Simulação de Ponte Aérea

universidade de aveiro

2021/2022
L. Engenharia Informática

Rafael Remígio 102435
João Correia 104360

Professor Nuno Lau
Professor Guilherme Campos

Índice

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).

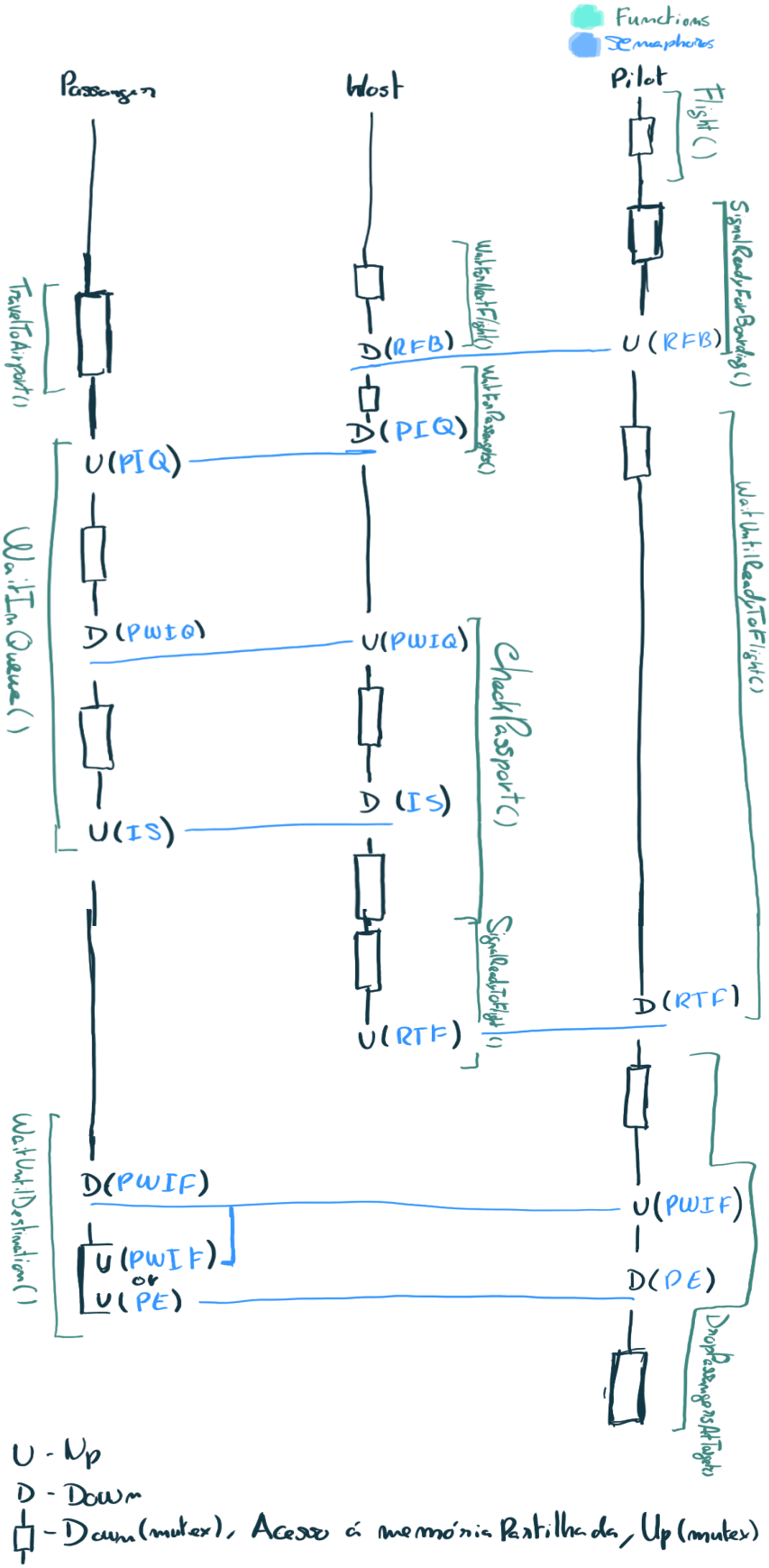
Tendo como ajuda principal o uso de semáfotos e memória partilhada pois os passageiros, piloto e hospedeira são processos independentes

Esquema 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

```
static bool checkPassport()
{
    bool last = false;

    /* insert your code here */
    if (semUp (semgid, sh->passengersWaitInQueue) == -1)  {                                  /* Signal Passenger */
        perror ("error on the up operation for semaphore access (HT)");
        exit (EXIT_FAILURE);

    }

    if (semDown (semgid, sh->mutex) == -1) {                                                /* enter critical region */
        perror ("error on the up operation for semaphore access (HT)");
        exit (EXIT_FAILURE);
    }

        // Update State
        sh->fSt.st.hostessStat = CHECK_PASSPORT;

        // Check departing conditions
        if(nPassengersInFlight() == MAXFC-1 || (nPassengersInQueue() == 1 && nPassengersInFlight() >= MINFC-1) || sh->fSt.totalPassBoarded == N-1){
            last = true;
        }

        // Save State
        saveState(nFic, &sh->fSt);


    if (semUp (semgid, sh->mutex) == -1)     {                                              /* exit critical region */
        perror ("error on the up operation for semaphore access (HT)");
        exit (EXIT_FAILURE);
    }


    if (semDown (semgid, sh->idShown) == -1)  {                                             /* Wait for Passengers to show id */
        perror ("error on the up operation for semaphore access (HT)");
        exit (EXIT_FAILURE);

    }

    if (semDown (semgid, sh->mutex) == -1)  {                                               /* enter critical region */
        perror ("error on the up operation for semaphore access (HT)");
        exit (EXIT_FAILURE);

    }

        // Updating variables
        sh->fSt.nPassInQueue--;
        sh->fSt.nPassInFlight++;
        sh->fSt.totalPassBoarded++;

        // "Save State"
        savePassengerChecked(nFic, &sh->fSt);


    if (semUp (semgid, sh->mutex) == -1) {                                                  /* exit critical region */
        perror ("error on the up operation for semaphore access (HT)");
        exit (EXIT_FAILURE);
    }

    /* insert your code here */

    return last;
}
```

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

```
static void waitInQueue (unsigned int passengerId)
{

    if (semUp (semgid, sh->passengersInQueue) == -1) {                                      /* Tell Host that he arrived at the airport */
        perror ("error on the down operation for semaphore access (PG)");
        exit (EXIT_FAILURE);
    }

    if (semDown (semgid, sh->mutex) == -1) {                                                /* enter critical region ----------------*/
        perror ("error on the down operation for semaphore access (PG)");
        exit (EXIT_FAILURE);
    }

        // Update number of passenger in queue and Update State
        sh->fSt.nPassInQueue++;
        sh->fSt.st.passengerStat[passengerId] = IN_QUEUE;
        // Save State
        saveState(nFic, &sh->fSt);

    if (semUp (semgid, sh->mutex) == -1)                                                    /* exit critical region ----------------*/
    { perror ("error on the up operation for semaphore access (PG)");
        exit (EXIT_FAILURE);
    }


    if (semDown (semgid, sh->passengersWaitInQueue) == -1) {                                /* Wait for host */
        perror ("error on the down operation for semaphore access (PG)");
        exit (EXIT_FAILURE);
    }

    if (semDown (semgid, sh->mutex) == -1) {                                                /* enter critical region --------------------*/
        perror ("error on the down operation for semaphore access (PG)");
        exit (EXIT_FAILURE);
    }

        // Provide its id to hostess and update state
        sh->fSt.passengerChecked = passengerId;
        sh->fSt.st.passengerStat[passengerId] = IN_FLIGHT;
        // Save State
        saveState(nFic, &sh->fSt);

    if (semUp (semgid, sh->mutex) == -1) {                                                  /* Exit critical region ------------------*/
        perror ("error on the down operation for semaphore access (PG)");
        exit (EXIT_FAILURE);
    }


    if (semUp (semgid, sh->idShown) == -1) {                                        /* Tell Host the id was given */
        perror ("error on the Up operation for semaphore access (PG)");
        exit (EXIT_FAILURE);
    }
}
```

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

```
static void dropPassengersAtTarget (){

    if (semDown (semgid, sh->mutex) == -1) {                                    /* enter critical region */
        perror ("error on the down operation for semaphore access (PT)");
        exit (EXIT_FAILURE);
    }

        // Updating State
        sh->fSt.st.pilotStat = DROPING_PASSENGERS;
        // Save State
        saveFlightArrived(nFic, &sh->fSt);
        saveState(nFic, &sh->fSt);



    if (semUp (semgid, sh->mutex) == -1)  {                                     /* exit critical region */
        perror ("error on the up operation for semaphore access (PT)");
        exit (EXIT_FAILURE);
    }


    if (semUp (semgid, sh->passengersWaitInFlight) == -1) {                          /* Let First Passenger leave */
        perror ("error on the down operation for semaphore access (PG)");
        exit (EXIT_FAILURE);
    }

    if (semDown (semgid, sh->planeEmpty) == -1) {                                /* Wait for the last to leave */
        perror ("error on the down operation for semaphore access (PG)");
        exit (EXIT_FAILURE);
    }



    if (semDown (semgid, sh->mutex) == -1) {                                    /* enter critical region */
        perror ("error on the down operation for semaphore access (PT)");
        exit (EXIT_FAILURE);
    }

        // Save State
        saveFlightReturning(nFic, &sh->fSt);

    if (semUp (semgid, sh->mutex) == -1)  {                                     /* exit critical region */
        perror ("error on the up operation for semaphore access (PT)");
        exit (EXIT_FAILURE);
    }
}
```

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)

```
static void waitUntilDestination (unsigned int passengerId)
{


    if (semDown (semgid, sh->passengersWaitInFlight) == -1) {                        /* Wait for Pilot */
        perror ("error on the down operation for semaphore access (PG)");
        exit (EXIT_FAILURE);
    }

    if (semDown (semgid, sh->mutex) == -1) {                                    /* enter critical region */
        perror ("error on the down operation for semaphore access (PG)");
        exit (EXIT_FAILURE);
    }

        // Update number of passenger in Flight and Update State
        sh->fSt.nPassInFlight--;
        sh->fSt.st.passengerStat[passengerId] = AT_DESTINATION;

        if(sh->fSt.nPassInFlight == 0){
            // Inform pilot that plane is empty.
            if (semUp (semgid, sh->planeEmpty) == -1) {                         /* Free Pilot */
                perror ("error on the down operation for semaphore access (PG)");
                exit (EXIT_FAILURE);
            }
        }else{
            if (semUp (semgid, sh->passengersWaitInFlight) == -1) {             /* Let NextPassenger leave */
                perror ("error on the down operation for semaphore access (PG)");
                exit (EXIT_FAILURE);
            }
        }

        // Save State
        saveState(nFic, &sh->fSt);

    if (semUp (semgid, sh->mutex) == -1) {                                      /* enter critical region */
        perror ("error on the down operation for semaphore access (PG)");
        exit (EXIT_FAILURE);
    }


}
```

Teste realizados

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

- Boarding:

```
                                Air Lift - Description of the internal state

 PT HT   P00 P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20  InQ InF toB
  0  0    0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0    1   0   0
  0  0    0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   1   0    2   0   0
  0  0    0   1   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   1   0    3   0   0
  0  0    0   1   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   1   0    3   0   0
  0  0    0   1   0   0   1   0   0   0   0   0   0   0   1   0   0   0   0   0   0   1   0    4   0   0
  0  0    0   1   0   0   1   0   0   0   0   0   0   0   1   0   0   0   0   0   0   1   0    4   0   0
  1  0    0   1   0   0   1   0   0   0   0   0   0   0   1   0   0   0   0   0   0   1   0    4   0   0
Flight 1 : Boarding Started
 PT HT   P00 P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20  InQ InF toB
  2  0    0   1   0   0   1   0   0   0   0   0   0   0   1   0   0   0   0   0   0   1   0    4   0   0
  2  1    0   1   0   0   1   0   0   0   0   0   0   0   1   0   0   0   0   0   0   1   0    4   0   0
  2  2    0   1   0   0   1   0   0   0   0   0   0   0   1   0   0   0   0   0   0   1   0    4   0   0
  2  2    0   1   0   0   1   0   0   0   0   0   0   0   2   0   0   0   0   0   0   1   0    4   0   0
Flight 1 : Passenger 12 checked
  2  1    0   1   0   0   1   0   0   0   0   0   0   0   2   0   0   0   0   0   0   1   0    3   1   1
  2  2    0   1   0   0   1   0   0   0   0   0   0   0   2   0   0   0   0   0   0   1   0    3   1   1
  2  2    0   1   0   0   1   0   0   0   0   0   0   0   2   0   0   0   0   0   0   2   0    3   1   1
Flight 1 : Passenger 19 checked
  2  1    0   1   0   0   1   0   0   0   0   0   0   0   2   0   0   0   0   0   0   2   0    2   2   2
  2  2    0   1   0   0   1   0   0   0   0   0   0   0   2   0   0   0   0   0   0   2   0    2   2   2
  2  2    0   2   0   0   1   0   0   0   0   0   0   0   2   0   0   0   0   0   0   2   0    2   2   2
Flight 1 : Passenger 1 checked
  2  1    0   2   0   0   1   0   0   0   0   0   0   0   2   0   0   0   0   0   0   2   0    1   3   3
  2  2    0   2   0   0   1   0   0   0   0   0   0   0   2   0   0   0   0   0   0   2   0    1   3   3
  2  2    0   2   0   0   2   0   0   0   0   0   0   0   2   0   0   0   0   0   0   2   0    1   3   3
Flight 1 : Passenger 4 checked
  2  1    0   2   0   0   2   0   0   0   0   0   0   0   2   0   0   0   0   0   0   2   0    0   4   4
  2  1    1   2   0   0   2   0   0   0   0   0   0   0   2   0   0   0   0   0   0   2   0    1   4   4
  2  2    1   2   0   0   2   0   0   0   0   0   0   0   2   0   0   0   0   0   0   2   0    1   4   4
  2  2    2   2   0   0   2   0   0   0   0   0   0   0   2   0   0   0   0   0   0   2   0    1   4   4
Flight 1 : Passenger 0 checked
  2  3    2   2   0   0   2   0   0   0   0   0   0   0   2   0   0   0   0   0   0   2   0    0   5   5
Flight 1 : Departed with 5 passengers
```

- Saida:

```
 PT HT   P00 P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20  InQ InF toB
  2  0    2   2   0   0   2   0   0   0   0   0   0   0   2   0   0   0   0   0   0   2   0    0   5   5
  3  0    2   2   0   0   2   0   0   0   0   0   0   0   2   0   0   0   0   0   0   2   0    0   5   5
Flight 1 : Arrived
 PT HT   P00 P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20  InQ InF toB
  4  0    2   2   0   0   2   0   0   0   0   0   0   0   2   0   0   0   0   0   0   2   0    0   5   5
  4  0    2   2   0   0   2   0   0   0   0   0   0   0   3   0   0   0   0   0   0   2   0    0   4   5
  4  0    2   2   0   0   2   0   0   0   0   0   0   0   3   0   0   0   0   0   0   3   0    0   3   5
  4  0    2   3   0   0   2   0   0   0   0   0   0   0   3   0   0   0   0   0   0   3   0    0   2   5
  4  0    2   3   0   0   3   0   0   0   0   0   0   0   3   0   0   0   0   0   0   3   0    0   1   5
  4  0    3   3   0   0   3   0   0   0   0   0   0   0   3   0   0   0   0   0   0   3   0    0   0   5
Flight 1 : Returning
```

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

N=21 | min= 5 | max=10

```
AirLift result
AirLift used 5 Flights
Flight 1 took  5 passengers
Flight 2 took  5 passengers
Flight 3 took  5 passengers
Flight 4 took  5 passengers
Flight 5 took  1 passengers
```

N=21 | min= 22 | max=23

```
AirLift result
AirLift used 1 Flights
Flight 1 took 21 passengers
```

N=50 | min=20 | max=30

```
AirLift result
AirLift used 3 Flights
Flight 1 took 21 passengers
Flight 2 took 20 passengers
Flight 3 took  9 passengers
```

N=100 | min= 25 | max= 100

```
AirLift result
AirLift used 2 Flights
Flight 1 took 84 passengers
Flight 2 took 16 passengers
```

N=1000 | min= 300 | max= 500

```
AirLift result
AirLift used 2 Flights
Flight 1 took 500 passengers
Flight 2 took 500 passengers
```

N =1000 | min= 10 | max=13

```
AirLift result
AirLift used 77 Flights
Flight 1 took 13 passengers
Flight 2 took 13 passengers
Flight 3 took 13 passengers
Flight 4 took 13 passengers
Flight 5 took 13 passengers
Flight 6 took 13 passengers
(...)
```