Sistemas Operativos – Trabalho 2

Simulação de Ponte Aérea

Professor – Nuno Lau (nunolau@ua.pt)

Manuel Diaz	103645	P6	(manu.guerra.diaz@ua.pt)
Tiago Carvalho	104142	P6	(tiagogcarvalho@ua.pt)

ÍNDICE

1 – Introdução ao Trabalho

No âmbito da disciplina de Sistemas Operativos foi-nos proposta a realização de um trabalho prático, com o objetivo da compreensão dos mecanismos associados à execução de processos e *threads*. Para isso, iremos alterar os *scripts* semSharedMemPilot.c, semSharedMemPassenger.c e semSharedMemHostess.c de modo que estes nos permitam simular uma ponte aérea.

A linguagem de programação utilizada no trabalho foi C e, no nosso caso, a implementação do código foi feita no VS Code. Tal como no trabalho anterior, a ferramenta "*live share*" disponível no mesmo permitiu realizar a partilha de código entre os membros do grupo e a elaboração do código no mesmo *script*. Para além disso, a maioria das alterações ao código foram registadas no GitHub.

De acordo com o guião e, tomando como ponto de partida o código presente em semaphore_airlift.tgz, iremos alterar os ficheiros correspondentes ao piloto, aos passageiros e à hospedeira e, uma vez que estes são processos independentes, vamos realizar a sua sincronização através de semáforos e memória partilhada. Para isso, iremos ativar processos quando for necessário e bloquear sempre que estes têm de esperar por algum evento, criando uma certa correspondência entre eles, para no fim obtermos uma ligação entre piloto, passageiros e hospedeira, que nos permita obter uma ponte aérea funcional e sem conflito de processos.

Com a realização deste trabalho esperamos alargar os nossos conhecimentos acerca da programação em C e perceber mais acerca do funcionamento de semáforos e memória partilhada.

2 – Introdução ao Problema

2.1 - Compilação e Execução

Para compilar o programa é necessário à partida ter um compilador C instalado na máquina, por exemplo o *gcc*. Posto isto, de forma a visualizar o resultado da execução de todos os processos numa versão pré-compilada, executamos dentro da pasta …/src/, o comando:

make all_bin

De seguida, entramos dentro da pasta …/run/, para que possamos "simular a ponte aérea", através do comando:

./probSemSharedMemAirLift

Para verificar a existência de algum *deadlock* executámos o ficheiro run.sh que nos permite executar um determinado número de vezes o ./probSemSharedMemAirLift.

2.2 - O Problema

Quando estamos na presença de vários processos a serem executados ao mesmo tempo e, por acaso, esses processos partilham uma ou mais variáveis entre si e, em cada um deles há manipulação dessas variáveis, é muito provável que a execução desse programa não seja a esperada. Para a resolução deste problema é necessário que seja criada uma certa sincronização no controle dessas variáveis. Cada processo irá ter regiões críticas, que são regiões onde são alteradas variáveis a vários processos, e é necessário e extremamente importante assegurar que, quando um certo processo entra nessa região crítica, mais nenhum entra na mesma.

É neste contexto que se incorpora a utilização dos semáforos, que são muito úteis para se obter a sincronização dos processos. No decorrer do trabalho iremos usar semáforos *mutex*, que funcionam só com os números 0 e 1, assegurando sempre exclusão mútua. Para fazermos a sincronização de forma correta iremos ter que seguir várias instruções:

- Os passageiros chegam ao aeroporto e esperam *inQueue*;
- O piloto notifica a hospedeira de que o "avião" está pronto para iniciar o processo de embarque;
- A hospedeira trata da verificação de identidade dos passageiros e dá a ordem para estes entrarem no "avião";
- A hospedeira volta a avisar o piloto que o *boarding* está completo e que o voo pode iniciar e dá a permissão aos passageiros para saírem do "avião" no fim do voo;

- O piloto é informado para iniciar o voo de regresso quando o último passageiro sai do “avião”.

Existe um ficheiro probConst.h que contém todos os estados possíveis que cada indivíduo pode adotar. Estados estes que serão abordados mais à frente.

Existem 3 intervenientes que dispõem das seguintes funções:

- ***Passenger*** – Chegar, embarcar e voar.
- ***Hostess*** – Tratar do processo de boarding
- ***Pilot*** – Vôo e levar passageiros ao destino

No ficheiro sharedDataSync.h encontramos informação relativa à memória partilhada na estrutura FULL_STAT.

A utilização dos semáforos serve essencialmente para controlar o acesso a esta memória através duma sincronização de processos, que impede conflitos de informação entre os 3 indivíduos. Assim, as notificações entre estes foram feitas através da ativação e desativação de semáforos evitando, desta forma, que o programa pudesse ser executado com problemas.

Elaborámos uma tabela com todos os semáforos que foram usados por nós, o que fez com que depois a implementação dos semáforos no código fosse mais fácil e rápida, pois a tabela estrutura de forma bastante eficaz as alterações que serão feitas no código-fonte.

Semáforo	Entidade Down	Função Down	#Downs	Entidade Up	Função Up	#Up
passengersInQueue	Hostess	waitForPassenger	21	Passengers	waitInQueue	21
passengersWaitInQueue	Passengers	waitInQueue	21	Hostess	checkPassport	21
passengersWaitInFlight	Passengers	waitUntilDestination	21	Pilot	dropPassengersAtTarger	21
readyForBoarding	Hostess	waitForNextFlight	1 each flight	Pilot	signalReadyForBoarding	1 each flight
readyToFlight	Pilot	waitUntilReadyToFlight	1 each flight	Hostess	signalReadyToFlight	1 each flight
idShown	Hostess	checkPassport	21	Passengers	waitInQueue	21
planeEmpty	Pilot	dropPassengersAtTarget	1 each flight	Passengers	waitUntilDestination	1 each flight

Figura 1 – Tabela de Semáforos

3 – Estrutura do Código Desenvolvido

3.1 - Pilot

Como foi referido em cima, um dos intervenientes nesta simulação é o piloto, sendo que este, ao longo de toda a simulação, irá alternar entre estados definidos pelo professor no código recebido:

- ***FLYING_BACK***: o piloto está a meio de um voo de regresso ao aeroporto simulado;
- ***READY_FOR_BOARDING***: o piloto chegou e encontra-se pronto para o embarque;
- ***WAITING_FOR_BOARDING***: o piloto aguarda o embarque dos passageiros;
- ***FLYING***: após o embarque, o piloto encontra-se a "pilotar o avião";
- ***DROPPING_PASSENGERS***: o piloto chegou ao destino e começa o desembarque.

```
/** \brief pilot flying to starting airport */
#define  FLYING_BACK                0
/** \brief pilot signals ready for boarding */
#define  READY_FOR_BOARDING         1
/** \brief pilot wait for boarding to complete */
#define  WAITING_FOR_BOARDING       2
/** \brief pilot takes passengers to destination */
#define  FLYING                     3
/** \brief pilot drops passengers at destination */
#define  DROPING_PASSENGERS         4
```

Figura 2 – Estados do Piloto

Para alternar entre os estados, utilizamos quatro funções, inicializadas pelo professor e consequentemente completadas por nós, sendo estas:

```
static void flight (bool go);
static void signalReadyForBoarding ();
static void waitUntilReadyToFlight ();
static void dropPassengersAtTarget ();
```

Figura 3 – Funções do Piloto

3.1.1 - Função flight

A primeira função do piloto, *flight*, simula o processo do voo, para o destino ou de regresso, que é comandado por um valor *booleano*, *go*. Caso este seja verdadeiro, então o "avião" começa o voo e, sendo assim, alteramos o estado do piloto para *FLYING* e guardamos esse estado. Estas alterações são feitas na região critica controlada pelo semáforo *mutex*.

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

    /* insert your code here */
    if(go){
        sh->fSt.st.pilotStat = FLYING;
        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);
    }

    usleep((unsigned int) floor ((MAXFLIGHT * random ()) / RAND_MAX + 100.0));
}
```

Figura 4 – Função flight

3.1.2 - Função signalReadyForBoarding

Nesta função é-nos pedido que o piloto altere o seu estado e notifique à hospedeira que o embarque pode começar assim como o número do voo a ser realizado.

Para tal, dentro da região crítica, alteramos o estado do piloto para *READY_FOR_BOARDING* e guardamos esse estado. Para além disso, ainda dentro da região crítica, incrementamos o número do voo e guardamos os dados do embarque num novo voo. Fora da região crítica, fazemos uma iteração *semUp* do semáforo *readyForBoarding* de forma a notificar a hospedeira que os passageiros podem começar a embarcar.

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

    /* insert your code here */
    sh->fSt.nFlight++;
    saveStartBoarding(nFic, &sh->fSt);
    sh->fSt.st.pilotStat = READY_FOR_BOARDING;
    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);
    }

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

}
```

Figura 5 – Função signalReadyForBoarding

3.1.3 - Função waitUntilReadyToFlight

Nesta função do piloto, simulamos a espera do piloto para que o embarque esteja completo.

Desta forma, alteramos o estado do piloto para *WAITING_FOR_BOARDING* e guardamos esse novo estado, e ainda fazemos um *semDown* do semáforo *readyToFlight*, semáforo este que é responsável por sinalizar o momento em que o embarque terminou.

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

    /* insert your code here */
    sh->fSt.st.pilotStat = WAITING_FOR_BOARDING;
    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);
    }

    /* insert your code here */
    if (semDown(semgid, sh->readyToFlight) == -1) {
        perror ("error on the up operation for semaphore access (PT)");
        exit (EXIT_FAILURE);
    }
}
```

Figura 6 – Função waitUntilReadyToFlight

3.1.4 - Função dropPassengersAtTarget

A quarta função do piloto tem como objetivos simular os passageiros a sair do "avião". Assim sendo, começamos por alterar, dentro da região crítica, o estado do piloto para *DROPING_PASSENGERS* e salvamos o seu novo estado.

De seguida, para sinalizar os passageiros do fim do voo, utilizamos um ciclo *for* para iterar (tantas vezes quantos os passageiros no "avião") o *semUp* para o semáforo *passengersWaitInFlight*, notificando cada um dos passageiros. Ainda fora da região crítica, fazemos *semDown* ao semáforo *planeEmpty*, informando assim o piloto que o "avião" já está a desembarcar.

Finalmente, dentro de outra região crítica, guardamos o novo estado do "avião", que agora se encontra a retornar, com a função *saveFlightReturning*.

```
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);
    }

    /* insert your code here */
    sh->fSt.st.pilotStat = DROPING_PASSENGERS;
    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);
    }

    /* insert your code here */
    for (int i = 0; i < sh->fSt.nPassengersInFlight[sh->fSt.nFlight-1]; i++) {
        if (semUp(semgid, sh->passengersWaitInFlight) == -1) {
            perror ("error on the down operation for semaphore access (PT)");
            exit (EXIT_FAILURE);
        }
    }
    if (semDown(semgid, sh->planeEmpty) == -1) {
        perror ("error on the down operation for semaphore access (PT)");
        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);
    }

    /* insert your code here */
    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);
    }
}
```

Figura 7 – Função dropPassengersAtTarget

3.2 - Passenger

Assim como o piloto, cada um dos passageiros (*passenger*) é um interveniente nesta simulação e, portanto, também vão alternando entre estados consoante o papel que estejam a desempenhar.

- ***GOING_TO_AIRPORT***: o passageiro encontra-se a ir para o aeroporto simulado;
- ***IN_QUEUE***: o passageiro está em fila de espera, a ponto de ser *checked* pela hospedeira e embarcar no voo;
- ***IN_FLIGHT***: o passageiro embarcou e encontra-se a meio de um voo;
- ***AT_DESTINATION***: o voo do passageiro acabou e este desembarca no destino.

```
/** \brief passenger is going to the airport */
#define  GOING_TO_AIRPORT            0
/** \brief passenger is waiting in queue */
#define  IN_QUEUE                    1
/** \brief passenger is flying */
#define  IN_FLIGHT                   2
/** \brief passenger arrives at destination */
#define  AT_DESTINATION              3
```

Figura 8 – Estados dos Passageiros

Para alternar entre estes estados, mais uma vez, usamos duas funções idealizadas pelo professor e completadas por nós, sendo estas as seguintes:

```
static void waitInQueue (unsigned int passengerId);
static void waitUntilDestination (unsigned int passengerId);
```

Figura 9 – Funções dos Passageiros

3.2.1 - Função waitInQueue

Esta função dos passageiros tem a função de descrever todo o processo de cada passageiro desde o momento em que este chega ao aeroporto até ao momento em que embarca no “avião”.

Primeiramente, dentro de uma primeira região crítica, começamos por trabalhar com a memória partilhada, incrementando o número de passageiros em fila de espera para serem *checked*. Para além disso, alteramos o passageiro que corresponde ao *passengerId* para o estado *IN_QUEUE* e guardamos o seu novo estado. De seguida, já fora da região crítica, fazemos *semUp* do semáforo *passengersInQueue* de forma a informar a hospedeira que há passageiros à espera na fila para serem *checked*. Após isso fazemos uma iteração de *semDown* do semáforo *passengersWaitInQueue*, sinalizando que o passageiro está à espera que a hospedeira lhe “*check* a identificação”.

Após a hospedeira ter sido notificada e o passageiro ter saído da fila de espera para ser *checked*, entramos em uma nova região crítica. Nessa região crítica, alteramos o estado do passageiro para *IN_FLIGHT* e guardamos esse estado. Alteramos também a variável *passengerChecked*, responsável por guardar o *ID* do último passageiro a ser *checked*. Finalmente, fazemos uma iteração de *semUp* ao semáforo *idShown*, de forma a fazer com que a hospedeira realize o *check* do último passageiro.

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

    /* insert your code here */
    sh->fSt.nPassInQueue++;
    sh->fSt.st.passengerStat[passengerId] = IN_QUEUE;
    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);
    }

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

    if (semDown (semgid, sh->passengersWaitInQueue) == -1) {
        perror ("error on the up 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);
    }

    /* insert your code here */
    sh->fSt.passengerChecked = passengerId;

    sh->fSt.st.passengerStat[passengerId] = IN_FLIGHT;
    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);
    }

    /* insert your code here */
    if (semUp (semgid, sh->idShown) == -1) {
        perror ("error on the up operation for semaphore access (PG)");
        exit (EXIT_FAILURE);
    }
}
```

Figura 10 – Função waitInQueue

3.2.2 - Função waitUntilDestination

A função *waitUntilDestination* do passageiro tem como objetivo simular o voo do passageiro e que este aguarde o desembarque.

Tendo isso em mente, inicialmente, fazemos um *semDown* do semáforo *passengersWaitInFlight*, sinalizando que os passageiros se encontram em um voo e que devem aguardar que este termine.

Posteriormente, aquando ao término da viagem, utilizamos uma região crítica para decrementar o número de passageiros em voo, *nPassInFlight*, de forma a representar que o passageiro chego ao destino, e, portanto, alteramos e guardamos o novo estado desse passageiro como *AT_DESTINATION*.

Ainda dentro da região crítica, usamos uma condição *if* para verificar quantos passageiros ainda se encontram no "avião" e, caso esse número seja 0 (todos já tenham desembarcado), então iteramos o semáforo *planeEmpty* com *semUp*, notificando o piloto que o desembarque acabou e o "avião" está vazio.

```
static void waitUntilDestination (unsigned int passengerId)
{
    /* insert your code here */
    if (semDown (semgid, sh->passengersWaitInFlight) == -1) {
        perror ("error on the up 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);
    }

    /* insert your code here */
    sh->fSt.nPassInFlight--;
    sh->fSt.st.passengerStat[passengerId] = AT_DESTINATION;
    saveState(nFic, &sh->fSt);

    if (sh->fSt.nPassInFlight == 0) {
        if (semUp(semgid, sh->planeEmpty) == -1) {
            perror ("error on the down operation for semaphore access (PG)");
            exit (EXIT_FAILURE);
        }
    }

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

}
```

Figura 11 – Função waitUntilDestination

3.3 - Hostess

Para terminar os intervenientes nesta simulação, temos a hospedeira (*hostess*), interveniente digamos que "principal", já que tem a capacidade de interagir tanto com o piloto como com os passageiros. Ao longo da simulação, também a hospedeira vai alternando entre estados consoante a sua função em cada momento, sendo estes:

- ***WAIT_FOR_FLIGHT***: a hospedeira espera que o "avião" regresse ao aeroporto;
- ***WAIT_FOR_PASSENGER***: a hospedeira espera que um novo passageiro chegue ao aeroporto;
- ***CHECK_PASSENGER***: a hospedeira realiza o *check-in* de um passageiro que esteja na fila de espera;
- ***READY_TO_FLIGHT***: a hospedeira sinaliza o fim do embarque de todos os passageiros e que o voo pode ser iniciado;

```
/** \brief hostess waits for plane to be ready for boarding */
#define  WAIT_FOR_FLIGHT             0
/** \brief hostess waits for passenger to arrive */
#define  WAIT_FOR_PASSENGER          1
/** \brief hostess checks passenger passport */
#define  CHECK_PASSPORT              2
/** \brief hostess signals boarding is complete */
#define  READY_TO_FLIGHT             3
```

Figura 12 – Estados da Hospedeira

De forma a realizar as suas "atividades" a hospedeira dispõe de quatro funções, sendo estas as seguintes:

```
/** \brief hostess waits for next flight */
static void waitForNextFlight ();

/** \brief hostess waits for passenger */
static void waitForPassenger();

/** \brief hostess checks passport */
static bool checkPassport ();

/** \brief hostess signals boarding is complete */
static void signalReadyToFlight ();
```

Figura 13 – Funções da Hospedeira

3.3.1 - Função waitForNextFlight

Na primeira função da hospedeira, o “avião” encontra-se a retornar para o aeroporto. Assim sendo, usamos a região crítica para alterar o estado da hospedeira para *WAIT_FOR_FLIGHT* e guardamos o seu estado.

Fazemos ainda um *semDown* ao semáforo *readyForBoarding* sinalizando à hospedeira que tem de aguardar que o embarque comece.

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

    /* insert your code here */
    sh->fSt.st.hostessStat = WAIT_FOR_FLIGHT;
    saveState(nFic,&sh->fSt);

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

    /* insert your code here */
    if (semDown(semgid, sh->readyForBoarding) == -1) {
        perror ("error on the down operation for semaphore access (HT)");
        exit (EXIT_FAILURE);
    }
}
```

Figura 14 – Função waitForNextFlight

3.3.2 - Função waitForPassenger

Nesta função, a hospedeira deve aguardar que um novo passageiro chegue ao aeroporto. Para tal, dentro da região crítica, atualizamos o estado da hospedeira para *WAIT_FOR_PASSENGER* e guardamos esse novo estado.

Além disso, também fazemos um *semDown* do semáforo *passengersInQueue*, informando assim que não existe nenhum passageiro que esteja a espera da hospedeira na fila de espera.

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

    /* insert your code here */
    sh->fSt.st.hostessStat = WAIT_FOR_PASSENGER;
    saveState(nFic, &sh->fSt);

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

    /* insert your code here */
    if (semDown(semgid, sh->passengersInQueue) == -1) {
        perror ("error on the down operation for semaphore access (HT)");
        exit (EXIT_FAILURE);
    }
}
```

Figura 15 – Função waitForPassenger

3.3.3 - Função checkPassport

A terceira e, possivelmente mais importante, função da hospedeira é a função responsável por dar *check* ao passaporte de cada passageiro enquanto aguarda que cada um mostre a sua identificação e ainda alertar caso o “avião” tenha os passageiros necessários para partir.

Inicialmente realizamos um *semUp* do semáforo *passengersWaitInQueue*, semáforo usado pelos passageiros para notificar a espera de que estão na fila de espera.

Numa primeira região crítica, fazemos uma alteração do estado da hospedeira para *CHECK_PASSPORT* e salvamos o seu novo estado. De seguida, fora da zona crítica, fazemos um *semDown* no semáforo *idShown*, indicando assim que a hospedeira deu *check* à identificação do passageiro.

Já na segunda região crítica, tratamos do segundo objetivo desta função. Começamos por sinalizar que mais um passaporte foi *checked* com o uso da função *savePassengerChecked*. Seguidamente, decrementamos o número de passageiros na fila de espera, incrementamos o número de pessoas no voo e o número de passageiros a bordo do “avião” e salvamos o estado do “avião”, simbolizando assim a passagem numérica desse passageiro para o “avião”.

De seguida, através do uso de um *if*, verificamos que o "avião" tem o número de passageiros desejado para voar. De forma a que isto se afirme, uma de três condições deve ser verdadeira: o número de passageiros no “avião” deve ser igual à lotação máxima do “avião”; o número máximo de passageiros no voo em questão foi alcançado; ou o número de passageiros é o mínimo necessário para o voo em questão e não existe nenhum passageiro na fila de espera.

Caso uma das condições se afirme, então a função *checkPassport* irá terminar, retornando o valor *true* (verdadeiro). Caso contrário (nenhuma afirmação se confirme), então alteramos a hospedeira para o seu estado anterior, *WAIT_FOR_PASSENGER*, guardamos o seu estado novamente e a função retorna com o valor de *false* (falso).

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

    /* insert your code here */
    if (semUp(semgid, sh->passengersWaitInQueue) == -1) {
        perror ("error on the down 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);

    }

    /* insert your code here */
    sh->fSt.st.hostessStat = CHECK_PASSPORT;
    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);
    }

    /* insert your code here */
    if (semDown (semgid, sh->idShown) == -1) {
        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);

    }

    /* insert your code here */
    savePassengerChecked(nFic, &sh->fSt);
    sh->fSt.nPassInQueue--;
    sh->fSt.nPassInFlight++;
    sh->fSt.totalPassBoarded++;
    saveState(nFic, &sh->fSt);
    if (nPassengersInFlight() == MAXFC || (nPassengersInFlight() >= MINFC && nPassengersInQueue() == 0) || sh->fSt.totalPassBoarded == N) {
        last = true;
    }
    else {
        last = false;
        sh->fSt.st.hostessStat = WAIT_FOR_PASSENGER;
        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);
    }

    /* insert your code here */


    return last;
}
```

Figura 16 – Função checkPassport

3.3.4 - Função signalReadyToFlight

A última função dos intervenientes, *signalReadyToFlight*, é uma função da hospedeira em que esta informa o piloto que o "avião" está pronto para partir. Assim sendo começamos por utilizar a zona crítica para, primeiramente, definir o número de passageiros com que o "avião" irá decolar, obtendo esse valor através da função *nPassengersInFlight*. De seguida definimos o estado da hospedeira para *READY_FOR_FLIGHT* e guardamos tanto esse estado como também informamos que o "avião" partiu com o uso da função *sabeFlightDeparted*.

Finalmente, usamos um *semUp* do semáforo *readyToFlight*, informando assim o piloto que o embarque de todos os passageiros do voo foi completo.

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

    /* insert your code here */
    sh->fSt.nPassengersInFlight[sh->fSt.nFlight-1] = nPassengersInFlight();
    sh->fSt.st.hostessStat = READY_TO_FLIGHT;
    saveState(nFic, &sh->fSt);
    saveFlightDeparted(nFic, &sh->fSt);

    if (sh->fSt.totalPassBoarded == N){
        sh->fSt.finished = true;
    }

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

Figura 17 – Função signalReadyToFlight

4 – Resultados

Com a execução do *probSemSharedMemAirLift* podemos obter o seguinte resultado, que consideramos estar parecido ao resultado que podemos ver ao executar os ficheiros 'bin' do professor. Através da execução do código e das impressões feitas no terminal podemos ver que cada um dos intervenientes está a agir corretamente e pela ordem desejada, tornando assim a simulação tal e qual a desejada. No final da simulação é impresso o resultado do *AirLift* e os números impressos são os corretos em relação aos voos feitos nesta execução.

Para efeitos de testagem, executámos o código implementado com o comando "./run 1000", executando o código mil vezes, número que consideramos suficiente para testagem (sem ser uma execução muito demorada), deforma a verificar se ocorreu algum caso de *deadlock*, concluindo, contudo, que o programa se encontra livre de *deadlocks* ou de qualquer outro problema que impossibilite a sua execução contínua.

```
tsora@DESKTOP-RRHS0DA:/mnt/c/Users/tiago/Documents/GitHub/SO-Project-2/run$ ./probSemSharedMemAirLift
                        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   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0    1   0   0
  0  0    0   1   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0    2   0   0
  0  0    0   1   0   0   0   1   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0    3   0   0
  0  0    0   1   0   0   0   1   0   0   0   0   0   0   1   0   0   0   1   0   0   0   0    4   0   0
  0  0    0   1   0   1   0   1   0   0   0   0   0   0   1   0   0   0   1   0   0   0   0    5   0   0
  0  0    0   1   0   1   1   1   0   0   0   0   0   0   1   0   0   0   1   0   0   0   0    6   0   0
  0  0    0   1   0   1   1   1   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0    7   0   0
  0  0    0   1   0   1   1   1   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0    7   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
  1  0    0   1   0   1   1   1   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0    7   0   0
  2  0    0   1   0   1   1   1   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0    7   0   0
  2  1    0   1   0   1   1   1   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0    7   0   0
  2  2    0   1   0   1   1   1   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0    7   0   0
  2  2    0   1   0   1   1   2   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0    7   0   0
Flight 1 : Passenger 5 checked
  2  2    0   1   0   1   1   2   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0    6   1   1
  2  1    0   1   0   1   1   2   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0    6   1   1
  2  1    0   1   0   1   1   2   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0    6   1   1
  2  2    0   1   0   1   1   2   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0    6   1   1
  2  2    0   2   0   1   1   2   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0    6   1   1
Flight 1 : Passenger 1 checked
  2  2    0   2   0   1   1   2   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0    5   2   2
  2  1    0   2   0   1   1   2   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0    5   2   2
  2  1    0   2   0   1   1   2   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0    5   2   2
  2  2    0   2   0   1   1   2   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0    5   2   2
  2  2    0   2   0   1   1   2   0   0   0   1   0   0   1   0   0   0   2   0   0   0   0    5   2   2
Flight 1 : Passenger 16 checked
  2  2    0   2   0   1   1   2   0   0   0   1   0   0   1   0   0   0   2   0   0   0   0    4   3   3
  2  1    0   2   0   1   1   2   0   0   0   1   0   0   1   0   0   0   2   0   0   0   0    4   3   3
  2  1    0   2   0   1   1   2   0   0   0   1   0   0   1   0   0   0   2   0   0   0   0    4   3   3
  2  2    0   2   0   1   1   2   0   0   0   1   0   0   1   0   0   0   2   0   0   0   0    4   3   3
  2  2    0   2   0   1   1   2   0   0   0   1   0   0   2   0   0   0   2   0   0   0   0    4   3   3
Flight 1 : Passenger 12 checked
  2  2    0   2   0   1   1   2   0   0   0   1   0   0   2   0   0   0   2   0   0   0   0    3   4   4
  2  1    0   2   0   1   1   2   0   0   0   1   0   0   2   0   0   0   2   0   0   0   0    3   4   4
  2  1    0   2   0   1   1   2   0   0   0   1   0   0   2   0   0   0   2   0   0   0   0    3   4   4
  2  2    0   2   0   1   1   2   0   0   0   1   0   0   2   0   0   0   2   0   0   0   0    3   4   4
  2  2    0   2   0   2   1   2   0   0   0   1   0   0   2   0   0   0   2   0   0   0   0    3   4   4
Flight 1 : Passenger 3 checked
  2  2    0   2   0   2   1   2   0   0   0   1   0   0   2   0   0   0   2   0   0   0   0    2   5   5
  2  1    0   2   0   2   1   2   0   0   0   1   0   0   2   0   0   0   2   0   0   0   0    2   5   5
  2  1    0   2   0   2   1   2   0   0   0   1   0   0   2   0   0   0   2   0   0   0   0    2   5   5
  2  2    0   2   0   2   1   2   0   0   0   1   0   0   2   0   0   0   2   0   0   0   0    2   5   5
  2  2    0   2   0   2   2   2   0   0   0   1   0   0   2   0   0   0   2   0   0   0   0    2   5   5
Flight 1 : Passenger 4 checked
```

```
  2  2    0   2   0   2   2   2   0   0   0   1   0   0   2   0   0   0   2   0   0   0   0    1   6   6
  2  1    0   2   0   2   2   2   0   0   0   1   0   0   2   0   0   0   2   0   0   0   0    1   6   6
  2  1    0   2   0   2   2   2   0   0   0   1   0   0   2   0   0   0   2   0   0   0   0    1   6   6
  2  2    0   2   0   2   2   2   0   0   0   1   0   0   2   0   0   0   2   0   0   0   0    1   6   6
  2  2    0   2   0   2   2   2   0   0   0   2   0   0   2   0   0   0   2   0   0   0   0    1   6   6
Flight 1 : Passenger 9 checked
  2  2    0   2   0   2   2   2   0   0   0   2   0   0   2   0   0   0   2   0   0   0   0    0   7   7
  2  3    0   2   0   2   2   2   0   0   0   2   0   0   2   0   0   0   2   0   0   0   0    0   7   7
Flight 1 : Departed with 7 passengers
 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  3    0   2   0   2   2   2   0   0   0   2   0   0   2   0   0   0   2   0   0   1   0    1   7   7
  2  0    0   2   0   2   2   2   0   0   0   2   0   0   2   0   0   0   2   0   0   1   0    1   7   7
  3  0    0   2   0   2   2   2   0   0   0   2   0   0   2   0   0   0   2   0   0   1   0    1   7   7
  4  0    0   2   0   2   2   2   0   0   0   2   0   0   2   0   0   0   2   0   0   1   0    1   7   7
  4  0    0   2   0   2   2   3   0   0   0   2   0   0   2   0   0   0   2   0   0   1   0    1   6   7
  4  0    0   3   0   2   2   3   0   0   0   2   0   0   2   0   0   0   2   0   0   1   0    1   5   7
  4  0    0   3   0   2   2   3   0   0   0   2   0   0   2   0   0   0   3   0   0   1   0    1   4   7
  4  0    0   3   0   2   2   3   0   0   0   2   0   0   3   0   0   0   3   0   0   1   0    1   3   7
  4  0    0   3   0   3   2   3   0   0   0   2   0   0   3   0   0   0   3   0   0   1   0    1   2   7
  4  0    0   3   0   3   3   3   0   0   0   2   0   0   3   0   0   0   3   0   0   1   0    1   1   7
  4  0    0   3   0   3   3   3   0   0   0   3   0   0   3   0   0   0   3   0   0   1   0    1   0   7
  4  0    0   3   0   3   3   3   0   1   0   3   0   0   3   0   0   0   3   0   0   1   0    2   0   7
Flight 1 : Returning
 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    0   3   0   3   3   3   0   1   0   3   0   1   3   0   0   0   3   0   0   1   0    3   0   7
  4  0    0   3   0   3   3   3   0   1   0   3   0   1   3   0   0   1   3   0   0   1   0    4   0   7
Flight 2 : 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
  1  0    0   3   0   3   3   3   0   1   0   3   0   1   3   0   0   1   3   0   0   1   0    4   0   7
  2  0    0   3   0   3   3   3   0   1   0   3   0   1   3   0   0   1   3   0   0   1   0    4   0   7
  2  1    0   3   0   3   3   3   0   1   0   3   0   1   3   0   0   1   3   0   0   1   0    4   0   7
  2  2    0   3   0   3   3   3   0   1   0   3   0   1   3   0   0   1   3   0   0   1   0    4   0   7
  2  2    0   3   0   3   3   3   0   1   0   3   0   1   3   0   0   1   3   0   0   2   0    4   0   7
  2  2    1   3   0   3   3   3   0   1   0   3   0   1   3   0   0   1   3   0   0   2   0    5   0   7
Flight 2 : Passenger 19 checked
  2  2    1   3   0   3   3   3   0   1   0   3   0   1   3   0   0   1   3   0   0   2   0    4   1   8
  2  1    1   3   0   3   3   3   0   1   0   3   0   1   3   0   0   1   3   0   0   2   0    4   1   8
  2  1    1   3   0   3   3   3   0   1   0   3   0   1   3   0   0   1   3   0   0   2   0    4   1   8
  2  2    1   3   0   3   3   3   0   1   0   3   0   1   3   0   0   1   3   0   0   2   0    4   1   8
  2  2    1   3   0   3   3   3   0   2   0   3   0   1   3   0   0   1   3   0   0   2   0    4   1   8
Flight 2 : Passenger 7 checked
  2  2    1   3   0   3   3   3   0   2   0   3   0   1   3   0   0   1   3   0   0   2   0    3   2   9
  2  1    1   3   0   3   3   3   0   2   0   3   0   1   3   0   0   1   3   0   0   2   0    3   2   9
  2  1    1   3   0   3   3   3   0   2   0   3   0   1   3   0   0   1   3   0   0   2   0    3   2   9
  2  2    1   3   0   3   3   3   0   2   0   3   0   1   3   0   0   1   3   0   0   2   0    3   2   9
  2  2    1   3   0   3   3   3   0   2   0   3   0   2   3   0   0   1   3   0   0   2   0    3   2   9
Flight 2 : Passenger 11 checked
  2  2    1   3   0   3   3   3   0   2   0   3   0   2   3   0   0   1   3   0   0   2   0    2   3  10
  2  1    1   3   0   3   3   3   0   2   0   3   0   2   3   0   0   1   3   0   0   2   0    2   3  10
  2  1    1   3   0   3   3   3   0   2   0   3   0   2   3   0   0   1   3   0   0   2   0    2   3  10
  2  2    1   3   0   3   3   3   0   2   0   3   0   2   3   0   0   1   3   0   0   2   0    2   3  10
  2  2    1   3   0   3   3   3   0   2   0   3   0   2   3   0   0   2   3   0   0   2   0    2   3  10
Flight 2 : Passenger 15 checked
```

```
  2  2    1   3   0   3   3   3   0   2   0   3   0   2   3   0   0   2   3   0   0   2   0    1   4  11
  2  1    1   3   0   3   3   3   0   2   0   3   0   2   3   0   0   2   3   0   0   2   0    1   4  11
  2  1    1   3   0   3   3   3   0   2   0   3   0   2   3   0   0   2   3   0   0   2   0    1   4  11
  2  1    1   3   0   3   3   3   1   2   0   3   0   2   3   0   0   2   3   0   0   2   0    2   4  11
  2  2    1   3   0   3   3   3   1   2   0   3   0   2   3   0   0   2   3   0   0   2   0    2   4  11
  2  2    2   3   0   3   3   3   1   2   0   3   0   2   3   0   0   2   3   0   0   2   0    2   4  11
Flight 2 : Passenger 0 checked
  2  2    2   3   0   3   3   3   1   2   0   3   0   2   3   0   0   2   3   0   0   2   0    1   5  12
  2  1    2   3   0   3   3   3   1   2   0   3   0   2   3   0   0   2   3   0   0   2   0    1   5  12
  2  1    2   3   0   3   3   3   1   2   0   3   0   2   3   0   0   2   3   0   0   2   0    1   5  12
  2  2    2   3   0   3   3   3   1   2   0   3   0   2   3   0   0   2   3   0   0   2   0    1   5  12
  2  2    2   3   0   3   3   3   2   2   0   3   0   2   3   0   0   2   3   0   0   2   0    1   5  12
Flight 2 : Passenger 6 checked
  2  2    2   3   0   3   3   3   2   2   0   3   0   2   3   0   0   2   3   0   0   2   0    0   6  13
  2  3    2   3   0   3   3   3   2   2   0   3   0   2   3   0   0   2   3   0   0   2   0    0   6  13
Flight 2 : Departed with 6 passengers
 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   3   0   3   3   3   2   2   0   3   0   2   3   0   0   2   3   0   0   2   0    0   6  13
  3  0    2   3   0   3   3   3   2   2   0   3   0   2   3   0   0   2   3   0   0   2   0    0   6  13
  4  0    2   3   0   3   3   3   2   2   0   3   0   2   3   0   0   2   3   0   0   2   0    0   6  13
  4  0    2   3   0   3   3   3   2   2   0   3   0   2   3   0   0   2   3   0   0   3   0    0   5  13
  4  0    2   3   0   3   3   3   2   3   0   3   0   2   3   0   0   2   3   0   0   3   0    0   4  13
  4  0    2   3   0   3   3   3   2   3   0   3   0   3   3   0   0   2   3   0   0   3   0    0   3  13
  4  0    2   3   0   3   3   3   2   3   0   3   0   3   3   0   0   3   3   0   0   3   0    0   2  13
  4  0    3   3   0   3   3   3   2   3   0   3   0   3   3   0   0   3   3   0   0   3   0    0   1  13
  4  0    3   3   0   3   3   3   3   3   0   3   0   3   3   0   0   3   3   0   0   3   0    0   0  13
Flight 2 : Returning
 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
Flight 3 : 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
  1  0    3   3   0   3   3   3   3   3   0   3   0   3   3   0   0   3   3   0   0   3   0    0   0  13
  2  0    3   3   0   3   3   3   3   3   0   3   0   3   3   0   0   3   3   0   0   3   0    0   0  13
  2  1    3   3   0   3   3   3   3   3   0   3   0   3   3   0   0   3   3   0   0   3   0    0   0  13
  2  1    3   3   0   3   3   3   3   3   1   3   0   3   3   0   0   3   3   0   0   3   0    1   0  13
  2  2    3   3   0   3   3   3   3   3   1   3   0   3   3   0   0   3   3   0   0   3   0    1   0  13
  2  2    3   3   0   3   3   3   3   3   2   3   0   3   3   0   0   3   3   0   0   3   0    1   0  13
Flight 3 : Passenger 8 checked
  2  2    3   3   0   3   3   3   3   3   2   3   0   3   3   0   0   3   3   0   0   3   0    0   1  14
  2  1    3   3   0   3   3   3   3   3   2   3   0   3   3   0   0   3   3   0   0   3   0    0   1  14
  2  1    3   3   0   3   3   3   3   3   2   3   0   3   3   0   0   3   3   0   0   3   0    0   1  14
  2  1    3   3   0   3   3   3   3   3   2   3   0   3   3   0   0   3   3   0   1   3   0    1   1  14
  2  1    3   3   1   3   3   3   3   3   2   3   0   3   3   0   0   3   3   0   1   3   0    2   1  14
  2  2    3   3   1   3   3   3   3   3   2   3   0   3   3   0   0   3   3   0   1   3   0    2   1  14
  2  2    3   3   1   3   3   3   3   3   2   3   0   3   3   0   0   3   3   0   2   3   0    2   1  14
Flight 3 : Passenger 18 checked
  2  2    3   3   1   3   3   3   3   3   2   3   0   3   3   0   0   3   3   0   2   3   0    1   2  15
  2  1    3   3   1   3   3   3   3   3   2   3   0   3   3   0   0   3   3   0   2   3   0    1   2  15
  2  1    3   3   1   3   3   3   3   3   2   3   0   3   3   0   0   3   3   0   2   3   0    1   2  15
  2  2    3   3   1   3   3   3   3   3   2   3   0   3   3   0   0   3   3   0   2   3   0    1   2  15
  2  2    3   3   2   3   3   3   3   3   2   3   0   3   3   0   0   3   3   0   2   3   0    1   2  15
Flight 3 : Passenger 2 checked
```

```
  2  2    3   3   2   3   3   3   3   3   2   3   0   3   3   0   0   3   3   0   2   3   0    0   3  16
  2  1    3   3   2   3   3   3   3   3   2   3   0   3   3   0   0   3   3   0   2   3   0    0   3  16
  2  1    3   3   2   3   3   3   3   3   2   3   0   3   3   0   0   3   3   0   2   3   0    0   3  16
  2  1    3   3   2   3   3   3   3   3   2   3   1   3   3   0   0   3   3   0   2   3   0    1   3  16
  2  2    3   3   2   3   3   3   3   3   2   3   1   3   3   0   0   3   3   0   2   3   0    1   3  16
  2  2    3   3   2   3   3   3   3   3   2   3   2   3   3   0   0   3   3   0   2   3   0    1   3  16
Flight 3 : Passenger 10 checked
  2  2    3   3   2   3   3   3   3   3   2   3   2   3   3   0   0   3   3   0   2   3   0    0   4  17
  2  1    3   3   2   3   3   3   3   3   2   3   2   3   3   0   0   3   3   0   2   3   0    0   4  17
  2  1    3   3   2   3   3   3   3   3   2   3   2   3   3   0   0   3   3   0   2   3   0    0   4  17
  2  1    3   3   2   3   3   3   3   3   2   3   2   3   3   0   1   3   3   0   2   3   0    1   4  17
  2  2    3   3   2   3   3   3   3   3   2   3   2   3   3   0   1   3   3   0   2   3   0    1   4  17
  2  2    3   3   2   3   3   3   3   3   2   3   2   3   3   0   2   3   3   0   2   3   0    1   4  17
Flight 3 : Passenger 14 checked
  2  2    3   3   2   3   3   3   3   3   2   3   2   3   3   0   2   3   3   0   2   3   0    0   5  18
  2  3    3   3   2   3   3   3   3   3   2   3   2   3   3   0   2   3   3   0   2   3   0    0   5  18
Flight 3 : Departed with 5 passengers
 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    3   3   2   3   3   3   3   3   2   3   2   3   3   0   2   3   3   0   2   3   0    0   5  18
  3  0    3   3   2   3   3   3   3   3   2   3   2   3   3   0   2   3   3   0   2   3   0    0   5  18
  4  0    3   3   2   3   3   3   3   3   2   3   2   3   3   0   2   3   3   0   2   3   0    0   5  18
  4  0    3   3   2   3   3   3   3   3   3   3   2   3   3   0   2   3   3   0   2   3   0    0   4  18
  4  0    3   3   2   3   3   3   3   3   3   3   2   3   3   0   2   3   3   0   3   3   0    0   3  18
  4  0    3   3   3   3   3   3   3   3   3   3   2   3   3   0   2   3   3   0   3   3   0    0   2  18
  4  0    3   3   3   3   3   3   3   3   3   3   3   3   3   0   2   3   3   0   3   3   0    0   1  18
  4  0    3   3   3   3   3   3   3   3   3   3   3   3   3   0   3   3   3   0   3   3   0    0   0  18
Flight 3 : Returning
 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
Flight 4 : 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
  1  0    3   3   3   3   3   3   3   3   3   3   3   3   3   0   3   3   3   0   3   3   0    0   0  18
  2  0    3   3   3   3   3   3   3   3   3   3   3   3   3   0   3   3   3   0   3   3   0    0   0  18
  2  1    3   3   3   3   3   3   3   3   3   3   3   3   3   0   3   3   3   0   3   3   0    0   0  18
  2  1    3   3   3   3   3   3   3   3   3   3   3   3   3   0   3   3   3   1   3   3   0    1   0  18
  2  2    3   3   3   3   3   3   3   3   3   3   3   3   3   0   3   3   3   1   3   3   0    1   0  18
  2  2    3   3   3   3   3   3   3   3   3   3   3   3   3   0   3   3   3   2   3   3   0    1   0  18
Flight 4 : Passenger 17 checked
  2  2    3   3   3   3   3   3   3   3   3   3   3   3   3   0   3   3   3   2   3   3   0    0   1  19
  2  1    3   3   3   3   3   3   3   3   3   3   3   3   3   0   3   3   3   2   3   3   0    0   1  19
  2  1    3   3   3   3   3   3   3   3   3   3   3   3   3   0   3   3   3   2   3   3   0    0   1  19
  2  1    3   3   3   3   3   3   3   3   3   3   3   3   3   1   3   3   3   2   3   3   0    1   1  19
  2  2    3   3   3   3   3   3   3   3   3   3   3   3   3   1   3   3   3   2   3   3   0    1   1  19
  2  2    3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   2   3   3   0    1   1  19
Flight 4 : Passenger 13 checked
  2  2    3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   2   3   3   0    0   2  20
  2  1    3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   2   3   3   0    0   2  20
  2  1    3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   2   3   3   0    0   2  20
  2  1    3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   2   3   3   1    1   2  20
  2  2    3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   2   3   3   1    1   2  20
  2  2    3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   2   3   3   2    1   2  20
Flight 4 : Passenger 20 checked
  2  2    3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   2   3   3   2    0   3  21
  2  3    3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   2   3   3   2    0   3  21
Flight 4 : Departed with 3 passengers


 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
  3  3    3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   2   3   3   2    0   3  21
  4  3    3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   2   3   3   2    0   3  21
  4  3    3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   3   3   3   2    0   2  21
  4  3    3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   2    0   1  21
  4  3    3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3    0   0  21
Flight 4 : Returning
 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
AirLift result
AirLift used 4 Flights
Flight 1 took  7 passengers
Flight 2 took  6 passengers
Flight 3 took  5 passengers
Flight 4 took  3 passengers
```

Figura 18 – Resultados Obtidos

5 – Conclusão

Com a realização deste trabalho, conseguimos adquirir um conhecimento muito maior acerca da utilização de semáforos e memória partilhada, o que revela a grande importância da sua realização. Antes do começo da mesma não tínhamos tanto a ideia da sua utilidade, porém à medida que fomos avançando e compreendendo o que era proposto pelo guião, percebemos que a partir da sua utilização podemos fazer com que vários ficheiros partilhem memória entre si e funcionem de acordo com uma certa sincronização, pela qual os semáforos são responsáveis.

Assim, fomos desenvolvendo o nosso código aos poucos, com cuidado, para que todos os processos ficassem na ordem certa e de acordo com aquilo que nos era pedido no guião. Agora que chegamos ao fim, e com o código mais organizado e sintetizado, estamos contentes com os resultados obtidos.

Tocando neste ponto e concluindo, achamos que conseguimos interpretar e responder devidamente ao guião que nos foi proposto pelo professor e podemos nos orgulhar de ter tido a oportunidade de ter feito este trabalho.

6 – Bibliografia

No decorrer da realização do trabalho e principalmente, na ajuda com as condições necessárias, para implementar todas as funcionalidades do programa, consultamos tanto os slides disponibilizados pelo Professor no *e-learning* na página da U.C. de Sistemas Operativos como também os sites que se seguem na lista seguinte:

- https://www.geeksforgeeks.org/use-posix-semaphores-c/
- https://stackoverflow.com/
- https://github.com/tiagosora
- https://www.embedded.com/synchronization-internals-the-semaphore/