoff planes which are lined up with a count of 5 or more. These criteria is also taken into consideration as a priority and checked right after the landing plane starvation. If there are 5 or more planes waiting for waiting to take off (controlled by take off counter) assign priority to these planes.

If the conditions are met, the exact same execution process with the previous part's taking off action will take place inside.

If both priority conditions are not satisfied, our algorithm uses the code generated in part 1 and favors landing planes over departure planes.

The outputs:

The parts highlighted in the above screenshots are examples of prioritized actions.

Keeping Log

We have two log files for keeping the logs: planes.log and tower.log. The planes.log keeps the plane no (ID), its status (landing (L) or departing (D) or emergency (E)), the request time to use the runway, the runway time (end of runway usage) and turnaround time (runway time - request time).
For planes.log, in the Control Tower method we log write into the planes.log file every time a plane has granted and left the airway(lock).
For tower.log, in the main method, we log the planes on the ground and air each time an interval passes in simulation time. For instance, if n=20 and s=60, it will log at the 20th and 40th seconds.

The outputs:

```
current queue counts, landing: 7, take off: 7
duru@fedora:~/Documents/p2$ cat planes.log
PlaneID  | Status  | Request Time  | Runway Time  | Turnaround Time
1        | D       | 1             | 1            | 0
4        | L       | 4             | 4            | 0
6        | L       | 6             | 6            | 0
8        | L       | 8             | 10           | 2
9        | D       | 9             | 14           | 5
10       | E       | 10            | 22           | 12
7        | D       | 7             | 22           | 15
15       | E       | 15            | 34           | 19
11       | D       | 11            | 34           | 23
22       | E       | 22            | 42           | 20
23       | D       | 23            | 42           | 19
29       | E       | 29            | 46           | 17
24       | L       | 24            | 46           | 22
33       | E       | 33            | 54           | 21
25       | D       | 25            | 54           | 29
34       | E       | 34            | 70           | 36
32       | L       | 32            | 70           | 38
duru@fedora:~/Documents/p2$ cat tower.log
At 0 sec ground:
At 0 sec air:
At 20 sec ground: 7 11
At 20 sec air: 10 15
At 40 sec ground: 23 24 25 32 35 36
At 40 sec air: 22 29 33 34 38 39
duru@fedora:~/Documents/p2$
```