universidade de aveiro

# Sistemas Operativos

## Simulação de Ponte Aérea

Trabalho n°2

2021/2022

Vânia Inês Magalhães Morais - 10238

Yanis Marina Faquir - 100181

# Índice

# Introdução

O 2º trabalho prático da unidade curricular Sistemas Operativos consiste em simular uma Ponte Aérea através de um programa escrito na linguagem C, sob forma de tabela, onde cada número corresponderá a um estado atribuído. Vamos precisar de um piloto (Pilot), uma hospedeira (hostess) e de passageiros(passengers), que serão as entidades do nosso programa.

Assim, o objetivo principal deste trabalho é desenvolver um script na linguagem C que nos permita simular várias viagens aéreas. Cada entidade é tratada como um processo independente, logo vamos ter de os sincronizar através de semáforos e memória partilhada

# Abordagem ao problema

Em primeiro lugar, para conseguirmos resolver este projeto, precisamos de compreender todas as regras de como funciona a ponte aérea e o código fornecido pelo docente. Foi-nos disponibilizado um zip **semaphore_airlift.tgz** com todos os scripts necessários, onde só precisamos alterar três destes: **semSharedMemPilot.c**, **semSharedMemPassenger.c**, **semSharedMemHostess.c.** Onde irão, cada um deles, mudar os estados e afetar o estado do piloto, dos passageiros e da hospedeira, respetivamente.

Resumidamente a maneira que este programa deve correr é a seguinte: Há um total de 21 passageiros, 1 hospedeira e um piloto. Os passageiros vão chegando aleatoriamente as regras são que o avião deve descolar de *Origin* sempre que está cheio, ou já tem o número mínimo de passageiros e a fila está vazia, ou ainda quando todos os N passageiros já embarcaram, o avião pode fazer várias viagens para conseguir transportar todos os passageiros. O piloto informa a hospedeira sempre que o avião está pronto para iniciar o processo de embarque. A hospedeira que trata da verificação de identidade dos passageiros, dando autorização a cada passageiro para sair da fila de espera e entrar no avião, esta informa que o boarding está completo e dá permissão aos passageiros para saírem do avião em *Target*. O piloto apenas inicia o voo de regresso quando o último passageiro sai do avião.

Dentro do zip **semaphore_airlift.tgz** encontramos 2 ficheiros distintos, o ficheiro *run* onde vamos correr o script principal do programa **probSemSharedMemAirLift**.c e o ficheiro *src* que é onde se encontram os códigos principais para a resolução do nosso problema; dentro deste último existe o **sharedDataSync.h** onde encontramos a criação dos semáforos e os seus IDS que iremos utilizar em conjunto com outras condições para chegar a uma solução final.

No ficheiro *probConst* encontramos os estados do piloto, da hospedeira e dos passageiros, que são os seguintes:

| | |
|---|---|
| FLYING_BACK | 0 |
| READY_FOR_BOARDING | 1 |
| WAITING_FOR_BOARDING | 2 |
| FLYING | 3 |
| DROPING_PASSENGERS | 4 |

| | |
|---|---|
| WAIT_FOR_FLIGHT | 0 |
| WAIT_FOR_PASSENGER | 1 |
| CHECK_PASSPORT | 2 |
| READY_TO_FLIGHT | 3 |

| | |
|---|---|
| GOING_TO_AIRPORT | 0 |
| IN_QUEUE | 1 |
| IN_FLIGHT | 2 |
| AT_DESTINATION | 3 |

# Estruturação do código

Para resolver o problema proposto, foram implementados 7 semáforos que iremos explicar mais à frente a função de cada um. Um semáforo ajuda a sincronizar a comunicação entre processos ou *threads*.

Para além dos semáforos referidos acima, foi definido um *mutex*, usado em todas as funções para aceder a chamada "zona crítica".

## Semáforos Aplicados

Foi-nos sugerida a construção de uma tabela para ser mais fácil a aplicação dos semáforos a cada função:

| **Semáforo** | **Quem? (Down)** | **Quando? (Down)** | **# Downs** | **Quem? (Up)** | **Quando? (Up)** | **# Ups** |
|---|---|---|---|---|---|---|
| **passengerInQueue** | hostess | WaitForPassenger() | 21 | Passenger | WaitInQueue() | 21 |
| **passengerWaitInQueue** | passenger | WaitInQueue() | 21 | hostess | CheckPassport() | 21 |
| **passengerWaitInFlight** | passenger | WaitUntilDestination() | 1 por voo | pilot | DropPassengersAtTarget() | 1 por voo |
| **readyForBoarding** | hostess | WaitFor NextFlight() | 1 por voo | pilot | SignalReadyForBoarding() | 1 por voo |
| **readyToFlight** | pilot | WaitUntilReadyToFlight() | 1 por voo | hostess | SignalReadyToFlight() | 1 por voo |
| **idShown** | hostess | CheckPassport() | 21 | passenger | WaitInQueue() | 21 |
| **planeEmpty** | pilot | DropPassengersAtTarget() | 1 por voo | passenger | WaitUntilDestination() | 1 por voo |

Grande parte do código foi-nos disponibilizado, sendo apenas necessário completar algumas funções, de modo a fazer o programa correr corretamente.

As funções que serão explicadas à frente serão apenas aquelas em que tivemos que completar o código.

# Explicação do código semSharedMemPassenger.c

## Função waitInQueue(unsigned int passengerId)

```c
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 */                                                     /* exit critical region */
    sh->fSt.st.passengerStat[passengerId]=IN_QUEUE;
    sh->fSt.nPassInQueue++;
    saveState(nFic, &sh->fSt);

    if (semUp (semgid, sh->mutex) == -1) {
        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 down operation for semaphore access (PG)");
        exit (EXIT_FAILURE);
    }
    if (semDown (semgid, sh->passengersWaitInQueue) == -1) {
        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);
}

/* 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 down operation for semaphore access (PG)");
    exit (EXIT_FAILURE);
}
```

Nesta função o passageiro espera pela sua vez para ser *checked* pela hospedeira. Primeiro mudamos o estado do passageiro para **IN_QUEUE** e incrementamos o número de pessoas na fila.

Depois, o passageiro dá *up* no semáforo ***passengersWaitInQueue*** e dá *down* no semáforo ***passengersWaitInQueue***, mostrando à hospedeira que está pronto para o embarque. A variável ***passengerChecked*** torna-se no ID do passageiro e o estado deste passa a ser **IN_FLIGHT**. Por fim é dado um *up* no semáforo ***idShown***, uma vez que o passageiro já mostrou o seu ID.

## Função waitUntilDestination(unsigned int passengerId)

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

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

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

}
```

Nesta função, o passageiro espera que o voo termine e chega ao destino. O passageiro dá *down* no semáforo ***passengersWaitInFlight*** enquanto está no voo e quando este termina, mudamos o estado do passageiro para **IN_DESTINATION**. Decrementamos o número de pessoas no voo e quando o avião estiver vazio, é feito um *up* no semáforo ***planeEmpty.***

# Explicação do código semSharedMemPilot.c

## Função flight()

```
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));
}
```

Nesta função, o piloto leva os passageiros ao seu destino. Se estiverem no voo, mudamos o estado do voo para **FLYING**.

## Função signalReadyForBoarding()

```c
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.st.pilotStat=READY_FOR_BOARDING;
    sh->fSt.nFlight++;
    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);
    }

}
```

Nesta função o piloto informa a hospedeira que o avião está pronto para o embarque. Mudamos o estado do piloto para **READY_FOR_BOARDING** e incrementamos o número do voo. Por fim, o piloto faz um *up* no semáforo ***readyForBoarding.***

## Função waitUntilReadyToFlight()

```c
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);
    }
}
```

Nesta função, o piloto espera que o avião fique cheio de passageiros. Mudamos o estado do piloto para **WAITING_FOR_BOARDING** e ele dá um *down* no semáforo ***readyToFlight***.

## Função dropPassengersAtTarget()

```
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 up 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);
    }
}
```

Nesta função, o piloto deixa os passageiros no seu destino. Mudamos o estado do piloto para **DROPING_PASSENGER** e enquanto houver passageiros dentro do avião, o piloto faz *up* no semáforo ***passengersWaitInFlight***. Depois, faz *down* no semáforo ***palneEmpty*** e o avião fica pronto para retornar.

# Explicação do código semSharedMemPilot.c

## Função waitForNextFlight()

```
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 up operation for semaphore access (HT)");
        exit (EXIT_FAILURE);
    }
}
```

Nesta função, a hospedeira espera pelo próximo voo. Mudamos o estado da hospedeira para **WAIT_FOR_FLIGHT** e esta faz um *down* no semáforo ***readyForBoarding***, mostrando que está à espera do avião.

## Função waitForPassenger()

```
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 up operation for semaphore access (HT)");
        exit (EXIT_FAILURE);
    }
}
```

Nesta função, a hospedeira espera pelos passageiros. Mudamos o seu estado para **WAIT_FOR_PASSENGER** e por fim, a hospedeira faz *down* no semáforo ***passengersInQueue***.

## Função checkPassport()

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

    /* insert your code here */
    if (semUp (semgid, sh->passengersWaitInQueue) == -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 */
    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;
}
```

Nesta função, a hospedeira verifica o passaporte do passageiro e espera que este mostre o seu ID. Primeiro, a hospedeira faz *up* do semáforo ***passengersWaitInQueue*** e depois o seu estado é mudado para **CHECK_PASSPORT**. A hospedeira faz *down* do semáforo ***idShown***. Agora que o passageiro foi "checked", o número de passageiros na fila é decrementado (*nPassInQueue*) e tanto o número de passageiros no voo (*nPassInFlight*) como o número de total de passageiros que embarcaram desde o primeiro voo (*totalPassBoarded*) é incrementado.

Por fim, se o número de passageiros no voo é o máximo, ou se o número de passageiros no voo é menor ou igual ao mínimo e não está ninguém na fila, ou se o número total de passageiros que embarcaram desde o 1º voo é igual a 21 (N), aquele passageiro é o último. Caso contrário, mudamos o estado da hospedeira para **WAIT_FOR_PASSENGER**, porque ainda há mais passageiros.

## Função signalReadyToFlight()

```
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);
    }
}
```

Nesta função, o voo está pronto para partir. A variável do número de passageiros no voo é atualizada, assim como o estado da hospedeira, que passa a ser **READY_TO_FLIGHT**. Agora que o voo partiu, se o número total de passageiros que embarcaram desde o 1º voo forem 21(N), a variável *finished* torna-se *true*, uma vez que não é necessário mais nenhum voo por todos os passageiros terem já sido transportados.

Por fim, a hospedeira faz *up* no semáforo ***readyToFlight***.

# Explicação do código MakeFile

Neste ficheiro fizemos uma alteração mínima, para nos ajudar a perceber se o código estava a correr de forma correta:

```
pg_pt:      passenger      hostess_bin pilot       main clean
```

Como a nossa linha de raciocínio foi fazer o código da hospedeira por último, esta alteração fez jeito para ver se os outro programas funcionavam, enquanto o da hospedeira não estava pronto.

# Testes Realizados

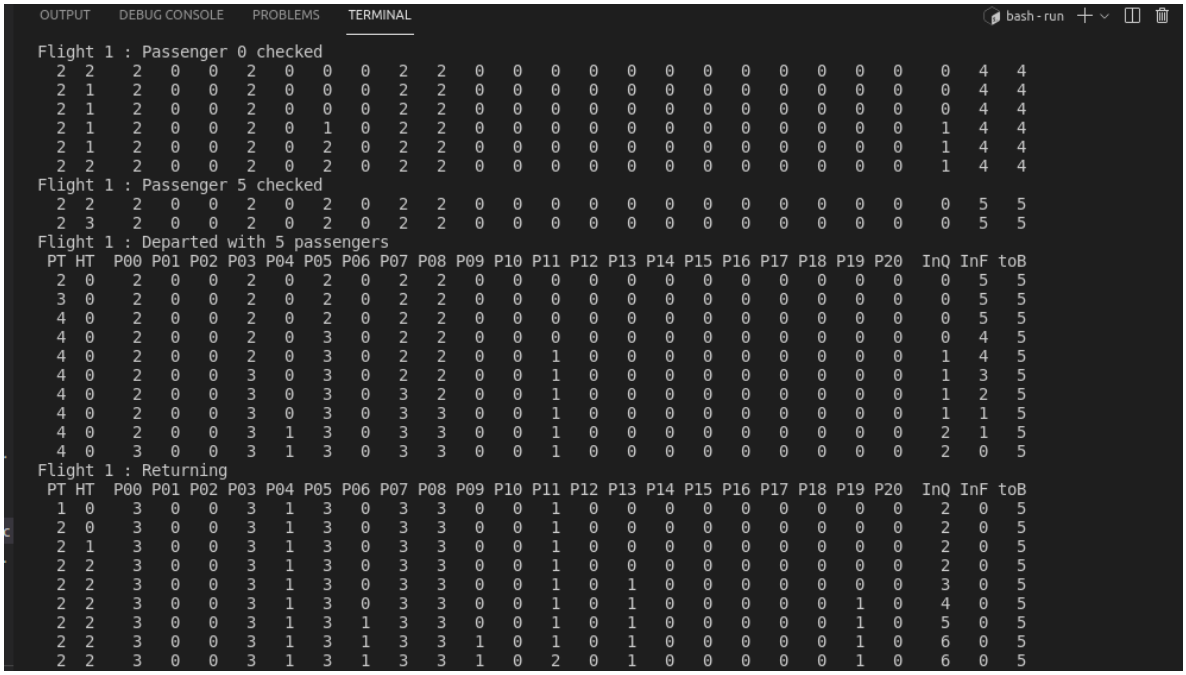
Para compilar os scripts mencionados anteriormente, no terminal, dirigimo-nos para a pasta *src* e compilamos ao escrever make all e obtemos o seguinte resultado:

```
yanisfaquir@yanisfaquir-Aspire-A315-53:~/Desktop/Ua/2ANO/SO/Pratica/Trabalho2/semaphore_airLift/src$ make all
gcc -Wall   -c -o semSharedMemPassenger.o semSharedMemPassenger.c
semSharedMemPassenger.c:47:13: warning: ‘leavePlane’ declared ‘static’ but never defined [-Wunused-function]
   47 | static void leavePlane (unsigned int passengerId);
      |             ^~~~~~~~~~
gcc -Wall   -c -o sharedMemory.o sharedMemory.c
gcc -Wall   -c -o semaphore.o semaphore.c
gcc -Wall   -c -o logging.o logging.c
gcc -o ../run/passenger semSharedMemPassenger.o sharedMemory.o semaphore.o logging.o -lm
gcc -Wall   -c -o semSharedMemHostess.o semSharedMemHostess.c
gcc -o ../run/hostess semSharedMemHostess.o sharedMemory.o semaphore.o logging.o
gcc -Wall   -c -o semSharedMemPilot.o semSharedMemPilot.c
gcc -o ../run/pilot semSharedMemPilot.o sharedMemory.o semaphore.o logging.o -lm
gcc -Wall   -c -o probSemSharedMemAirLift.o probSemSharedMemAirLift.c
gcc -o ../run/probSemSharedMemAirLift probSemSharedMemAirLift.o sharedMemory.o semaphore.o logging.o -lm
rm -f *.o
```

Depois, direcionamo-nos para a pasta *run* e executamos o código **./probSemSharedMemAirLift**

```
yanisfaquir@yanisfaquir-Aspire-A315-53:~/Desktop/Ua/2ANO/SO/Pratica/Trabalho2/semaphore_airLift/run$ ./probSemSharedMemAi
ft
                            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   1   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
  0  0    0   0   0   1   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   2   0   0
  0  0    0   0   0   1   0   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0   3   0   0
  0  0    1   0   0   1   0   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0   4   0   0
  0  0    1   0   0   1   0   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0   4   0   0
  1  0    1   0   0   1   0   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0   4   0   0
  2  0    1   0   0   1   0   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0   4   0   0
  2  1    1   0   0   1   0   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0   4   0   0
  2  2    1   0   0   1   0   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0   4   0   0
  2  2    1   0   0   1   0   0   0   1   2   0   0   0   0   0   0   0   0   0   0   0   0   4   0   0
Flight 1 : Passenger 8 checked
  2  2    1   0   0   1   0   0   0   1   2   0   0   0   0   0   0   0   0   0   0   0   0   3   1   1
  2  1    1   0   0   1   0   0   0   1   2   0   0   0   0   0   0   0   0   0   0   0   0   3   1   1
  2  1    1   0   0   1   0   0   0   1   2   0   0   0   0   0   0   0   0   0   0   0   0   3   1   1
  2  1    1   0   0   2   0   0   0   1   2   0   0   0   0   0   0   0   0   0   0   0   0   3   1   1
  2  2    1   0   0   2   0   0   0   1   2   0   0   0   0   0   0   0   0   0   0   0   0   3   1   1
Flight 1 : Passenger 3 checked
  2  2    1   0   0   2   0   0   0   1   2   0   0   0   0   0   0   0   0   0   0   0   0   2   2   2
  2  1    1   0   0   2   0   0   0   1   2   0   0   0   0   0   0   0   0   0   0   0   0   2   2   2
  2  1    1   0   0   2   0   0   0   1   2   0   0   0   0   0   0   0   0   0   0   0   0   2   2   2
  2  1    1   0   0   2   0   0   0   2   2   0   0   0   0   0   0   0   0   0   0   0   0   2   2   2
  2  2    1   0   0   2   0   0   0   2   2   0   0   0   0   0   0   0   0   0   0   0   0   2   2   2
Flight 1 : Passenger 7 checked
  2  2    1   0   0   2   0   0   0   2   2   0   0   0   0   0   0   0   0   0   0   0   0   1   3   3
  2  1    1   0   0   2   0   0   0   2   2   0   0   0   0   0   0   0   0   0   0   0   0   1   3   3
  2  1    1   0   0   2   0   0   0   2   2   0   0   0   0   0   0   0   0   0   0   0   0   1   3   3
  2  1    2   0   0   2   0   0   0   2   2   0   0   0   0   0   0   0   0   0   0   0   0   1   3   3
  2  2    2   0   0   2   0   0   0   2   2   0   0   0   0   0   0   0   0   0   0   0   0   1   3   3
```




Podemos verificar que os passageiros: P00, P01, P07, P08, encontram-se na fila ou seja estão no estado 1. Conseguimos ver que a variável *InQ* apresenta o número total de passageiros que se encontram na fila, que neste caso são 4 passageiros. Podemos verificar que o P08 passa do estado 1 para o estado 2, o que significa que ele está no voo e será o primeiro a fazer o *checked.*

O piloto passa do estado 0 ao estado 1, o que nos mostra que ele está pronto para o *boarding* em seguida passa para o estado 2 que significa que ele esta a espera do *boarding*. No caso da hospedeira ela passa do estado 0 para o estado 1, o que nos mostra que ela esta a espera dos passageiros e no estado 2 esta pronta para fazer o *check* dos passaportes.

No ***flight 1*** e feito o *checked* do passageiro P08 que vimos anteriormente que seria o primeiro a fazer o *checked*. Entretanto verificamos que o passageiro P03 será o próximo a fazer o *checked e o* passageiro P07 já se encontra na fila. Durante esse processo na variável *InQ mostra* o número de pessoas que ainda estão na fila e a medida que os passageiros vão fazendo *checked* esse número vai decrementando. O *InF* é o número de pessoas que já estão no voo e vai aumentado a mediada que os passageiros vão fazendo *checked*.

Depois do passageiro P03 fazer o *checked* o passageiro P07 já esta pronto para fazer o checked, ou seja, já se encontra no estado 2. Neste momento temos na fila *InQ* 2 passageiros que faltam fazer o *checked*, no InF temos 2 passageiros que já se encontram no voo e temos 2 passageiros dos 21 em bordo na variavel *toB.*

Em seguida o P07 faz o *checked* e temos o P00 no estado 2 que esta pronto para fazer o checked. Neste momento temos na fila *InQ* 1 passageiro que falta fazer o *checked*, no InF temos 3 passageiros que já se encontram no voo e temos 3 passageiros dos 21 a bordo na variavel *toB.*

O último passageiro P00 faz o checked e chega nesse momento o passageiro P05 que é o próximo na fila para fazer o checked. E as variáveis InQ, InF, toB passam aos valores 1, 4 e 4, respetivamente.

E por fim é feito o checked do passageiro P05. O estado da hospedeira muda nesta altura para 3 pois já não há mais passageiros que vão entrar no voo e significa que tudo está apostos para fazer o voo. O comando *Departed with 5 passengers* é impresso para mostar que o voo partiu com 5 passageiros. Entretanto o estado do piloto para de 2 para 3 que significa que estava a pilotar e chegou ao destino e fez o *dropping* dos passageiros que neste momento encontram-se todos no estado 3 que significa que chegaram ao destino. A hospedeira esta no estado 0 significa que esta a espera do próximo voo, entretanto já começam a chegar os passageiros para o voo neste caso são os passageiros P04 e P11, sendo que o passageiro que fará o *checked* primeiro é o P11. Quando o voo está a retornar vemos que mais passageiros estão a chegar: P06, P09, P13 e P19

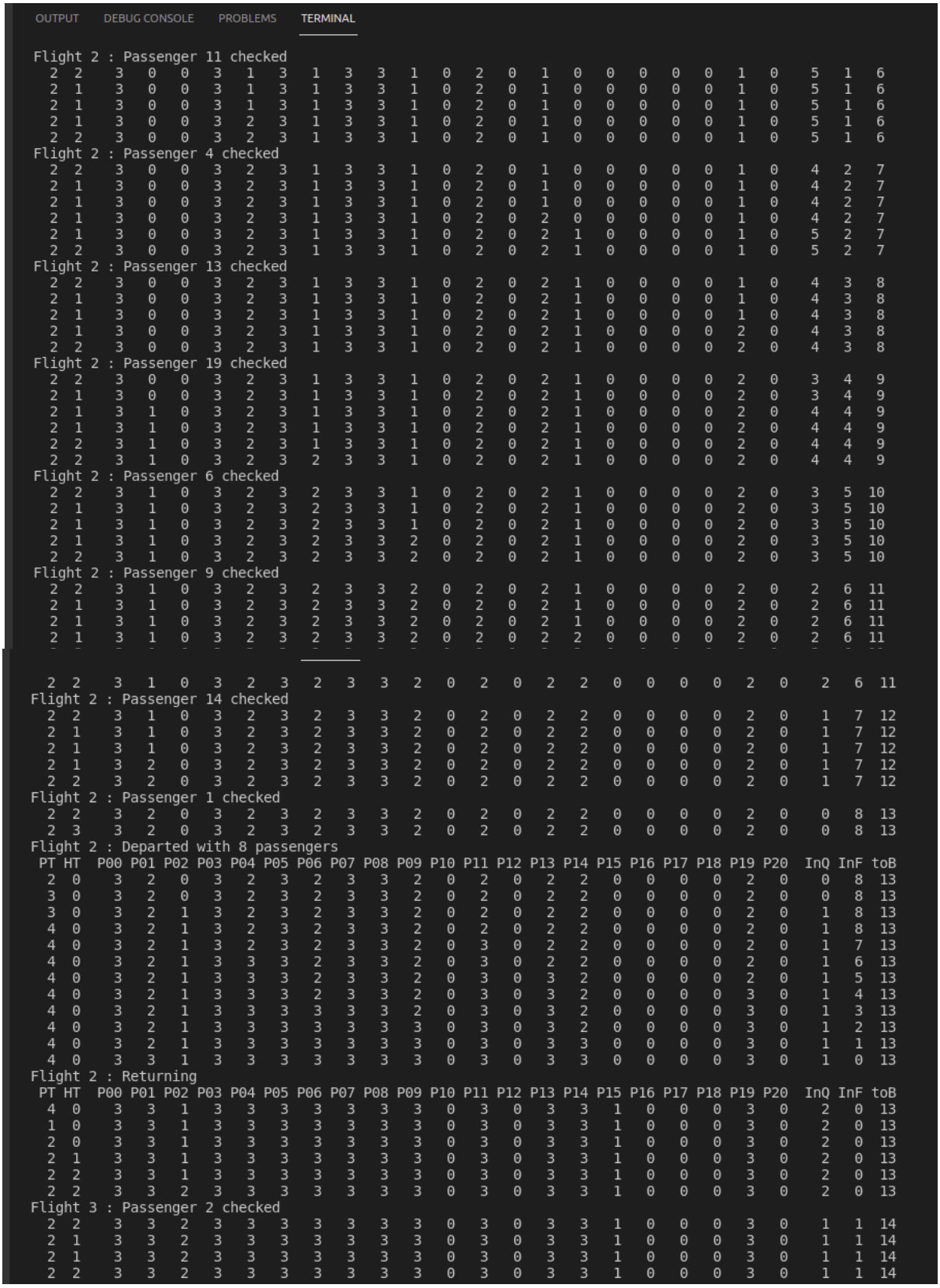
Termina-se assim o primeiro voo com 6 passageiros na fila, InQ, 0 passageiros no voo e 5 passageiros no total dos 21.

Em seguida realizam-se os voos: **Flight 2**, **Flight 3**, **Flight 4** e os resultados obtidos são conforme o esperado.

- O **Flight 1** levou 5 passageiros
- O **Flight 2** levou 8 passageiros
- O **Flight 3** levou 5 passageiros
- O **Flight 4** levou 3passageiros

Em anexo encontram se os resultados de todos os voos.

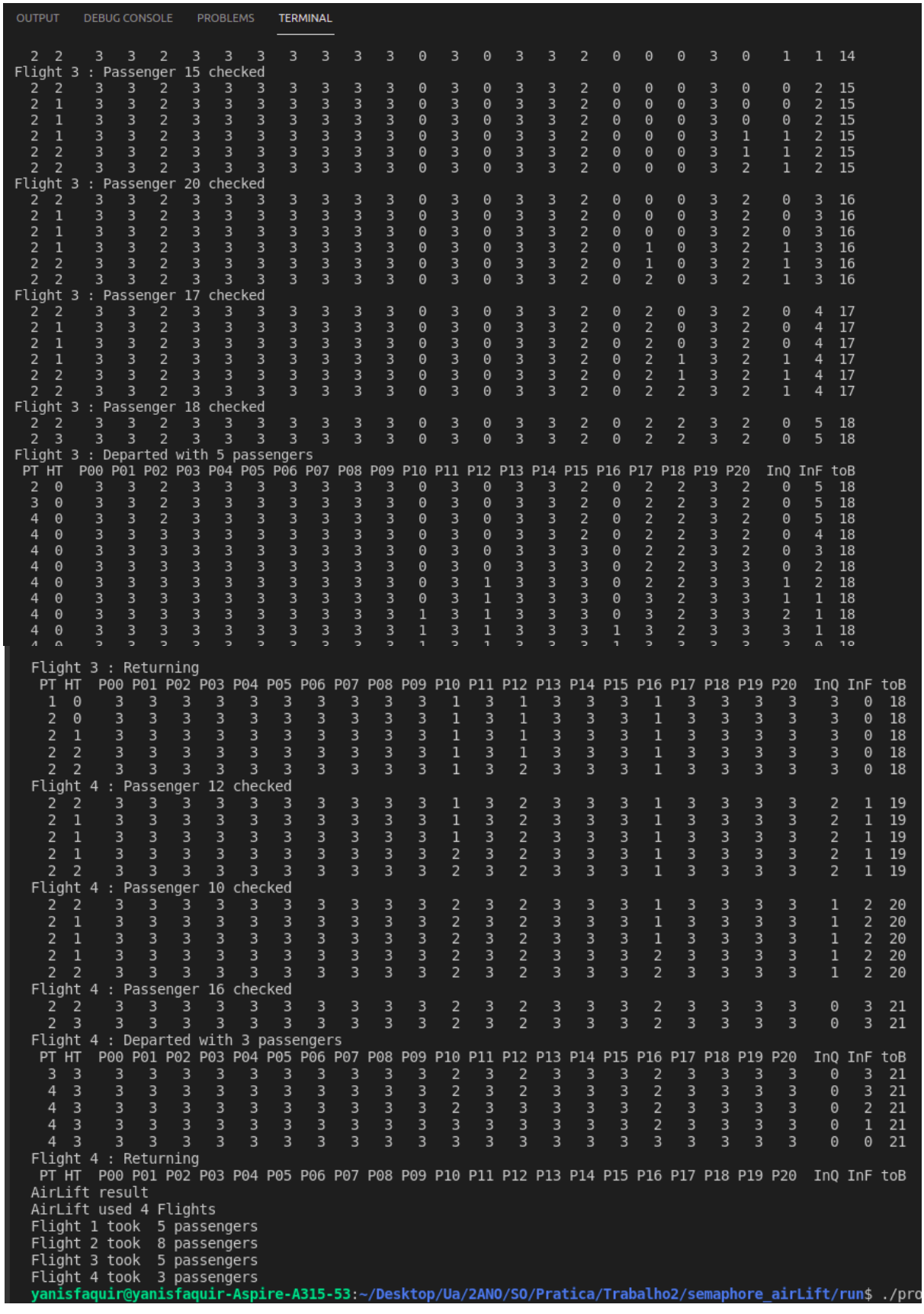
# Anexo



```
OUTPUT   DEBUG CONSOLE   PROBLEMS   TERMINAL

Flight 2 : Passenger 11 checked
  2  2    3   0   0   3   1   3   1   3   3   1   0   2   0   1   0   0   0   0   0   1   0    5   1   6
  2  1    3   0   0   3   1   3   1   3   3   1   0   2   0   1   0   0   0   0   0   1   0    5   1   6
  2  1    3   0   0   3   1   3   1   3   3   1   0   2   0   1   0   0   0   0   0   1   0    5   1   6
  2  1    3   0   0   3   2   3   1   3   3   1   0   2   0   1   0   0   0   0   0   1   0    5   1   6
  2  2    3   0   0   3   2   3   1   3   3   1   0   2   0   1   0   0   0   0   0   1   0    5   1   6
Flight 2 : Passenger 4 checked
  2  2    3   0   0   3   2   3   1   3   3   1   0   2   0   1   0   0   0   0   0   1   0    4   2   7
  2  1    3   0   0   3   2   3   1   3   3   1   0   2   0   1   0   0   0   0   0   1   0    4   2   7
  2  1    3   0   0   3   2   3   1   3   3   1   0   2   0   1   0   0   0   0   0   1   0    4   2   7
  2  1    3   0   0   3   2   3   1   3   3   1   0   2   0   2   0   0   0   0   0   1   0    4   2   7
  2  1    3   0   0   3   2   3   1   3   3   1   0   2   0   2   1   0   0   0   0   1   0    5   2   7
  2  2    3   0   0   3   2   3   1   3   3   1   0   2   0   2   1   0   0   0   0   1   0    5   2   7
Flight 2 : Passenger 13 checked
  2  2    3   0   0   3   2   3   1   3   3   1   0   2   0   2   1   0   0   0   0   1   0    4   3   8
  2  1    3   0   0   3   2   3   1   3   3   1   0   2   0   2   1   0   0   0   0   1   0    4   3   8
  2  1    3   0   0   3   2   3   1   3   3   1   0   2   0   2   1   0   0   0   0   1   0    4   3   8
  2  1    3   0   0   3   2   3   1   3   3   1   0   2   0   2   1   0   0   0   0   2   0    4   3   8
  2  2    3   0   0   3   2   3   1   3   3   1   0   2   0   2   1   0   0   0   0   2   0    4   3   8
Flight 2 : Passenger 19 checked
  2  2    3   0   0   3   2   3   1   3   3   1   0   2   0   2   1   0   0   0   0   2   0    3   4   9
  2  1    3   0   0   3   2   3   1   3   3   1   0   2   0   2   1   0   0   0   0   2   0    3   4   9
  2  1    3   1   0   3   2   3   1   3   3   1   0   2   0   2   1   0   0   0   0   2   0    4   4   9
  2  1    3   1   0   3   2   3   1   3   3   1   0   2   0   2   1   0   0   0   0   2   0    4   4   9
  2  2    3   1   0   3   2   3   1   3   3   1   0   2   0   2   1   0   0   0   0   2   0    4   4   9
  2  2    3   1   0   3   2   3   2   3   3   1   0   2   0   2   1   0   0   0   0   2   0    4   4   9
Flight 2 : Passenger 6 checked
  2  2    3   1   0   3   2   3   2   3   3   1   0   2   0   2   1   0   0   0   0   2   0    3   5  10
  2  1    3   1   0   3   2   3   2   3   3   1   0   2   0   2   1   0   0   0   0   2   0    3   5  10
  2  1    3   1   0   3   2   3   2   3   3   1   0   2   0   2   1   0   0   0   0   2   0    3   5  10
  2  1    3   1   0   3   2   3   2   3   3   2   0   2   0   2   1   0   0   0   0   2   0    3   5  10
  2  2    3   1   0   3   2   3   2   3   3   2   0   2   0   2   1   0   0   0   0   2   0    3   5  10
Flight 2 : Passenger 9 checked
  2  2    3   1   0   3   2   3   2   3   3   2   0   2   0   2   1   0   0   0   0   2   0    2   6  11
  2  1    3   1   0   3   2   3   2   3   3   2   0   2   0   2   1   0   0   0   0   2   0    2   6  11
  2  1    3   1   0   3   2   3   2   3   3   2   0   2   0   2   1   0   0   0   0   2   0    2   6  11
  2  1    3   1   0   3   2   3   2   3   3   2   0   2   0   2   2   0   0   0   0   2   0    2   6  11

  2  2    3   1   0   3   2   3   2   3   3   2   0   2   0   2   2   0   0   0   0   2   0    2   6  11
Flight 2 : Passenger 14 checked
  2  2    3   1   0   3   2   3   2   3   3   2   0   2   0   2   2   0   0   0   0   2   0    1   7  12
  2  1    3   1   0   3   2   3   2   3   3   2   0   2   0   2   2   0   0   0   0   2   0    1   7  12
  2  1    3   1   0   3   2   3   2   3   3   2   0   2   0   2   2   0   0   0   0   2   0    1   7  12
  2  1    3   2   0   3   2   3   2   3   3   2   0   2   0   2   2   0   0   0   0   2   0    1   7  12
  2  2    3   2   0   3   2   3   2   3   3   2   0   2   0   2   2   0   0   0   0   2   0    1   7  12
Flight 2 : Passenger 1 checked
  2  2    3   2   0   3   2   3   2   3   3   2   0   2   0   2   2   0   0   0   0   2   0    0   8  13
  2  3    3   2   0   3   2   3   2   3   3   2   0   2   0   2   2   0   0   0   0   2   0    0   8  13
Flight 2 : Departed with 8 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   2   0   3   2   3   2   3   3   2   0   2   0   2   2   0   0   0   0   2   0    0   8  13
  3  0    3   2   0   3   2   3   2   3   3   2   0   2   0   2   2   0   0   0   0   2   0    0   8  13
  3  0    3   2   1   3   2   3   2   3   3   2   0   2   0   2   2   0   0   0   0   2   0    1   8  13
  4  0    3   2   1   3   2   3   2   3   3   2   0   2   0   2   2   0   0   0   0   2   0    1   8  13
  4  0    3   2   1   3   2   3   2   3   3   2   0   3   0   2   2   0   0   0   0   2   0    1   7  13
  4  0    3   2   1   3   3   3   2   3   3   2   0   3   0   2   2   0   0   0   0   2   0    1   6  13
  4  0    3   2   1   3   3   3   2   3   3   2   0   3   0   3   2   0   0   0   0   2   0    1   5  13
  4  0    3   2   1   3   3   3   2   3   3   2   0   3   0   3   2   0   0   0   0   3   0    1   4  13
  4  0    3   2   1   3   3   3   3   3   3   2   0   3   0   3   2   0   0   0   0   3   0    1   3  13
  4  0    3   2   1   3   3   3   3   3   3   3   0   3   0   3   2   0   0   0   0   3   0    1   2  13
  4  0    3   2   1   3   3   3   3   3   3   3   0   3   0   3   3   0   0   0   0   3   0    1   1  13
  4  0    3   3   1   3   3   3   3   3   3   3   0   3   0   3   3   0   0   0   0   3   0    1   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
  4  0    3   3   1   3   3   3   3   3   3   3   0   3   0   3   3   1   0   0   0   3   0    2   0  13
  1  0    3   3   1   3   3   3   3   3   3   3   0   3   0   3   3   1   0   0   0   3   0    2   0  13
  2  0    3   3   1   3   3   3   3   3   3   3   0   3   0   3   3   1   0   0   0   3   0    2   0  13
  2  1    3   3   1   3   3   3   3   3   3   3   0   3   0   3   3   1   0   0   0   3   0    2   0  13
  2  2    3   3   1   3   3   3   3   3   3   3   0   3   0   3   3   1   0   0   0   3   0    2   0  13
  2  2    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   1   0   0   0   3   0    2   0  13
Flight 3 : Passenger 2 checked
  2  2    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   1   0   0   0   3   0    1   1  14
  2  1    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   1   0   0   0   3   0    1   1  14
  2  1    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   1   0   0   0   3   0    1   1  14
  2  2    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   1   0   0   0   3   0    1   1  14
```



```
OUTPUT   DEBUG CONSOLE   PROBLEMS   TERMINAL

  2  2    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   0   0   3   0    1   1  14
Flight 3 : Passenger 15 checked
  2  2    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   0   0   3   0    0   2  15
  2  1    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   0   0   3   0    0   2  15
  2  1    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   0   0   3   0    0   2  15
  2  1    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   0   0   3   1    1   2  15
  2  2    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   0   0   3   1    1   2  15
  2  2    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   0   0   3   2    1   2  15
Flight 3 : Passenger 20 checked
  2  2    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   0   0   3   2    0   3  16
  2  1    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   0   0   3   2    0   3  16
  2  1    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   0   0   3   2    0   3  16
  2  1    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   1   0   3   2    1   3  16
  2  2    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   1   0   3   2    1   3  16
  2  2    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   2   0   3   2    1   3  16
Flight 3 : Passenger 17 checked
  2  2    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   2   0   3   2    0   4  17
  2  1    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   2   0   3   2    0   4  17
  2  1    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   2   0   3   2    0   4  17
  2  1    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   2   1   3   2    1   4  17
  2  2    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   2   1   3   2    1   4  17
  2  2    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   2   2   3   2    1   4  17
Flight 3 : Passenger 18 checked
  2  2    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   2   2   3   2    0   5  18
  2  3    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   2   2   3   2    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   3   3   0   3   0   3   3   2   0   2   2   3   2    0   5  18
  3  0    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   2   2   3   2    0   5  18
  4  0    3   3   2   3   3   3   3   3   3   3   0   3   0   3   3   2   0   2   2   3   2    0   5  18
  4  0    3   3   3   3   3   3   3   3   3   3   0   3   0   3   3   2   0   2   2   3   2    0   4  18
  4  0    3   3   3   3   3   3   3   3   3   3   0   3   0   3   3   3   0   2   2   3   2    0   3  18
  4  0    3   3   3   3   3   3   3   3   3   3   0   3   0   3   3   3   0   2   2   3   3    0   2  18
  4  0    3   3   3   3   3   3   3   3   3   3   0   3   1   3   3   3   0   2   2   3   3    1   2  18
  4  0    3   3   3   3   3   3   3   3   3   3   0   3   1   3   3   3   0   3   2   3   3    1   1  18
  4  0    3   3   3   3   3   3   3   3   3   3   1   3   1   3   3   3   0   3   2   3   3    2   1  18
  4  0    3   3   3   3   3   3   3   3   3   3   1   3   1   3   3   3   1   3   2   3   3    3   1  18
  4  0    3   3   3   3   3   3   3   3   3   3   1   3   1   3   3   3   1   3   2   3   3    3   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
  1  0    3   3   3   3   3   3   3   3   3   3   1   3   1   3   3   3   1   3   3   3   3    3   0  18
  2  0    3   3   3   3   3   3   3   3   3   3   1   3   1   3   3   3   1   3   3   3   3    3   0  18
  2  1    3   3   3   3   3   3   3   3   3   3   1   3   1   3   3   3   1   3   3   3   3    3   0  18
  2  2    3   3   3   3   3   3   3   3   3   3   1   3   1   3   3   3   1   3   3   3   3    3   0  18
  2  2    3   3   3   3   3   3   3   3   3   3   1   3   2   3   3   3   1   3   3   3   3    3   0  18
Flight 4 : Passenger 12 checked
  2  2    3   3   3   3   3   3   3   3   3   3   1   3   2   3   3   3   1   3   3   3   3    2   1  19
  2  1    3   3   3   3   3   3   3   3   3   3   1   3   2   3   3   3   1   3   3   3   3    2   1  19
  2  1    3   3   3   3   3   3   3   3   3   3   1   3   2   3   3   3   1   3   3   3   3    2   1  19
  2  1    3   3   3   3   3   3   3   3   3   3   2   3   2   3   3   3   1   3   3   3   3    2   1  19
  2  2    3   3   3   3   3   3   3   3   3   3   2   3   2   3   3   3   1   3   3   3   3    2   1  19
Flight 4 : Passenger 10 checked
  2  2    3   3   3   3   3   3   3   3   3   3   2   3   2   3   3   3   1   3   3   3   3    1   2  20
  2  1    3   3   3   3   3   3   3   3   3   3   2   3   2   3   3   3   1   3   3   3   3    1   2  20
  2  1    3   3   3   3   3   3   3   3   3   3   2   3   2   3   3   3   1   3   3   3   3    1   2  20
  2  1    3   3   3   3   3   3   3   3   3   3   2   3   2   3   3   3   2   3   3   3   3    1   2  20
  2  2    3   3   3   3   3   3   3   3   3   3   2   3   2   3   3   3   2   3   3   3   3    1   2  20
Flight 4 : Passenger 16 checked
  2  2    3   3   3   3   3   3   3   3   3   3   2   3   2   3   3   3   2   3   3   3   3    0   3  21
  2  3    3   3   3   3   3   3   3   3   3   3   2   3   2   3   3   3   2   3   3   3   3    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   2   3   2   3   3   3   2   3   3   3   3    0   3  21
  4  3    3   3   3   3   3   3   3   3   3   3   2   3   2   3   3   3   2   3   3   3   3    0   3  21
  4  3    3   3   3   3   3   3   3   3   3   3   2   3   3   3   3   3   2   3   3   3   3    0   2  21
  4  3    3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   3    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  5 passengers
Flight 2 took  8 passengers
Flight 3 took  5 passengers
Flight 4 took  3 passengers
yanisfaquir@yanisfaquir-Aspire-A315-53:~/Desktop/Ua/2ANO/SO/Pratica/Trabalho2/semaphore_airLift/run$ ./pro
```

# Conclusão

Ao longo deste trabalho, a nível teórico, consolidamos os nossos conhecimentos sobre os processos, semáforos, o que são e como funcionam, a sua utilidade e diversidade. A nível prático melhoramos o nosso conhecimento sobre a linguagem C, aprimorando as nossas habilidades de programação ao implementar novas metodologias de trabalho e pesquisa.

# Fonte

Para a concretização deste trabalho, para além as nossas bases adquiridas em semestres transatos, também nos baseamos na matéria lecionada durante as aulas teóricas e práticas da unidade curricular Sistemas Operativos, recorremos ao docente e pesquisamos em fóruns online.