universidade de aveiro

deti

Simulação de Ponte Aérea

Sistemas Operativos 21/22

Trabalho Prático 2

Elementos do grupo:

- André Butuc 103530 (50%)
- Gonçalo Silva 103668 (50%)

Índice

Introdução

O presente relatório tem como objetivo analisar e estudar a solução produzida pelo grupo ao problema proposto pelo professor Nuno Lau, no âmbito da cadeira de Sistemas Operativos, nomeadamente a “Simulação de Ponte Aérea”.

O problema proposto trata de temas como a sincronização de processos/threads e o acesso à memória partilhada por eles. Mais especificamente, teremos 24 processos: 21 “Passenger(s)”, 1 “Hostess” e 1 “Pilot” (nota: no decorrer do relatório iremos usar os seguintes termos “passageiro” para o processo “Passenger”, “hospedeira” para o processo “Hostess” e “piloto” para o processo “Pilot” para facilitar as explicações). Estes processos/entidades vão participar na simulação de uma situação que acontece diariamente num aeroporto, nomeadamente a chegada de passageiros, o seu check-in, o seu embarque no avião, a descolagem do avião e a consequente chegada ao destino.

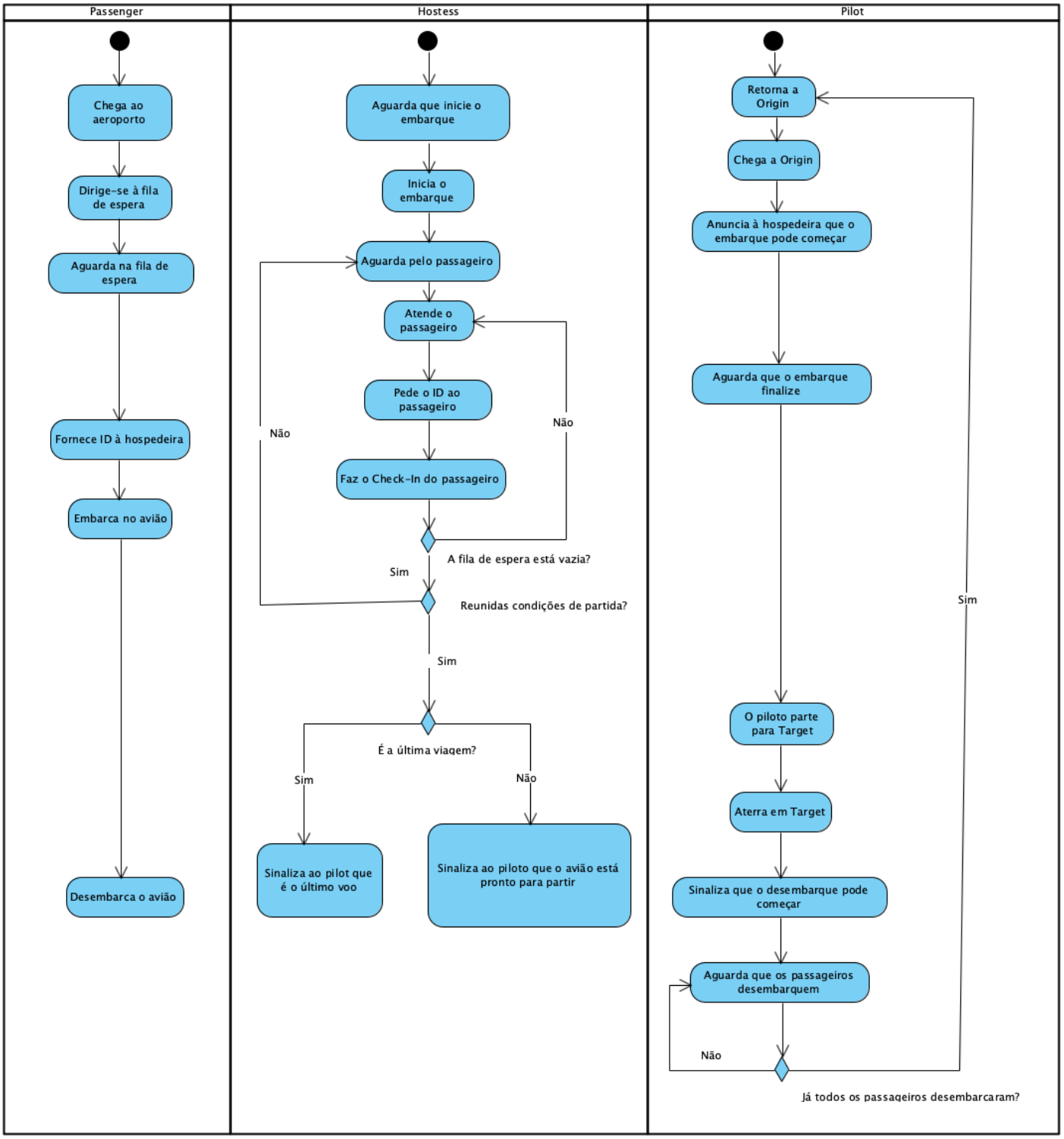
Para além dos processos referidos também teremos 2 locais/estruturas que vão atuar na simulação: a fila de espera dos passageiros e o avião. Os passageiros demoram a chegar ao aeroporto, sendo que este tempo de deslocação é gerado de forma pseudo-aleatória, ao chegarem devem se colocar na fila de espera, enquanto que aguardam que a hospedeira comece o processo de check-in e o consequente embarque. Ao serem atendidos os passageiros devem fornecer o seu ID, após fornecerem embarcam no avião. Quando o avião estiver pronto para descolar a hospedeira dá permissão ao piloto de partir. O piloto, de seguida, parte e os passageiros aguardam no avião. Ao chegar ao destino, o piloto inicia o desembarque e deixa os passageiros saírem do avião, quando o avião se encontrar vazio (sendo esta situação sinalizada pelo último passageiro a desembarcar) o piloto retorna ao aeroporto “Origin”.

O avião está pronto para descolar em direção a “Target” se uma destas três condições se verificar:

- O avião atingiu a sua lotação máxima (no nosso avião esta tem o valor de 10 pessoas)
- O avião atingiu a sua lotação mínima e já não há nenhum passageiro na fila de espera (no nosso avião esta tem o valor de 5 pessoas)
- Os passageiros que já se encontram embarcados são os últimos a serem transportados.

O resultado desta simulação consiste na impressão de um diário de bordo com as informações sobre os estados dos processos, o número de passageiros que estão na fila de espera, o número de passageiros que se encontram em voo e o número de passageiros que já embarcaram e chegaram ao seu destino pretendido.

Fluxograma do Problema



Fluxograma 1 - Fluxograma do problema

Estados dos Processos

Para clarificação dos estados utilizados para cada processo e o seu significado no código-solução, apresentamos um breve sumário para os estados dos processos “Pilot”, “Hostess” e “Passenger” (estes estados encontram-se definidos e explicados no ficheiro “probConst.h” fornecido pelo professor Nuno Lau).

Pilot

Nome do Estado	Valor do Estado	Significado do Estado
FLYING_BACK	0	O piloto regressa ao aeroporto “Origin”.
READY_FOR_BOARDING	1	O piloto sinaliza que já está pronto para o embarque.
WAITING_FOR_BOARDING	2	O piloto aguarda até que o embarque finalize
FLYING	3	O piloto leva os passageiros ao seu destino.
DROPING_PASSENGERS	4	O piloto deixa os passageiros no seu destino.

Tabela 1 - Sumário dos estados do processo “Pilot”

Hostess

Nome do Estado	Valor do Estado	Significado do Estado
WAIT_FOR_FLIGHT	0	A hospedeira aguarda que o avião esteja pronto para o embarque.
WAIT_FOR_PASSENGER	1	A hospedeira espera que o passageiro chegue.
CHECK_PASSPORT	2	A hospedeira verifica o passaporte do passageiro.
READY_TO_FLIGHT	3	A hospedeira sinaliza que o embarque terminou.

Tabela 2 - Sumário dos estados do processo “Hostess”

Passenger

Nome do Estado	Valor do Estado	Significado do Estado
GOING_TO_AIRPORT	0	O passageiro dirige-se ao aeroporto.
IN_QUEUE	1	O passageiro aguarda na fila de espera.
IN_FLIGHT	2	O passageiro está dentro do avião a voar.
AT_DESTINATION	3	O passageiro chega ao seu destino.

Tabela 3 - Sumário dos estados do processo “Passenger”

Estrutura de Dados

Com o mesmo intuito da secção anterior, para contextualizar o código fornecido, nomeadamente, as estruturas de dados, e a sua utilização nos tópicos das funções individuais de cada processo, apresentamos uma tabela com as variáveis inteiras, array de inteiros e uma variável booleana do problema e as suas definições:

Nome da Variável	Tipo	Utilidade
nPassengersInFlight[MAXNF[1]]	ponteiro para lista de unsigned int	Regista o número de passageiros de cada voo numa lista indexada pelo número do voo.
nFlight	unsigned int	Variável que guarda o número do voo que está a ser processado.
nPassInQueue	unsigned int	Variável que guarda o número de passageiros que se encontram na fila de espera.
nPassInFlight	unsigned int	Variável que guarda o número de passageiros que se encontram no voo.
totalPassBoarded	unsigned int	Variável que guarda o número total de passageiros já embarcados.
finished	bool	Variável que sinaliza o fim da sequência de voos.
passengerChecked	int	Variável que guarda o ID do último passageiro a fazer o check-in.
pilotStat	unsigned int	Variável que guarda o estado do processo Pilot.
hostessStat	unsigned int	Variável que guarda o estado do processo Hostess.
passengerStat[N][2]	ponteiro para unsigned int	Regista o estado do passageiro na lista dos estados dos N passageiros, indexada pelos IDs dos passageiros.

(1) No ficheiro “probConst.h”, a constante “MAXFN” está definida com o valor 10.
(2) No ficheiro “probConst.h”, a constante “N” está definida com o valor 21.

Tabela 4 - Sumário dos estados do processo “Passenger”

Abordagem ao problema

A nossa abordagem ao problema consistiu, primeiramente, em uma interpretação cuidadosa de toda a informação que nos foi proporcionada, através do enunciado e dos comentários no código-fonte, sobre as relações e as sincronizações necessárias para o funcionamento simultâneo dos processos. Foi de extrema importância a análise feita aos semáforos fornecidos e como estes deveriam ser manipulados pelos processos e para não perdermos o desenho mental de como tudo se relacionava, antes de desenvolver o código criámos a seguinte tabela que mapeia as relações intra-processos:

Semáforo	Processo que faz UP	Processo que faz DOWN	Utilidade do semáforo
mutex	Todos	Todos	Identifica a entrada/saída da região crítica
passengersInQueue	Passenger	Hostess	Usado pela hospedeira para aguardar pelos passageiros
passengersWaitInQueue	Hostess	Passenger	Usado pelos passageiros para aguardarem pela hospedeira.
passengersWaitInFlight	Pilot	Passenger	Usado pelos passageiros para aguardarem que o voo termine.
readyForBoarding	Pilot	Hostess	Usado pela hospedeira para esperar que o embarque comece.
readyToFlight	Hostess	Pilot	Usado pelo piloto para esperar que o embarque finalize.
idShown	Passenger	Hostess	Usado pela hospedeira para aguardar que o passageiro mostre identificação.
planeEmpty	Passenger	Pilot	Usado pelo piloto para esperar que o último passageiro desembarque.

Tabela 4 - Sumário dos semáforos do problema

Pilot (Funções)

Função flight(bool go)

A função flight(bool go) do piloto sinaliza o início do voo, podendo este ser em direção a “Target”, que neste caso implica que haja passageiros no avião, ou podendo ser em direção a “Origin”, levando o avião vazio. Nesta função, é acedida a memória partilhada para registar o estado do piloto consoante o valor booleano do parâmetro “go”. Caso este seja verdadeiro, o piloto fica no estado “FLYING”, caso contrário fica no estado “FLYING_BACK”. É utilizada a função "saveState", fornecida pelo professor Nuno Lau, para fazer o anúncio da mudança de estado do piloto, que leva à consequente impressão de uma linha com os mais recentes estados do diário de bordo,mencionado no último parágrafo da Introdução. Após feito o registo do novo estado, o processo sai da memória partilhada e é “adormecido” pelo comando “usleep” durante um período de tempo pseudo-aleatório (simula o tempo que o voo demora até alcançar o seu destino).

```c
static void flight(bool go)
{
    /* enter critical region */

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

    if (go)
    {
        sh->fSt.st.pilotStat = FLYING;
    }
    else
    {
        sh->fSt.st.pilotStat = FLYING_BACK;
    }
    saveState(nFic, &sh->fSt);

    /* exit critical region */

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

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

Figura 1 - Função flight(bool go)

Função signalReadyForBoarding()

Na função signalReadyForBoarding(), é suposto o piloto informar à hospedeira que o avião encontra-se pronto para começar o embarque. Para tal, é acedida novamente a memória partilhada para atualizar o estado do piloto para “READY_FOR_BOARDING”, sendo também incrementado o ID do voo. É anunciada a mudança no estado do piloto e também é impresso um anúncio que o embarque começou. Após sair da memória partilhada, o piloto sinaliza à hospedeira que o embarque já pode começar ao fazer semUp() ao semáforo “readyForBoarding”.

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

    sh->fSt.st.pilotStat = READY_FOR_BOARDING; // o piloto fica no estado READY_FOR_BOARDING
    sh->fSt.nFlight++;                         // incrementa o ID do voo
    saveState(nFic, &sh->fSt);                 // guarda o estado do piloto
    saveStartBoarding(nFic, &sh->fSt);         // emite anuncio a anunciar o começo do boarding

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

    // sinaliza à hospedeira que o boarding já pode começar
    if (semUp(semgid, sh->readyForBoarding) == -1)
    {
        perror("error on the up operation for semaphore access (PT)");
        exit(EXIT_FAILURE);
    }
}
```

Figura 2 - Função signalReadyForBoarding()

Função waitUntilReadyToFlight()

Na função waitUntilReadyToFlight(), o piloto aguarda que o embarque dos passageiros acabe. Novamente, a memória partilhada é acedida e é atualizado o estado do piloto para “WAITING_FOR_BOARDING”, sendo também anunciada esta atualização. Ao sair da

memória partilhada, o piloto sinaliza à hospedeira, ao fazer semDown() do semáforo “readyToFlight”, que está à espera que o embarque finalize.

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

    sh->fSt.st.pilotStat = WAITING_FOR_BOARDING;    // muda o estado do piloto para WAITING_FOR_BOARDING
    saveState(nFic, &sh->fSt);                      // guarda o estado do piloto

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

     // o piloto espera que o boarding acabe
    if (semDown(semgid, sh->readyToFlight) == -1)
    {
        perror("error on the down operation for semaphore access (PT)");
        exit(EXIT_FAILURE);
    }
}
```

Figura 3 - Função waitUntilReadyToFlight()

Função dropPassengersAtTarget()

Na função dropPassengersAtTarget(), o piloto dá permissão aos passageiros para começarem a desembarcar o avião, já que chegaram ao seu destino.

Inicialmente o processo acede à memória partilhada e anuncia a chegada do avião ao destino. Ainda na memória partilhada o estado do piloto é atualizado para “DROPING_PASSENGERS”, sendo anunciada a atualização e o piloto sinaliza a cada passageiro presente no voo que podem desembarcar, isto é feito através de um for-loop que itera “nPassInFlight” vezes o comando semUp() relativamente ao semáforo “passengerWaitInFlight”. Este loop vai “libertar” os processos-passageiros que ficaram retidos aquando das sucessivas semDown() do semáforo “passengerWaitInFlight” na função dos passageiros waitUntilDestination().

Após sair da memória partilhada o processo faz semDown() ao semáforo “planeEmpty”, aguardando que o último passageiro saia do avião. De seguida, após o avião estar vazio, o processo entra novamente na memória partilhada, fazendo o anúncio de que o avião encontra-se a retornar.

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


    saveFlightArrived(nFic, &sh->fSt); // emite anuncio que o avião chegou ao destino
    sh->fSt.st.pilotStat = DROPING_PASSENGERS;  // muda o estado do piloto para DROPING_PASSENGERS
    saveState(nFic, &sh->fSt);                  // guarda o estado


    // para cada passageiro dentro do avião, o piloto sinaliza que pode desembarcar
    for (int i = sh->fSt.nPassInFlight; i > 0; i--)
    {
        if (semUp(semgid, sh->passengersWaitInFlight) == -1)
        {
            perror("error on the up operation for semaphore access (PT)");
            exit(EXIT_FAILURE);
        }
    }
     /* exit critical region */
    if (semUp(semgid, sh->mutex) == -1)
    {
        perror("error on the up operation for semaphore access (PT)");
        exit(EXIT_FAILURE);
    }

    // o piloto espera que o último passageiro saia do avião
    if (semDown(semgid, sh->planeEmpty) == -1)
    {
        perror("error on the up operation for semaphore access (PT)");
        exit(EXIT_FAILURE);
    }

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

    saveFlightReturning(nFic, &sh->fSt);        // faz o anuncio do voo em retorno

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

Figura 4 - Função waitUntilReadyToFlight()

Hostess (Funções)

Função waitForNextFlight()

Na função waitForNextFlight a hospedeira deve aguardar pelo próximo voo. Mais uma vez, o processo da hospedeira entra na região crítica da memória partilhada e atualiza o seu estado para “WAIT_FOR_FLIGHT”, anunciando a atualização. Após sair da região crítica, faz semDown() do semáforo “readyForBoarding” sinalizando ao piloto que se encontra preparada para fazer o embarque, aguardando, deste modo, até autorização do processo piloto.

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

    sh->fSt.st.hostessStat = WAIT_FOR_FLIGHT; // muda o estado da hospedeira para WAIT_FOR_FLIGHT
    saveState(nFic, &sh->fSt);                // regista a mudança do estado

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

    // espera que o piloto sinalize que já pode começar o boarding
    if (semDown(semgid, sh->readyForBoarding) == -1)
    {
        perror("error on the down operation for semaphore access (HT)");
        exit(EXIT_FAILURE);
    }
}
```

Figura 5 - Função waitForNextFlight()

Função waitForPassenger()

Na função waitForPassenger() a hospedeira aguarda que os passageiros cheguem ao aeroporto. Ao entrar na região crítica, o processo hospedeira atualiza o seu estado para “WAIT_FOR_PASSENGER”, anunciando a atualização e após sair da região crítica, faz semDown() do semáforo “passengersInQueue”, esperando que os passageiros quando chegarem à fila de espera, façam semUp() do semáforo em questão.

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

    sh->fSt.st.hostessStat = WAIT_FOR_PASSENGER; // muda o estado da hospedeira para WAIT_FOR_PASSENGER
    saveState(nFic, &sh->fSt);                   // guarda o estado

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

    // Espera que os passageiros chegam à fila de espera
    if (semDown(semgid, sh->passengersInQueue) == -1)
    {
        perror("error on the down operation for semaphore access (HT)");
        exit(EXIT_FAILURE);
    }
}
```

Figura 6 - Função waitForPassenger()

Função checkPassport()

Na função checkPassport() a hospedeira verifica o passaporte do passageiro e espera que o mesmo mostre o seu ID. Após o procedimento finalizar, a hospedeira faz o “check-in” do passageiro com sucesso, permitindo que este embarque no avião. Esta função tem um valor booleano de retorno que indica se o passageiro acabado de embarcar é o último passageiro a embarcar o avião, sendo que este é o último se:

- O avião atingiu a sua lotação máxima;
- O avião atingiu a sua lotação mínima e já não existe nenhum passageiro na fila de espera
- Já não há mais passageiros em geral para embarcar.

Inicialmente na função é definida a variável booleana anteriormente referida e de seguida é feito o semUp() do semáforo “passengersWaitInQueue” que se traduz ao atendimento de um passageiro. De seguida, o processo hospedeira entra na região crítica na qual é atualizado e anunciado o seu estado para “CHECK_PASSPORT”. Após sair da região crítica faz o semDown() do semáforo “idShown”, esperando que o passageiro forneça então o seu ID. Ao receber o ID do passageiro, entra novamente na região crítica e faz o seguinte:

- Decrementa o número de passageiros que se encontra na fila de espera;
- Incrementa o número de passageiros que se encontram embarcados no avião;
- Incrementa o número de passageiros que já embarcaram no total;
- Anuncia que o passageiro com o ID fornecido fez o check-in;
- Anuncia as mudanças dos dados decrementados/incrementados;

Ainda na região crítica faz a avaliação das três condições mencionadas para identificar se o último passageiro a fazer o check-in é de facto o último passageiro a embarcar. Após sair da região crítica, no fim da função, esta faz o retorno da variável booleana.

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

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

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


    sh->fSt.st.hostessStat = CHECK_PASSPORT; // atualiza o estado da hospedeira para CHECK_PASSAPO
    saveState(nFic, &sh->fSt);               // guarda o estado

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

    // espera que o passageiro forneça o ID
    if (semDown(semgid, sh->idShown) == -1)
    {
        perror("error on the down operation for semaphore access (HT)");
        exit(EXIT_FAILURE);
    }
    /* enter critical region */
    if (semDown(semgid, sh->mutex) == -1)
    {
        perror("error on the down operation for semaphore access (HT)");
        exit(EXIT_FAILURE);
    }

    sh->fSt.nPassInQueue--;               // decrementa a fila de espera
    sh->fSt.nPassInFlight++;              // incrementa a lotação no avião
    sh->fSt.totalPassBoarded++;           // incrementa o registo de já embarcados no total
savePassengerChecked(nFic, &sh->fSt); // imprime a mensagem de que o passageiro deu checked-in
saveState(nFic, &sh->fSt);            // guarda os valores dos contadores

// Verifica se o avião está pronto para partir
if (nPassengersInFlight() == MAXFC)     // se a lotação do avião chegou ao seu máximo
{
    last = true;
}
else if (nPassengersInFlight() >= MINFC && nPassengersInQueue() == 0){      // já há numero minimo de lotação e ninguem na fila de espera
    last = true;
}
else if (sh->fSt.totalPassBoarded == N){              // já todos os passageiros embarcaram
    last = true;
}
else
{
    last = false;
}

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

return last;
}
```

Figura 7 - Função checkPassport()

Função signalReadyToFlight()

A função signalReadyToFlight() é a função utilizada pela hospedeira para sinalizar que o avião está pronto para partir. Como tem acontecido nas funções dos processos, inicialmente o processo hospedeira entra na região crítica, atualizando e anunciando o seu estado para “READY_TO_FLIGHT”. Também, na região crítica, regista o número de passageiros que o avião com ID “nFlight” leva, fazendo o anúncio que o avião descolou. É avaliado se o avião acabado de partir será o último da sequência de voos, nomeadamente na avaliação se já todos os N passageiros esperados já embarcaram. Após sair da região crítica, a hospedeira faz semUp do semáforo “readyToFlight”, sinalizando ao piloto que o avião já está pronto para voar.

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

    sh->fSt.st.hostessStat = READY_TO_FLIGHT; // atualiza o estado da hospedeira para READY_TO_FLIGHT
    saveState(nFic, &sh->fSt); // atualiza os dados

    sh->fSt.nPassengersInFlight[sh->fSt.nFlight - 1] = sh->fSt.nPassInFlight;      // regista o número de passageiros que o avião nFlight leva.
    saveFlightDeparted(nFic, &sh->fSt);         // emite o anúncio que o voo descolou

    // avalia se este será o último voo necessário
    if (sh->fSt.totalPassBoarded == N)
    {
        sh->fSt.finished = true;
    }

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

    // sinaliza ao piloto que já está pronto para voar
    if (semUp(semgid, sh->readyToFlight))
    {
        perror("error on the up operation for semaphore access (HT)");
        exit(EXIT_FAILURE);
    }
}
```

Figura 8 - Função signalReadyToFlight()

Passenger (Funções)

Função waitInQueue(unsigned int passengerId)

Na função waitInQueue o passageiro espera que a sua vez chegue para fazer o check-in, tem como parâmetro o ID do passageiro. Inicialmente o processo passageiro entra na região crítica, incrementa o número de passageiros que estão na fila de espera e atualiza e anuncia o seu estado para “IN_QUEUE”. Após sair da região crítica, faz semUp do semáforo “passengersInQueue” para sinalizar à hospedeira que já se encontra na fila de espera, de seguida, faz semDown do semáforo “passengerWaitInQueue”, aguardando desta forma que chegue a sua vez de ser atendido.
Ao ser atendido, o processo passageiro entra na região crítica, fornece o seu ID à hospedeira e atualiza e anuncia o seu estado para “IN_FLIGHT”. Já fora da região crítica, faz semUp do semáforo “idShown”, sinalizando à hospedeira que já mostrou o seu ID, podendo desta forma entrar no avião.

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

    sh->fSt.nPassInQueue++;                         // incrementa o número de passageiros que estão na fila de espera
    sh->fSt.st.passengerStat[passengerId] = IN_QUEUE; // atualiza o estado do passageiro
    saveState(nFic, &sh->fSt);                      // regista o estado do passageiro

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

    // Sinaliza à hospedeira que já há passageiros na fila de espera
    if (semUp(semgid, sh->passengersInQueue) == -1)
    {
        perror("error on the up operation for semaphore access (PG)");
        exit(EXIT_FAILURE);
    }

    // aguarda na fila de espera até ser atendido pela hospedeira
    if (semDown(semgid, sh->passengersWaitInQueue) == -1)
    {
        perror("error on the down operation for semaphore access (PG)");
        exit(EXIT_FAILURE);
    }

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

        sh->fSt.passengerChecked = passengerId;           // o passageiro fornece o seu id
        sh->fSt.st.passengerStat[passengerId] = IN_FLIGHT; // entra no aviao
        saveState(nFic, &sh->fSt);                        // regista o estado


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

// sinaliza à hospedeira que mostrou o ID e assim pode entrar no avião, finalizando o check-in
if (semUp(semgid, sh->idShown))
{
    perror("error on the up operation for semaphore access (PG)");
    exit(EXIT_FAILURE);
}
```

Figura 8 - Função waitInQueue()

Função waitUntilDestination(unsigned int passengerId)

A função waitUntilDestination é usada pelo passageiro para aguardar que o voo termine e possa depois desembarcar, esta função à semelhança da descrita anteriormente também tem como parâmetro o ID do passageiro.

O passageiro faz semDown do semáforo “passengersWaitInFlight”, sinalizando ao piloto que se encontra a aguardar o desembarque. Após ser dada permissão para desembarcar, entra na região crítica e atualiza o seu estado para “AT_DESTINATION”, decrementando também o número de passageiros que se encontram no avião. Caso este decremento resulte num número de passageiros no avião nulo, então o passageiro faz semUp do semáforo “planeEmpty”, informando ao piloto que o ele próprio, o passageiro, é o último a desembarcar.

```
static void waitUntilDestination(unsigned int passengerId)
{
    // sinaliza ao piloto que está a aguardar no avião
    semDown(semgid, sh->passengersWaitInFlight);

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

    sh->fSt.st.passengerStat[passengerId] = AT_DESTINATION;     // o passageiro chegou ao seu destino
    sh->fSt.nPassInFlight--;                                    // e consequentemente sai do avião

    // caso o passageiro observe que é o ultimo a sair do aviáo, então avisa ao piloto que o avião encontra-se vazio
    if (sh->fSt.nPassInFlight == 0)
    {
        if (semUp(semgid, sh->planeEmpty) == -1)
        {
            perror("error on the up operation for semaphore access (PG)");
            exit(EXIT_FAILURE);
        }
    }

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

Figura 9 - Função waitUntilDestination()

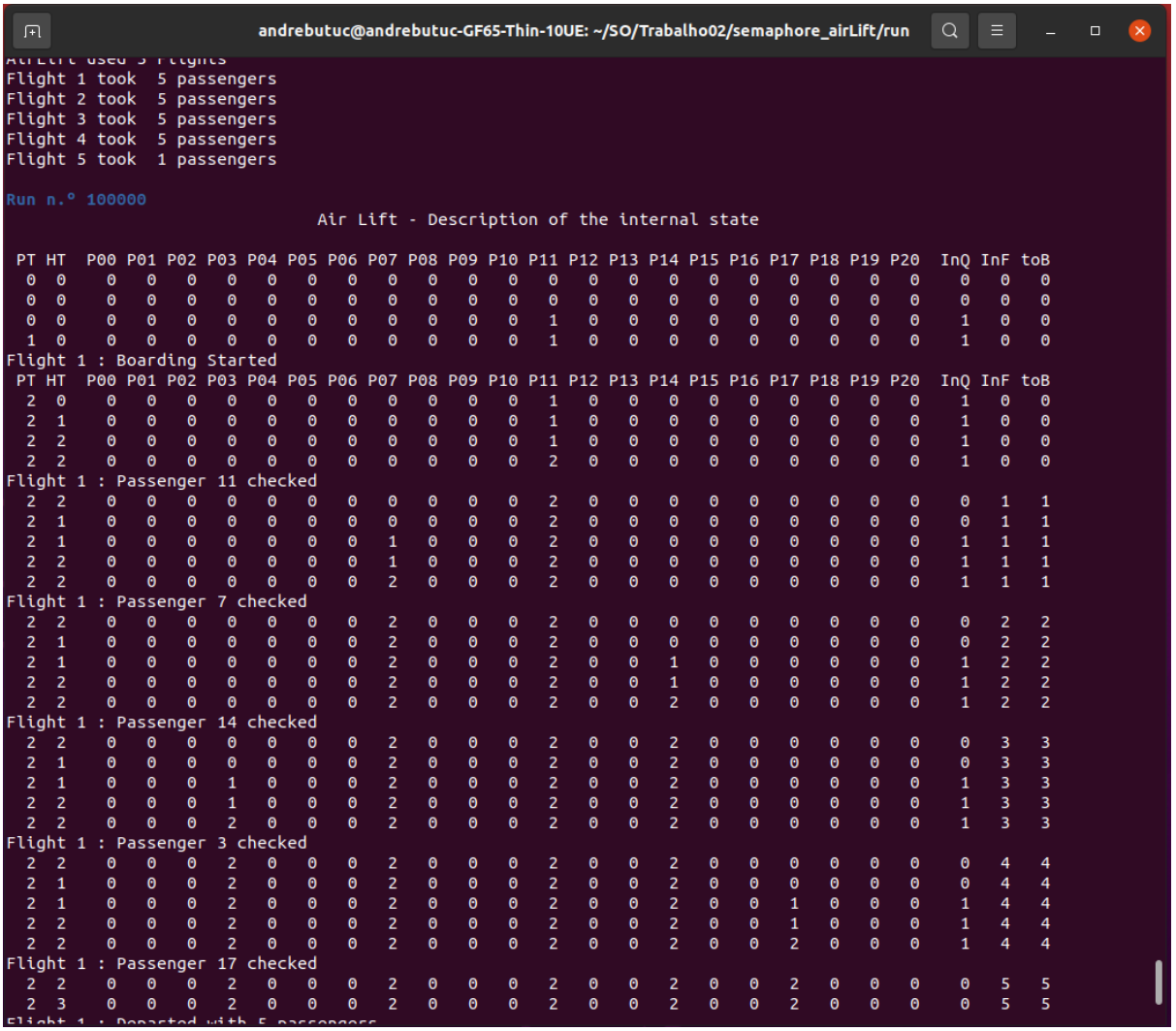
Validação da solução

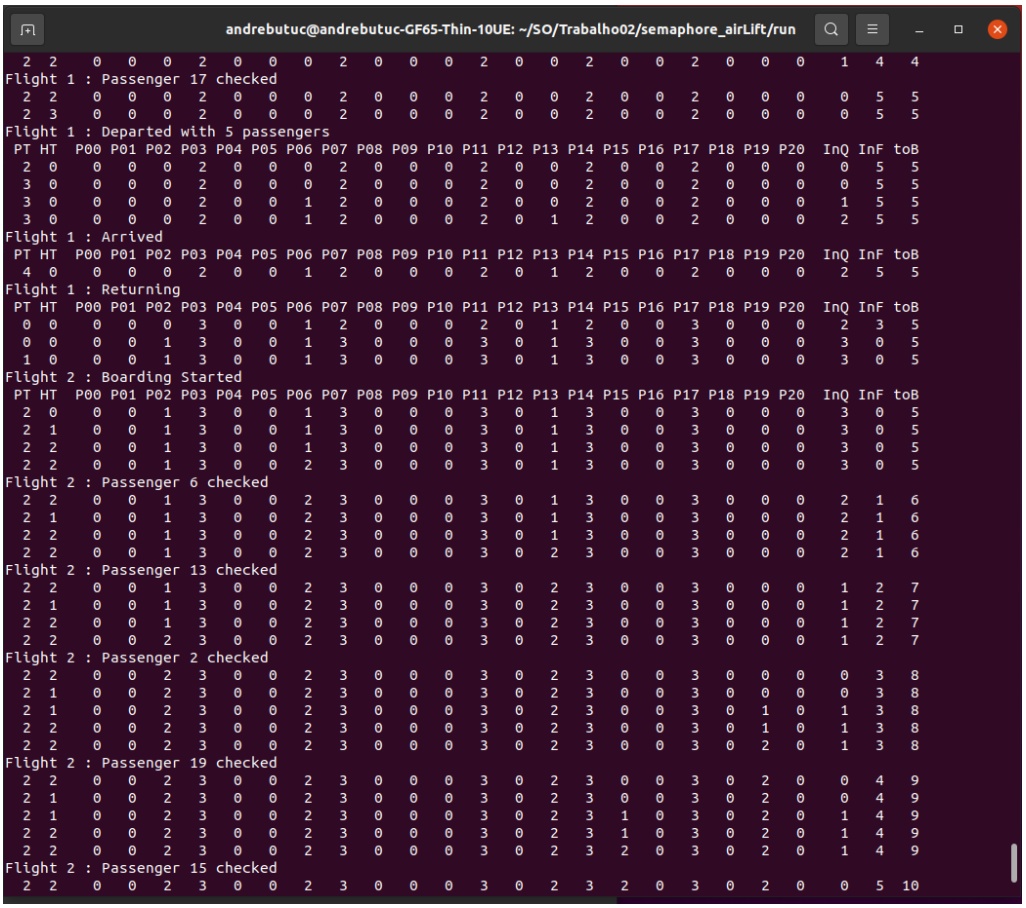
Para validarmos a solução obtida fizémos três testes:

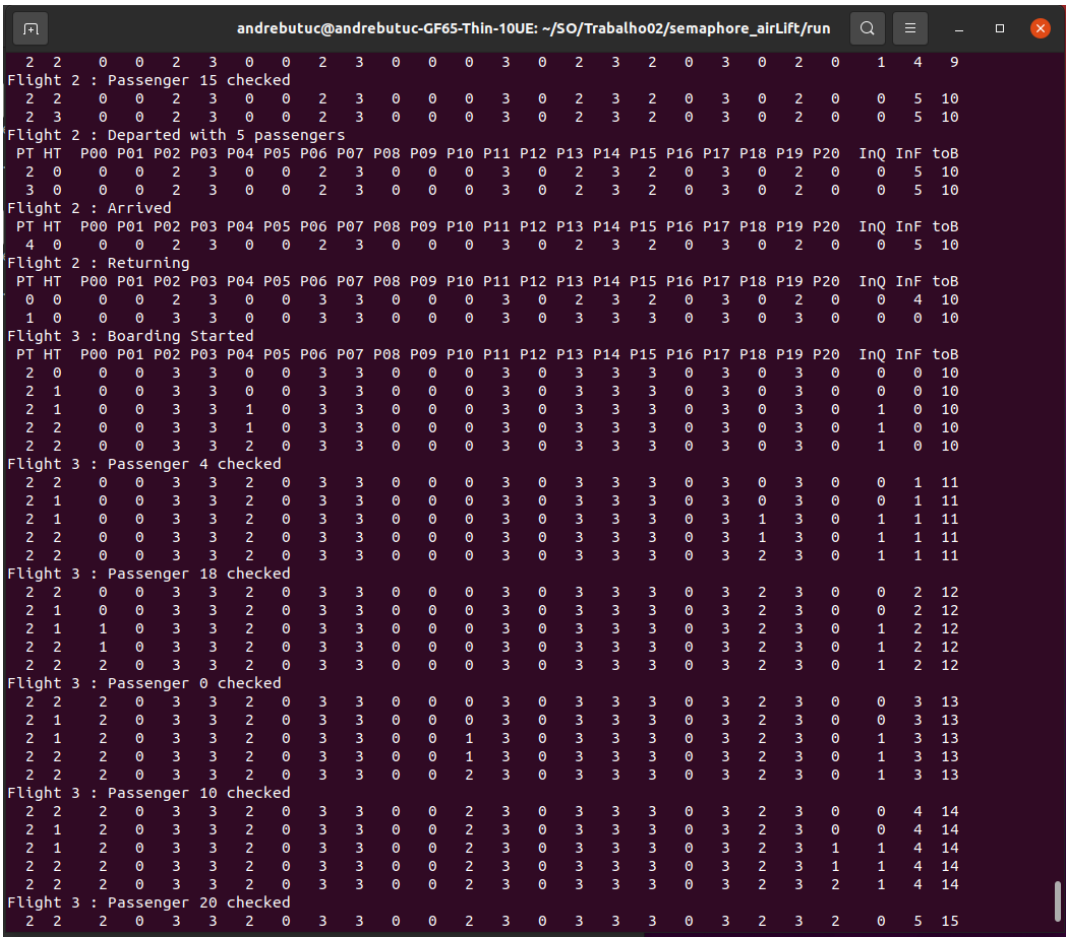
- Verificação manual das transições de estado entre os processos;
- Testagem de situações vulneráveis a deadlocks;
- Manipulação do ficheiro das constantes para averiguar situações diferentes.

Inicialmente, apesar de ser algo trivial, validámos manualmente os resultados de uma só simulação (obtida através da chamada do executável ./promSemSharedMemAirLift) consoante as tabelas definidas para as entidades que manipulam os semáforos. Após quatro análises manuais feitas de duas simulações diferentes (duas análises por cada simulação feita em simultâneo pelos dois integrantes do grupo, sistema semelhante à contagem dupla dos votos eleitorais para assegurar que não há erros humanos inerentes), concluímos que a solução estava a originar os resultados esperados.

Por último, para avaliar se havia ocorrência de deadlocks, corremos 100 000 simulações duas vezes (totalizando no fim 200 000 simulações) com o auxílio do bash script “run.sh”:




andrebutuc@andrebutuc-GF65-Thin-10UE: ~/SO/Trabalho02/semaphore_airLift/run
2 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 2 0 0 2 0 0 0 1 4 4
Flight 1 : Passenger 17 checked
2 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 2 0 0 2 0 0 0 0 5 5
2 3 0 0 0 2 0 0 0 2 0 0 0 2 0 0 2 0 0 2 0 0 0 0 5 5
Flight 1 : 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 0 0 0 2 0 0 0 2 0 0 0 2 0 0 2 0 0 2 0 0 0 0 5 5
3 0 0 0 0 2 0 0 0 2 0 0 0 2 0 0 2 0 0 2 0 0 0 0 5 5
3 0 0 0 0 2 0 0 1 2 0 0 0 2 0 0 2 0 0 2 0 0 0 1 5 5
3 0 0 0 0 2 0 0 1 2 0 0 0 2 0 1 2 0 0 2 0 0 0 2 5 5
Flight 1 : Arrived
PT HT P00 P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20 InQ InF toB
4 0 0 0 0 2 0 0 1 2 0 0 0 2 0 1 2 0 0 2 0 0 0 2 5 5
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
0 0 0 0 0 3 0 0 1 2 0 0 0 2 0 1 2 0 0 3 0 0 0 2 3 5
0 0 0 0 1 3 0 0 1 3 0 0 0 3 0 1 3 0 0 3 0 0 0 3 0 5
1 0 0 0 1 3 0 0 1 3 0 0 0 3 0 1 3 0 0 3 0 0 0 3 0 5
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
2 0 0 0 1 3 0 0 1 3 0 0 0 3 0 1 3 0 0 3 0 0 0 3 0 5
2 1 0 0 1 3 0 0 1 3 0 0 0 3 0 1 3 0 0 3 0 0 0 3 0 5
2 2 0 0 1 3 0 0 1 3 0 0 0 3 0 1 3 0 0 3 0 0 0 3 0 5
2 2 0 0 1 3 0 0 2 3 0 0 0 3 0 1 3 0 0 3 0 0 0 3 0 5
Flight 2 : Passenger 6 checked
2 2 0 0 1 3 0 0 2 3 0 0 0 3 0 1 3 0 0 3 0 0 0 2 1 6
2 1 0 0 1 3 0 0 2 3 0 0 0 3 0 1 3 0 0 3 0 0 0 2 1 6
2 2 0 0 1 3 0 0 2 3 0 0 0 3 0 1 3 0 0 3 0 0 0 2 1 6
2 2 0 0 1 3 0 0 2 3 0 0 0 3 0 2 3 0 0 3 0 0 0 2 1 6
Flight 2 : Passenger 13 checked
2 2 0 0 1 3 0 0 2 3 0 0 0 3 0 2 3 0 0 3 0 0 0 1 2 7
2 1 0 0 1 3 0 0 2 3 0 0 0 3 0 2 3 0 0 3 0 0 0 1 2 7
2 2 0 0 1 3 0 0 2 3 0 0 0 3 0 2 3 0 0 3 0 0 0 1 2 7
2 2 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 0 0 3 0 0 0 1 2 7
Flight 2 : Passenger 2 checked
2 2 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 0 0 3 0 0 0 0 3 8
2 1 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 0 0 3 0 0 0 0 3 8
2 1 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 0 0 3 0 1 0 1 3 8
2 2 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 0 0 3 0 1 0 1 3 8
2 2 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 0 0 3 0 2 0 1 3 8
Flight 2 : Passenger 19 checked
2 2 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 0 0 3 0 2 0 0 4 9
2 1 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 0 0 3 0 2 0 0 4 9
2 1 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 1 0 3 0 2 0 1 4 9
2 2 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 1 0 3 0 2 0 1 4 9
2 2 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 2 0 3 0 2 0 1 4 9
Flight 2 : Passenger 15 checked
2 2 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 2 0 3 0 2 0 0 5 10


andrebutuc@andrebutuc-GF65-Thin-10UE: ~/SO/Trabalho02/semaphore_airLift/run
2 2 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 2 0 3 0 2 0 1 4 9
Flight 2 : Passenger 15 checked
2 2 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 2 0 3 0 2 0 0 5 10
2 3 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 2 0 3 0 2 0 0 5 10
Flight 2 : 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 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 2 0 3 0 2 0 0 5 10
3 0 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 2 0 3 0 2 0 0 5 10
Flight 2 : Arrived
PT HT P00 P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20 InQ InF toB
4 0 0 0 2 3 0 0 2 3 0 0 0 3 0 2 3 2 0 3 0 2 0 0 5 10
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
0 0 0 0 2 3 0 0 3 3 0 0 0 3 0 2 3 2 0 3 0 2 0 0 4 10
1 0 0 0 3 3 0 0 3 3 0 0 0 3 0 3 3 3 0 3 0 3 0 0 0 10
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
2 0 0 0 3 3 0 0 3 3 0 0 0 3 0 3 3 3 0 3 0 3 0 0 0 10
2 1 0 0 3 3 0 0 3 3 0 0 0 3 0 3 3 3 0 3 0 3 0 0 0 10
2 1 0 0 3 3 1 0 3 3 0 0 0 3 0 3 3 3 0 3 0 3 0 1 0 10
2 2 0 0 3 3 1 0 3 3 0 0 0 3 0 3 3 3 0 3 0 3 0 1 0 10
2 2 0 0 3 3 2 0 3 3 0 0 0 3 0 3 3 3 0 3 0 3 0 1 0 10
Flight 3 : Passenger 4 checked
2 2 0 0 3 3 2 0 3 3 0 0 0 3 0 3 3 3 0 3 0 3 0 0 1 11
2 1 0 0 3 3 2 0 3 3 0 0 0 3 0 3 3 3 0 3 0 3 0 0 1 11
2 1 0 0 3 3 2 0 3 3 0 0 0 3 0 3 3 3 0 3 1 3 0 1 1 11
2 2 0 0 3 3 2 0 3 3 0 0 0 3 0 3 3 3 0 3 1 3 0 1 1 11
2 2 0 0 3 3 2 0 3 3 0 0 0 3 0 3 3 3 0 3 2 3 0 1 1 11
Flight 3 : Passenger 18 checked
2 2 0 0 3 3 2 0 3 3 0 0 0 3 0 3 3 3 0 3 2 3 0 0 2 12
2 1 0 0 3 3 2 0 3 3 0 0 0 3 0 3 3 3 0 3 2 3 0 0 2 12
2 1 1 0 3 3 2 0 3 3 0 0 0 3 0 3 3 3 0 3 2 3 0 1 2 12
2 2 1 0 3 3 2 0 3 3 0 0 0 3 0 3 3 3 0 3 2 3 0 1 2 12
2 2 2 0 3 3 2 0 3 3 0 0 0 3 0 3 3 3 0 3 2 3 0 1 2 12
Flight 3 : Passenger 0 checked
2 2 2 0 3 3 2 0 3 3 0 0 0 3 0 3 3 3 0 3 2 3 0 0 3 13
2 1 2 0 3 3 2 0 3 3 0 0 0 3 0 3 3 3 0 3 2 3 0 0 3 13
2 1 2 0 3 3 2 0 3 3 0 0 1 3 0 3 3 3 0 3 2 3 0 1 3 13
2 2 2 0 3 3 2 0 3 3 0 0 1 3 0 3 3 3 0 3 2 3 0 1 3 13
2 2 2 0 3 3 2 0 3 3 0 0 2 3 0 3 3 3 0 3 2 3 0 1 3 13
Flight 3 : Passenger 10 checked
2 2 2 0 3 3 2 0 3 3 0 0 2 3 0 3 3 3 0 3 2 3 0 0 4 14
2 1 2 0 3 3 2 0 3 3 0 0 2 3 0 3 3 3 0 3 2 3 0 0 4 14
2 1 2 0 3 3 2 0 3 3 0 0 2 3 0 3 3 3 0 3 2 3 1 1 4 14
2 2 2 0 3 3 2 0 3 3 0 0 2 3 0 3 3 3 0 3 2 3 1 1 4 14
2 2 2 0 3 3 2 0 3 3 0 0 2 3 0 3 3 3 0 3 2 3 2 1 4 14
Flight 3 : Passenger 20 checked
2 2 2 0 3 3 2 0 3 3 0 0 2 3 0 3 3 3 0 3 2 3 2 0 5 15

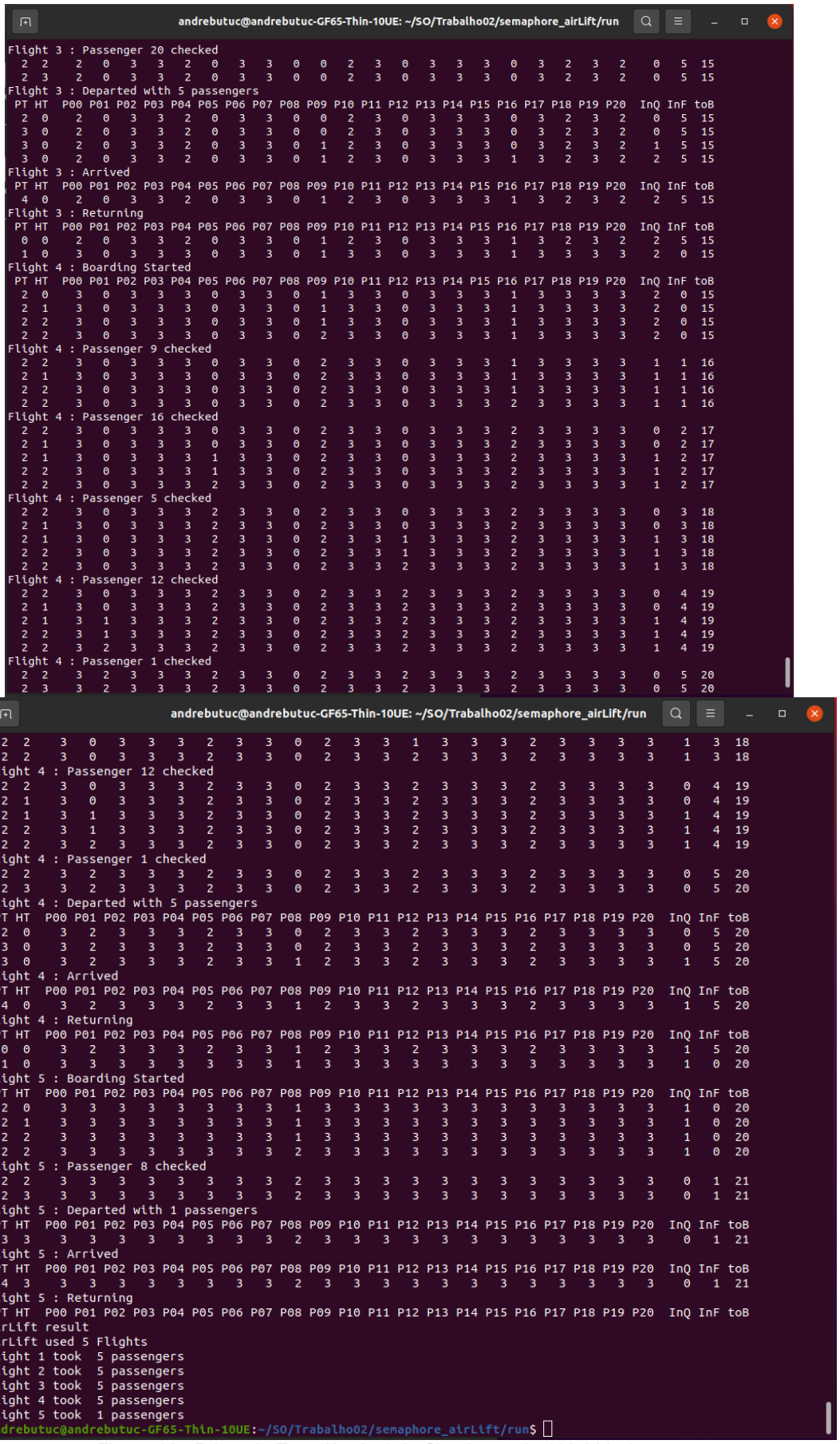
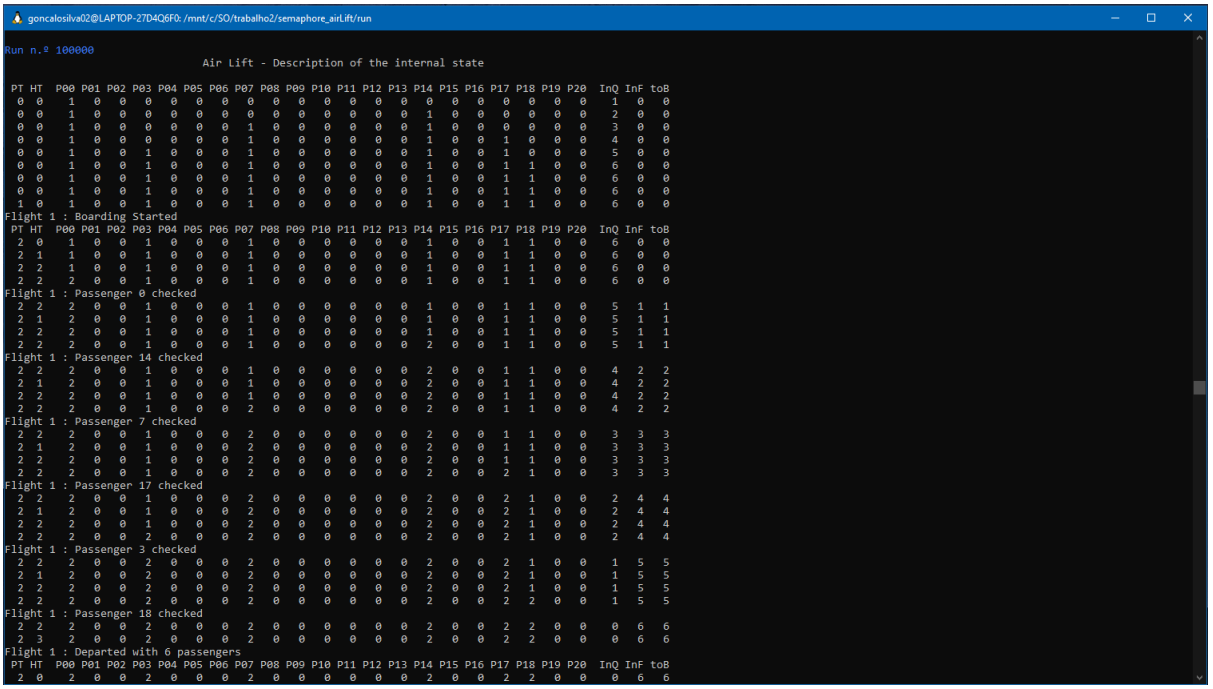


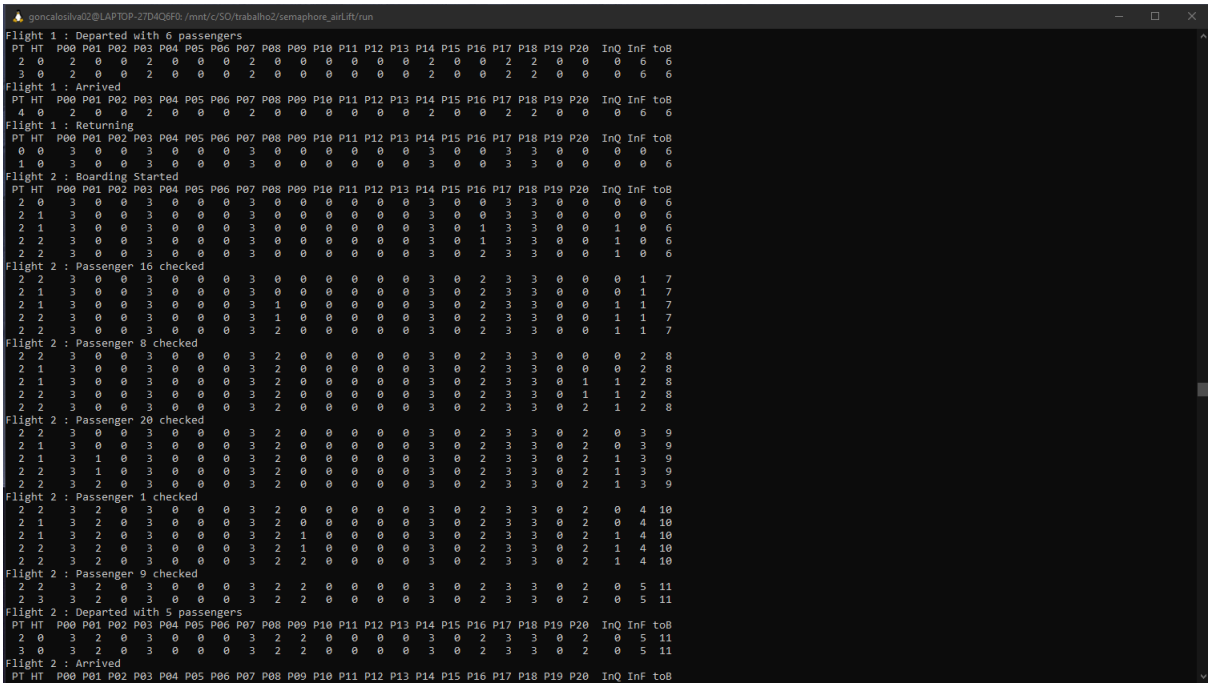
Figura 10 - Prints da Experiência 1 da Simulação de 100 000 situações



```
goncalosilva02@LAPTOP-27D4Q6F0: /mnt/c/SO/trabalho2/semaphore_airLift/run

Run n.º 100000
                        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    1   0   0   0   0   0   0   0   0   0   0   0   0   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   0   1   0   0   0   0   0   2   0   0
  0  0    1   0   0   0   0   0   0   0   1   0   0   0   0   0   1   0   0   0   0   0   3   0   0
  0  0    1   0   0   0   0   0   0   0   1   0   0   0   0   0   1   0   0   1   0   0   4   0   0
  0  0    1   0   0   1   0   0   0   0   1   0   0   0   0   0   1   0   0   1   0   0   5   0   0
  0  0    1   0   0   1   0   0   0   0   1   0   0   0   0   0   1   0   0   1   1   0   0   6   0   0
  0  0    1   0   0   1   0   0   0   0   1   0   0   0   0   0   1   0   0   1   1   0   0   6   0   0
  0  0    1   0   0   1   0   0   0   0   1   0   0   0   0   0   1   0   0   1   1   0   0   6   0   0
  1  0    1   0   0   1   0   0   0   1   0   0   0   0   0   0   1   0   0   1   1   0   0   6   0   0
Flight 1 : Boarding Started
 PT HT  P00 P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20  InQ InF toB
  2  0    1   0   0   1   0   0   0   1   0   0   0   0   0   0   1   0   0   1   1   0   0   6   0   0
  2  1    1   0   0   1   0   0   0   1   0   0   0   0   0   0   1   0   0   1   1   0   0   6   0   0
  2  2    1   0   0   1   0   0   0   1   0   0   0   0   0   0   1   0   0   1   1   0   0   6   0   0
  2  2    2   0   0   1   0   0   0   1   0   0   0   0   0   0   1   0   0   1   1   0   0   6   0   0
Flight 1 : Passenger 0 checked
  2  2    2   0   0   1   0   0   0   1   0   0   0   0   0   0   1   0   0   1   1   0   0   5   1   1
  2  1    2   0   0   1   0   0   0   1   0   0   0   0   0   0   1   0   0   1   1   0   0   5   1   1
  2  2    2   0   0   1   0   0   0   1   0   0   0   0   0   0   1   0   0   1   1   0   0   5   1   1
  2  2    2   0   0   1   0   0   0   1   0   0   0   0   0   0   2   0   0   1   1   0   0   5   1   1
Flight 1 : Passenger 14 checked
  2  2    2   0   0   1   0   0   0   1   0   0   0   0   0   0   2   0   0   1   1   0   0   4   2   2
  2  1    2   0   0   1   0   0   0   1   0   0   0   0   0   0   2   0   0   1   1   0   0   4   2   2
  2  2    2   0   0   1   0   0   0   1   0   0   0   0   0   0   2   0   0   1   1   0   0   4   2   2
  2  2    2   0   0   1   0   0   0   2   0   0   0   0   0   0   2   0   0   1   1   0   0   4   2   2
Flight 1 : Passenger 7 checked
  2  2    2   0   0   1   0   0   0   2   0   0   0   0   0   0   2   0   0   1   1   0   0   3   3   3
  2  1    2   0   0   1   0   0   0   2   0   0   0   0   0   0   2   0   0   1   1   0   0   3   3   3
  2  2    2   0   0   1   0   0   0   2   0   0   0   0   0   0   2   0   0   1   1   0   0   3   3   3
  2  2    2   0   0   1   0   0   0   2   0   0   0   0   0   0   2   0   0   2   1   0   0   3   3   3
Flight 1 : Passenger 17 checked
  2  2    2   0   0   1   0   0   0   2   0   0   0   0   0   0   2   0   0   2   1   0   0   2   4   4
  2  1    2   0   0   1   0   0   0   2   0   0   0   0   0   0   2   0   0   2   1   0   0   2   4   4
  2  2    2   0   0   1   0   0   0   2   0   0   0   0   0   0   2   0   0   2   1   0   0   2   4   4
  2  2    2   0   0   2   0   0   0   2   0   0   0   0   0   0   2   0   0   2   1   0   0   2   4   4
Flight 1 : Passenger 3 checked
  2  2    2   0   0   2   0   0   0   2   0   0   0   0   0   0   2   0   0   2   1   0   0   1   5   5
  2  1    2   0   0   2   0   0   0   2   0   0   0   0   0   0   2   0   0   2   1   0   0   1   5   5
  2  2    2   0   0   2   0   0   0   2   0   0   0   0   0   0   2   0   0   2   1   0   0   1   5   5
  2  2    2   0   0   2   0   0   0   2   0   0   0   0   0   0   2   0   0   2   2   0   0   1   5   5
Flight 1 : Passenger 18 checked
  2  2    2   0   0   2   0   0   0   2   0   0   0   0   0   0   2   0   0   2   2   0   0   0   6   6
  2  3    2   0   0   2   0   0   0   2   0   0   0   0   0   0   2   0   0   2   2   0   0   0   6   6
Flight 1 : 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   0   0   2   0   0   0   2   0   0   0   0   0   0   2   0   0   2   2   0   0   0   6   6
```



```
goncalosilva02@LAPTOP-27D4Q6F0: /mnt/c/SO/trabalho2/semaphore_airLift/run

Flight 1 : 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   0   0   2   0   0   0   2   0   0   0   0   0   0   2   0   0   2   2   0   0   0   6   6
  3  0    2   0   0   2   0   0   0   2   0   0   0   0   0   0   2   0   0   2   2   0   0   0   6   6
Flight 1 : Arrived
 PT HT  P00 P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20  InQ InF toB
  4  0    2   0   0   2   0   0   0   2   0   0   0   0   0   0   2   0   0   2   2   0   0   0   6   6
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
  0  0    3   0   0   3   0   0   0   3   0   0   0   0   0   0   3   0   0   3   3   0   0   0   0   6
  1  0    3   0   0   3   0   0   0   3   0   0   0   0   0   0   3   0   0   3   3   0   0   0   0   6
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
  2  0    3   0   0   3   0   0   0   3   0   0   0   0   0   0   3   0   0   3   3   0   0   0   0   6
  2  1    3   0   0   3   0   0   0   3   0   0   0   0   0   0   3   0   0   3   3   0   0   0   0   6
  2  1    3   0   0   3   0   0   0   3   0   0   0   0   0   0   3   0   1   3   3   0   0   1   0   6
  2  2    3   0   0   3   0   0   0   3   0   0   0   0   0   0   3   0   1   3   3   0   0   1   0   6
  2  2    3   0   0   3   0   0   0   3   0   0   0   0   0   0   3   0   2   3   3   0   0   1   0   6
Flight 2 : Passenger 16 checked
  2  2    3   0   0   3   0   0   0   3   0   0   0   0   0   0   3   0   2   3   3   0   0   0   1   7
  2  1    3   0   0   3   0   0   0   3   0   0   0   0   0   0   3   0   2   3   3   0   0   0   1   7
  2  1    3   0   0   3   0   0   0   3   1   0   0   0   0   0   3   0   2   3   3   0   0   1   1   7
  2  2    3   0   0   3   0   0   0   3   1   0   0   0   0   0   3   0   2   3   3   0   0   1   1   7
  2  2    3   0   0   3   0   0   0   3   2   0   0   0   0   0   3   0   2   3   3   0   0   1   1   7
Flight 2 : Passenger 8 checked
  2  2    3   0   0   3   0   0   0   3   2   0   0   0   0   0   3   0   2   3   3   0   0   0   2   8
  2  1    3   0   0   3   0   0   0   3   2   0   0   0   0   0   3   0   2   3   3   0   0   0   2   8
  2  1    3   0   0   3   0   0   0   3   2   0   0   0   0   0   3   0   2   3   3   0   1   1   2   8
  2  2    3   0   0   3   0   0   0   3   2   0   0   0   0   0   3   0   2   3   3   0   1   1   2   8
  2  2    3   0   0   3   0   0   0   3   2   0   0   0   0   0   3   0   2   3   3   0   2   1   2   8
Flight 2 : Passenger 20 checked
  2  2    3   0   0   3   0   0   0   3   2   0   0   0   0   0   3   0   2   3   3   0   2   0   3   9
  2  1    3   0   0   3   0   0   0   3   2   0   0   0   0   0   3   0   2   3   3   0   2   0   3   9
  2  1    3   1   0   3   0   0   0   3   2   0   0   0   0   0   3   0   2   3   3   0   2   1   3   9
  2  2    3   1   0   3   0   0   0   3   2   0   0   0   0   0   3   0   2   3   3   0   2   1   3   9
  2  2    3   2   0   3   0   0   0   3   2   0   0   0   0   0   3   0   2   3   3   0   2   1   3   9
Flight 2 : Passenger 1 checked
  2  2    3   2   0   3   0   0   0   3   2   0   0   0   0   0   3   0   2   3   3   0   2   0   4  10
  2  1    3   2   0   3   0   0   0   3   2   0   0   0   0   0   3   0   2   3   3   0   2   0   4  10
  2  1    3   2   0   3   0   0   0   3   2   1   0   0   0   0   3   0   2   3   3   0   2   1   4  10
  2  2    3   2   0   3   0   0   0   3   2   1   0   0   0   0   3   0   2   3   3   0   2   1   4  10
  2  2    3   2   0   3   0   0   0   3   2   2   0   0   0   0   3   0   2   3   3   0   2   1   4  10
Flight 2 : Passenger 9 checked
  2  2    3   2   0   3   0   0   0   3   2   2   0   0   0   0   3   0   2   3   3   0   2   0   5  11
  2  3    3   2   0   3   0   0   0   3   2   2   0   0   0   0   3   0   2   3   3   0   2   0   5  11
Flight 2 : 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   2   0   3   0   0   0   3   2   2   0   0   0   0   3   0   2   3   3   0   2   0   5  11
  3  0    3   2   0   3   0   0   0   3   2   2   0   0   0   0   3   0   2   3   3   0   2   0   5  11
Flight 2 : Arrived
 PT HT  P00 P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20  InQ InF toB
```

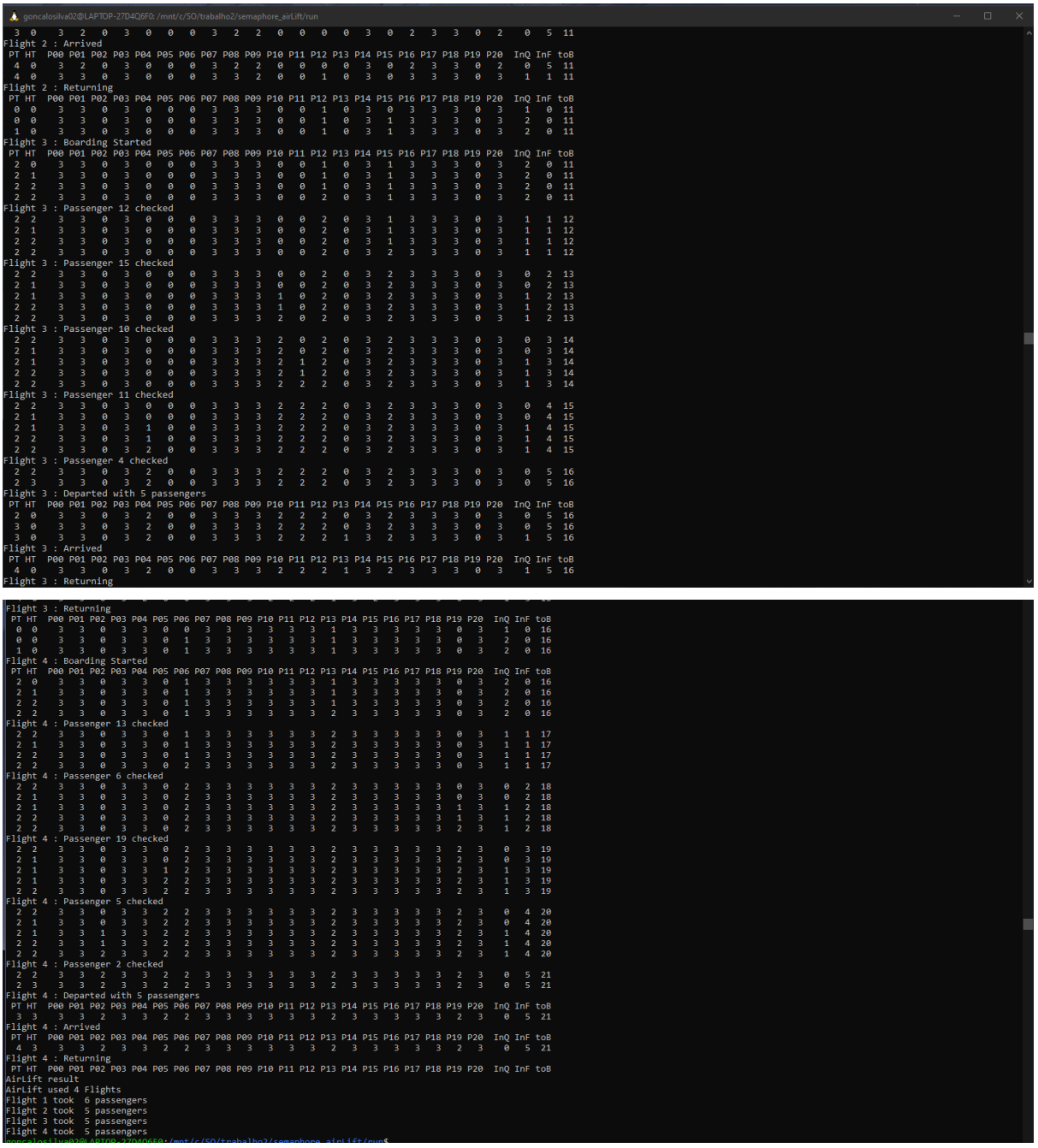


Figura 11 - Prints da Experiência 2 da Simulação de 100 000 situações

Como em nenhuma das experiências efetuadas enfrentámos deadlocks, podemos concluir que a nossa solução está bem implementada.

Por último, a manipulação do ficheiro das constantes foi fruto da curiosidade do grupo para analisarmos se a nossa solução funcionava tanto para valores superior de lotação do avião e número de passageiros, tendo também manipulado o tempo de chegada dos utilizadores ao aeroporto para que os voos tivessem quase sempre a lotação máxima. Pelo facto da solução ter funcionado em todas as combinações de dados que pensámos, concluímos, mais uma vez, que a nossa implementação é segura e adaptável a qualquer cenário.

Conclusão

A realização deste trabalho permitiu-nos consolidar a matéria lecionada nas aulas teóricas e nas aulas práticas sobre Processos, Threads, Semáforos e Memória Partilhada da cadeira de Sistemas Operativos de uma forma desafiante e interessante. Pelo facto da temática escolhida para o problema ser algo bastante familiar e conhecido permitiu-nos também aperceber que este tipo de situação/sincronização de mecanismos acontecem em tempo real em todos os aeroportos, motivando-nos e despertando o nosso interesse para as implementações práticas que a sincronização de processos/threads podem trazer para o nosso percurso académico e profissional.

No decorrer do desenvolvimento da implementação, também foi importante ter calma inicialmente e não saltar imediatamente para a produção de código em si, para conseguir entender e interpretar cada aspeto necessário para a implementação e para o desenvolvimento dos testes da implementação.