|  |  |
| --- | --- |
|  |  |
| Simulação de Ponte Aérea |  |
|  | 2021/2022L. Engenharia Informática |
|  | Rafael Remígio 102435João Correia 104360 |
|  | Professor Nuno LauProfessor Guilherme Campos |

|  |  |  |
| --- | --- | --- |
| Índice |  | |
| [***Introdução \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3***](#_bookmark0)  [***Esquema de semaforos \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4***](#_bookmark0)  [***Implementação \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***](#_bookmark0)***5***  [***Testes Realizados \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***](#_bookmark0) ***\_8***  ***Bibliografia \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_14*** | |  |
|  | |  |

##### Introdução

#### Neste documento, explicaremos a resolução do 2º Trabalho Prático da disciplina de Sistemas Operativos que tem como objetivo o desenvolvimento e teste de uma aplicação em C que simula uma ponte aérea (constituido por um avião, uma hospdeira, um piloto e N passageiros).

#### Tendo como ajuda principal o uso de semáfotos e memória partilhada pois os passageiros, piloto e hospedeira são processos independentes

#### 

##### image1643595242197.pngEsquema de Semáforos

Este esquema ajuda nos a entender

exatamente onde cada semaforo

toma lugar.

Mesmo este esquema tendo sido

desenhado após o codigo, algumas

versões menores foram feitas ao

longo do projeto para ter sempre

uma boa noção do que era suposto

implementar.

Semáforos:

RFB- readyForBoarding

PIQ- passengersInQueue

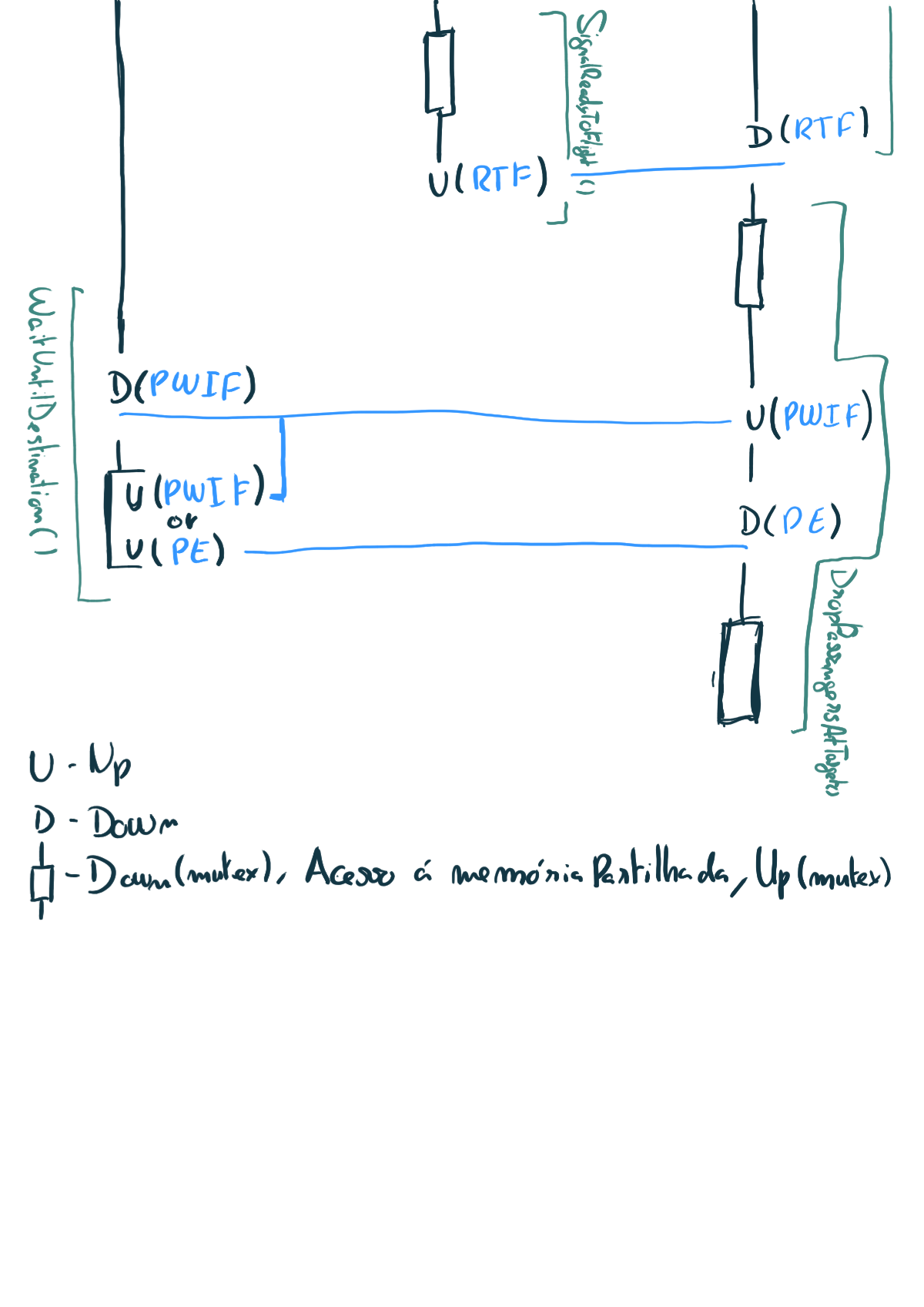
PWIQ- passengersWaitInQueue

IS- idShown

RTF- readyToFlight

PWIF- passengersWaitInFlight

PE- planeEmpty



##### Implementação

##### 

Em relação á implementação após entender a localização dos semáforos fica bastante simples quase sempre só sendo preciso (dentro de um down e up de um mutex) alterar uma variavel ou estado.

Porém, existiram duas interaçoes um pouco mais complexas, uma entre os passageiros e a hospedeira (WaitInQueue() – CheckPassport()) e uma entre os passageiros e o piloto (Waitdestination() - DropPassengersAtTarget()).

WaitInQueue() – CheckPassport():

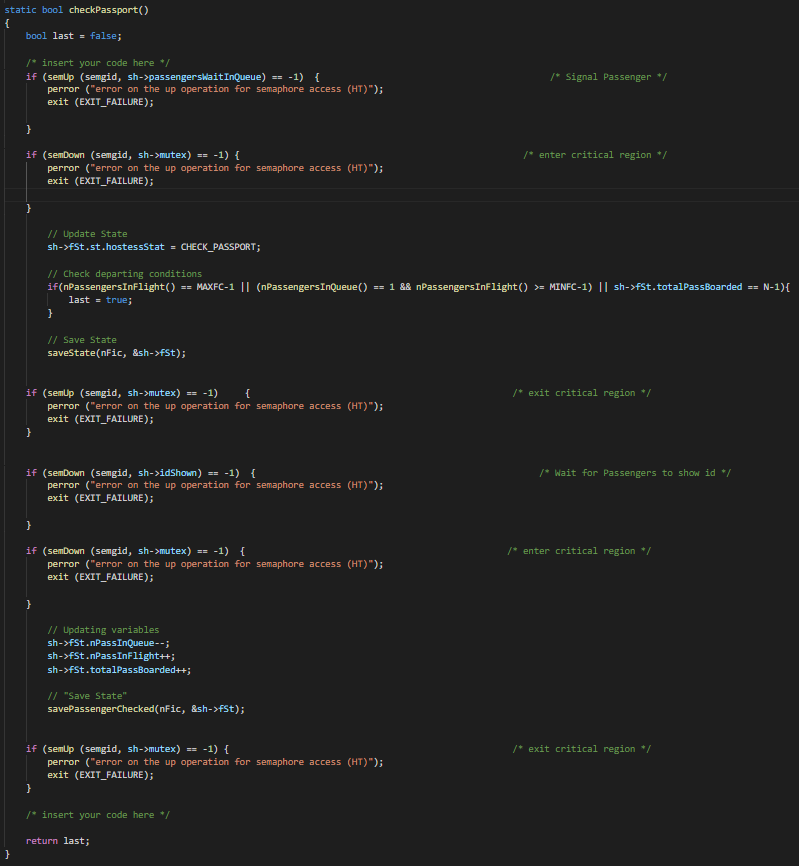
ºAqui o interessante é a forma como se usa o semaforo ‘idShown’ para que processos independentes comunicam entre si, o id do passageiro é armazenado pelo passageiro e usado pela hospedeira isto só sendo possivel através do uso do semáforo.

Waitdestination() - DropPassengersAtTarget():

ºEste foi o mais desafiador, pois como existe N passageiros e estas funções não eram chamadas dentro de um loop como as ateriores, foi preciso usar os semaforos não só para organizar a interação entre piloto e passageiro mas também entre os próprios passageiros.

Como estas são tambem as maiores funções vamos entrar em detalhe apenas nestas quatro, pois para além de terem operações basicas que todas têm(alterar ou ler valor de variaveis), têm também os desafios falados ateriormente.

CheckPassport() [Hostess]



1º- A hospedeira espera

pelo passageiro

2º- Dentro dos semaforos que dão

acesso á area de memoria partilhada

ela testa se já existe alguma das

condições de saida necessárias

e se sim entao muda a variavel de

retorno para indicar que entrou o ultimo

passageiro

3º- A hospedeira espera pelo semaforo

‘idShown’, assim garantindo que mais

tarde ao chamar a função

‘savePassengerChecked’, esta ira ler

o id correto do passageiro que entrou

4º- Dentro dos semaforos que dão

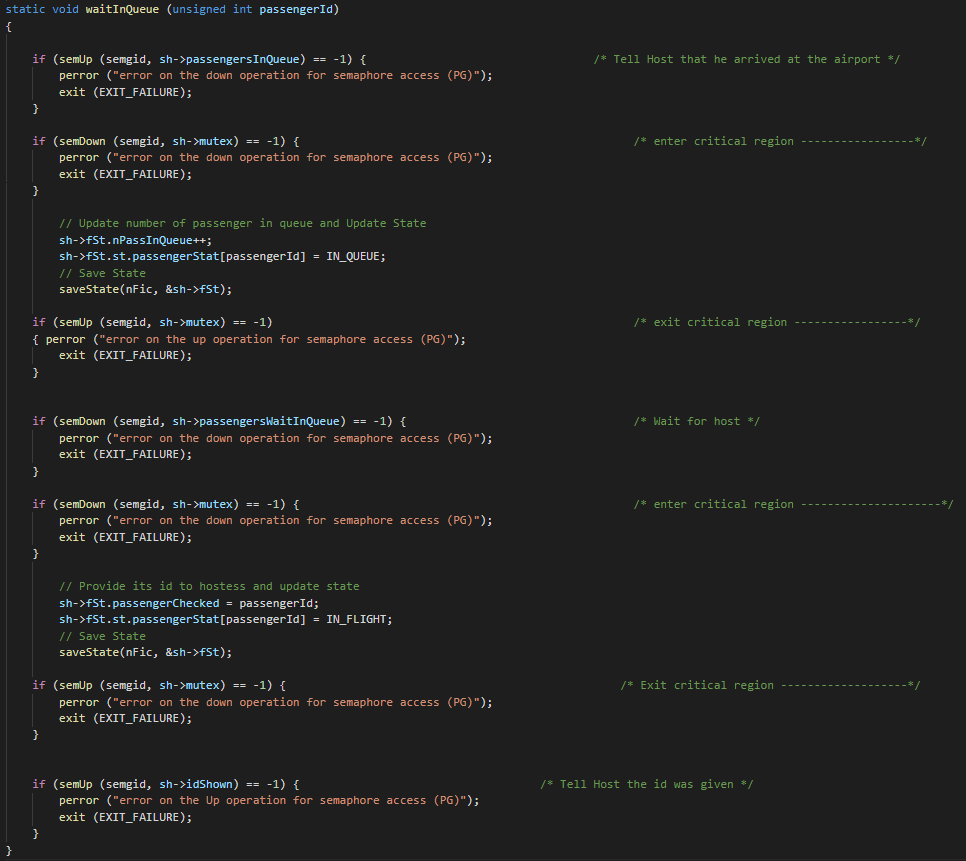
acesso á area de memoria partilhada

são atualizados alguns valores

e é chamada então a funcão para a

qual era importante o id do passageiro

WaitInQueue() [Passenger]



1º- Passageiro anuncia á hospedeira

que chegou dando Up ao semáforo

2º- Dentro dos semaforos que dão

acesso á area de memoria partilhada

são atualizadas algumas variaveis

3º- Espera pela hospedeira pedir o

seu id

4º- Dentro dos semaforos que dão

acesso á area de memoria o passageiro

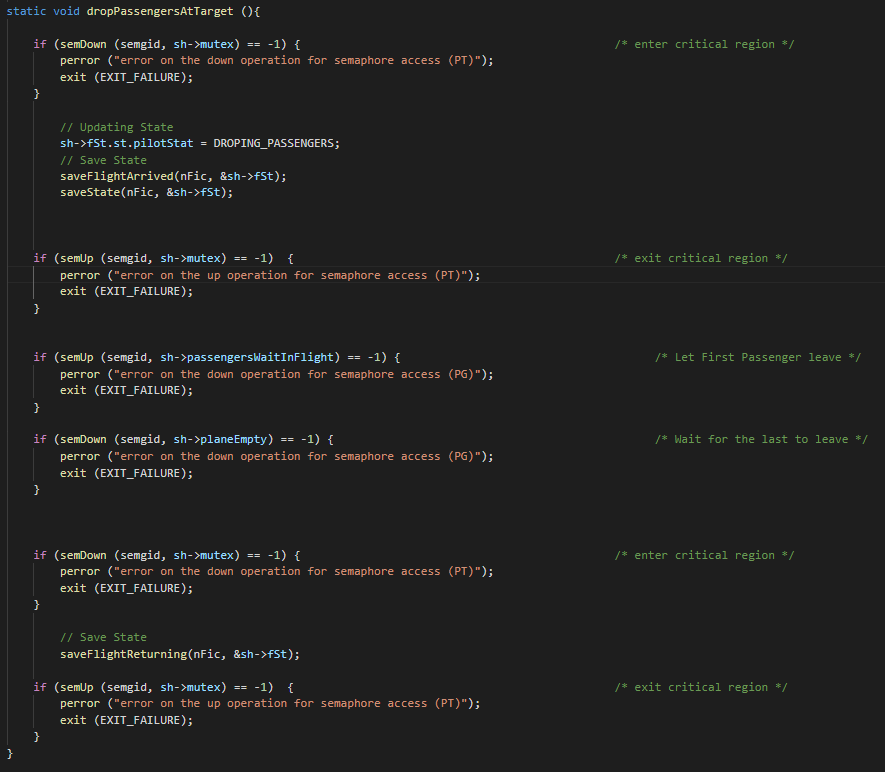
dá o seu id á variavel ‘passengerChecked’

5º- Anuncia á hospedeira que o seu id foi

dado, para que assim a hospedeira o

possa buscar á variavel antes mencionada

Waitdestination() [Passenger]



1º- Dentro dos semaforos que dão

acesso á area de memoria o piloto

atualiza o estado e salva o estado

avisando que o voo aterrou

2º- Avisa o primeiro passageiro que ja

pode sair

3º- Espera que o ultimo passageiro

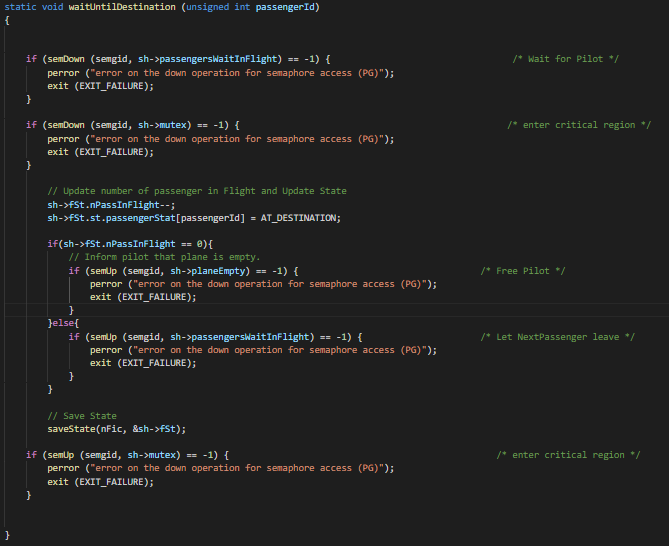
sinalize que saiu

4º- Dentro dos semaforos que dão

acesso á area de memoria o piloto

avisa que o voo vai retornar

DropPassengersAtTarget() [Pilot]



1º- Espera pela sua vez de sair, algo

que vai ser sinalizado pelo piloto caso

ele seja o primeiro, ou por outro

passageiro caso não seja

2º- Dentro dos semaforos que dão

acesso á area de memoria o passageiro,

depois de atualizar o numero de pessoas

do voo e o seu estado, verifica se ele é

o ultimo a sair do voo, caso ele seja

vai sinalizar ao piloto(dando Up a

‘planeEmpty’), caso ele não seja então

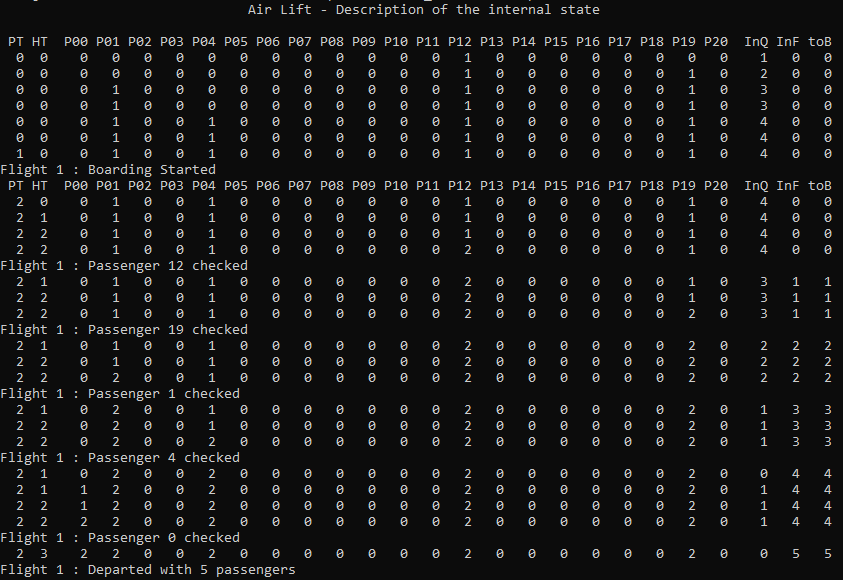
vai avisar a outro passageiro que é a sua

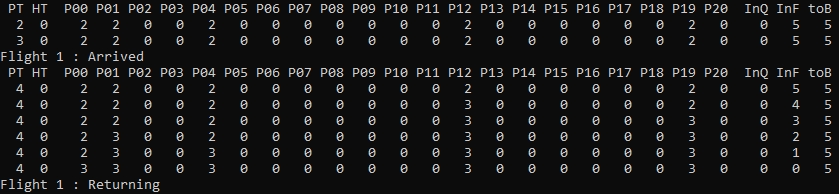
vez de sair(isto vai acontecer em ‘loop’

até que todos saiam)

##### Teste realizados

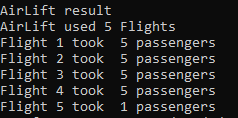
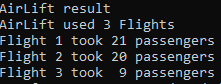
Todas as partes da ponte aeria funcionam corretamente, tal como na versão pré compilada

- Boarding:

- Saida:

-Os resultados finais (N e numero min/max passageiros foram alteradas):

N=21 | min= 5 | max=10 N=21 | min= 22 | max=23 N=50 | min=20 | max=30



### 

N=100 | min= 25 | max= 100 N=1000 | min= 300 | max= 500 N =1000 | min= 10 | max=13

### 

#### 

### Trabalho Realizado por:

## João Correia 104360

## Rafael Remígio 102435

## 

