Relatório Trabalho Prático nº2

João Eduardo Santos Alcatrão
76763

Introdução

Este relatório visa elucidar e explicar a solução encontrada para o problema proposto (simulação de uma ponte aérea usando vários processos), em especial, o aspeto de execução e sincronização de processos e da sua utilização de recursos partilhados.

Neste problema, existem 3 tipos de entidades principais: 1 pilot, 1 hostess, e N passengers. Todas incarnam como processos únicos. Todos têm um ciclo de vida pré-definido. E todos têm que ser coordenados entre si (usando ferramentas como semáforos) para completarem os seus respetivos ciclos de vida corretamente e atingirem as suas finalidades, sendo este o problema em concreto a resolver.

Dados do problema explícitos no guião:

(O piloto de) um avião tem que transportar os N passageiros (que chegam ao aeroporto em tempos aleatórios dentro de um determinado intervalo) de um ponto de origem para um ponto de destino, mas não pode transportar todos os passageiros de uma só vez, pois o avião tem uma capacidade máxima (sendo necessárias várias viagens para transportar todos os passageiros). Para resolver esta incompatibilidade, existe uma hospedeira que faz o embarque dos passageiros.

O avião (piloto) parte da origem sempre que: o avião está cheio; tem um número mínimo aceitável de passageiros a bordo e não está mais ninguém à espera para embarcar; e quando já todos os N passageiros já estiverem embarcados ou entregues ao destino. Quem verifica esta condições, é a hospedeira. Ela informa o piloto quando é para partir. Por sua vez, o piloto informa a hospedeira quando o avião estiver preparado para receber passageiros, e também informa estes quando for para sair do avião. O avião voa de volta para a origem quando estiver vazio, e é este o estado inicial do problema: tudo começa com o avião a chegar.

Dados inferidos do código base:

É-nos providenciado um código base em que o programa está quase completo. Todas as constantes, regiões de memória partilhada, semáforos, estruturas e até o programa "main" estão completos. Falta apenas completar as entidades (pilot, hostess, passenger), sendo que estas também estão na sua maioria completas, mas faltam completar as funções declaradas e implementar uma correta utilização dos semáforos providenciados. Todo o código está bem documentado e com descrições precisas e úteis, incluindo as funções que faltam completar (todas têm uma breve descrição do que devem fazer em concreto).

O piloto e a hospedeira têm um ciclo de vida que se repete até que a hospedeira verifique que todos os N passageiros já foram embarcados. Os passageiros têm um ciclo de vida não repetitivo, e que acaba assim que as suas 3 funções que definem o seu comportamento sejam executadas.

Olhando para o código em primeira instância, foi possível inferir uma sequência de eventos que, embora pudesse ser pouco detalhada e ter alguns erros, permitiu ter uma visão de que era suposto as diferentes entidades, estruturas e funções fazerem e de como era suposto as entidades comportarem-se ao longo dos seus ciclos de vida. A sequência inferida foi a seguinte:

1) o piloto (avião) viaja de "origin" (flight(false)), e muda o seu estado quando chegar (de FLYING_BACK para READY_FOR_BORDING);
2) a hospedeira espera que o avião esteja pronto para o embarque (waitForNextFlight());
3) os passageiros vão chegando ao aeroporto (travelToAirport()), e ficam numa fila de espera à medida que chegam, guardando a sua posição na mesma (waitInQueue(n)), e mudam do estado GOING_TO_AIRPORT para IN_QUEUE quando chegarem;
4) o piloto, ao chegar, comunica com a hospedeira (signalReadyForBoarding()), a informá-la que o avião está pronto para receber os passageiros, e muda para o estado READY_FOR_BOARDING. De seguida, muda para o estado WAITING_FOR_BOARDING (waitUntilReadyToFlight());
5) a hospedeira muda de estado, de WAIT_FOR_FLIGHT para WAIT_FOR_PASSENGER (waitForPassenger()). Casam hajam passageiros à espera, verifica o passaporte do primeiro na fila (checkPassport()), e muda o estado para CHECK_PASSPORT, e verifica as condições de embarque e, caso não estiverem satisfeitas (se a função anterior retornar Falso), verifica outro passaporte;

O passo 5) repete-se (a segunda frase apenas) até a

hospedeira verificar a realização de alguma das condições de embarque (a função retorna Verdadeiro). Ao verificar alguma dessas condições:

6) a hospedeira muda de estado de CHECK_PASSPORT para READY_TO_FLIGHT e informa o piloto de que o embarque terminou (signalReadyToFlight());
7) o piloto muda do estado WAITING_FOR_BOARDING para FLYING e dá início ao voo (flight(true));
8) o piloto, quando chegar ao destino, muda de estado para DROPING_PASSENGERS, deixa os passageiros saírem, e só voa de volta quando todos tiverem saído (dropPassengersAtTarget());
9) os passageiros saem do avião, mudando o estado para AT_DESTINATION, sendo que o último passageiro a sair informa o piloto que é o último (waitUntilDestination(n));
10) o piloto muda de estado para FLYING_BACK (flight(false)), repetindo todo este procedimento até os N passageiros tiverem sido transportados.

Tendo isto em conta, bem como as descrições das secções "TO DO", senti reunir conhecimento sobre o problema e o programa suficientes para os traduzir em código, e comecei a escrever as funções. Faltava apenas ter noção de onde integrar os semáforos, mas os seus nomes indicativos, bem como as suas descrições, apontavam para aquiles que me pareceram ser os locais certos. Também se entendeu que o programa deve imprimir mensagens (tipicamente, estados das entidades), de forma a saber-se e perceber-se o que os vários processos fazem na prática.
Junto do código fonte, também foi providenciada uma solução pré-compilada correta deste problema, cuja execução gera várias linhas no "stdout" que indicam o que o programa está a fazer, e qual deve ser o resultado de cada operação, de início ao fim. Percebe-se que cada mudança de estado deve ser imprimida no terminal usando uma função já feita no documento "logging.c", e assim obtém-se uma evolução do programa e das suas várias entidades a cada passo. Temos o estado do piloto, da hospedeira, e dos N passageiros, todos seguidos, na horizontal, bem como 3 valores que devem ser consistentes com o progresso do programa e a evolução dos estados dos passageiros. Sendo que a cada mudança efetuada, é impressa uma nova linha no "stdout", acabamos por ter uma evolução temporal do programa na vertical, em que as últimas mudanças de estado das entidades em execução correspondem às últimas linhas impressas.
Comecei por escrever o código do piloto, pois o guião indicava que o estado do problema começa com o piloto a voar até à origem, e pareceu-me fazer sentido começar por este aparente início.

Completar o Piloto (semSharedMemPilot.c)

-Função flight(bool go) {
Esta foi a primeira função a ser atacada e resolvida. Em comentário, encontra-se um breve mas sucinta explicação do que a função deve fazer: o piloto voa até à origem, ou ao destino, e guarda o estado do piloto. Este também é o primeiro passo delineado na primeira sequência inferida e delineada acima.
O parâmetro "go" corresponde um valor booleano que indica se o piloto voa para o destino (True) ou voa para a origem (False). No final, a função foi completa da seguinte maneira:

1) Se "go" for falso, mudar estado do piloto para 0 (FLYING_BACK). Caso contrário, mudar para 3 (FLYING). Guardar estado (o que resulta num "log"; que imprime as alterações no "stdout").

Esta é a única operação efetuada nesta função. A função já feita acima desta no código (isFinished()) mostra como aceder ao estado de uma entidade. No ficheiro "probConst.h" podem-se encontrar todos os possíveis estados de cada entidade, bem como outras constantes, como o número total de passageiros (e consequentemente, o número total de processos "Passenger" que serão executados). E no ficheiro "probDataStruct.h" pode-se encontrar uma estrutura das entidades intervenientes, onde se podem encontrar os estados atuais e concretos destas mesmas.
A verificação do valor do parâmetro "go" é feita com recurso a uma condição "if", e o "log" é feito com através da função/comando saveState(nFic, &sh→fSt).
Nota: todas as mudanças de estado e execuções de saveStates são feitas dentro das regiões críticas denotadas no código base.
}

-Função signalReadyForBoarding() {
Nesta função, tendo em conta a descrição do que deve ser feito, e a sequência de eventos delineada que ajuda a contextualizar a relação desta entidade com outras (em específico, a hospedeira), foram materializadas as seguintes operações:

1) O piloto atualiza o seu estado para 1 (READY_FOR_BOARDING). O número do voo é incrementado por 1. O estado é guardado, e é apresentada uma mensagem de início o embarque;

2) O piloto sinaliza a hospedeira que esta pode começar operações de embarque.
Esta última operação é o (meu) primeiro contacto com a necessidade de sincronizar processos (neste problema). Previamente já tinha observado e lido o ficheiro "sharedDataSync.h", onde se encontram definidos todos os semáforos que se podem utilizar, tendo assim acesso ao estado do problema. Visto que o piloto deve sinalizar a hospedeira para ela prosseguir com operações, é intuitivo o pensamento de que o semáforo dedicado a esta conjuntura (readyForBoarding) deva ser desbloqueado.
}

-Função waitUntilReadyToFlight() {
As operações efetuadas para completar esta função foram:

1) O piloto muda de estado para 2 (WAITING_FOR_BOARDING);

2) O piloto tem de esperar que o embarque termine.

Relativamente a esta última operação, o processo correspondente ao piloto não deve prosseguir com a sua execução até que realmente o embarque termine. Visto que na introdução/sequência delineada, a hospedeira trata do embarque, e que existe um semáforo que indica e é descrito de forma a sinalizar o seu uso nesta situação (readyToFlight), é feito um Down() a este semáforo. O piloto só prossegue quando este semáforo for desbloqueado (e tudo indica que é a hospedeira a fazê-lo, e já veremos que nesta solução, ela faz-o mesmo).
}

-Função dropPassengersAtTarget() {
Esta é a última função apresentada no código do piloto, e também a última a ser executada por parte deste idealmente, como se pode observar no seu ciclo de vida. Esta também foi, pessoalmente, a função mais frustrante e que mais tempo consumiu, e em que tive mais dificuldades em implementar corretamente, visto que as minhas soluções/tentativas iniciais resultariam eventualmente quase sempre em deadlocks, e eu não conseguia perceber o porquê, até chegar à solução atual através do uso de imensos "printfs" e conseguir perceber um detalhe bastante importante sobre semáforos. As operações implementadas para completar esta função foram:

1) É apresentada a informação de que o voo chegou. O piloto muda para o estado 4 (DROPING_PASSENGERS), que é guardado;

2) O piloto sinaliza os passageiros que podem sair do avião;

Esta é a operação que causou alguma frustração e consumiu imenso, e que no final, a operação estaria correta se estivesse dentro da região crítica do mutex acima. Aqui percebi um detalhe de como um semáforo se diferencia de um mutex. Percebi que a solução pré-compilada fazia um Down() ao semáforo em questão por cada passageiro (na altura, ainda não tinha implementado funções do passageiro), algo que me demorou bastante tempo até fazer "luz", e mais tarde percebi que fazer um número igual (eventualmente até utilizei um ciclo "while" no desespero) de Ups() utilizando o valor do estado do número de passageiros atualmente embarcados (que correspondia ao número de Downs() feitos) não podia ser feito com o próprio estado, visto que este faz parte da memória partilhada, e que vários processos o alteram, e que num ciclo "for", como está aqui implementado, isto causa um fenómeno conhecido como condição de corrida.

Como cada passageiro faz um Down(), também temos que fazer um Up() por passageiro. Para evitar deadlocks e condições de corrida, esta secção inteira deve estar dentro da região crítica, para que o valor de estado "nPassInFlight" não seja inconsistente entre interações. Alternativamente, simplesmente fixa-se o número inicial de passageiros em voo antes de se fazerem as iterações, que é a solução atual nesta operação (que vejo como seguro visto que o Up() é atómico).

```
int passageiros=sh->fSt.nPassInFlight;

for (int i=0; i<passageiros; i++) {
    if (semUp (semgid, sh->passengersWaitInFlight) == -1)  {
        perror ("erro ao desbloquear semáforo que sinaliza passageiros que podem abandonar o avião");
        exit (EXIT_FAILURE);
    }
    //printf("Up %d\n", i);
```

Fig.1: Incremento do valor do semáforo "passengersWaitInFlight", que é usado para os passageiros não prosseguirem enquanto estão a voar, e que é usado pelo piloto para os informar que o voo acabou e que podem sair.

3) O piloto só parte quando o semáforo "planeEmpty" estiver desbloqueado;

Se o processo do piloto chegar aqui primeiro do que o processo do último passageiro, o piloto bloqueia à espera do último passageiro, e depois deste desbloquear o semáforo, o piloto prossegue. Se a thread do último passageiro chegar primeiro, o piloto quando chegar não espera, mas também já não tem que esperar.

4) O piloto muda para o estado 0 (FLYING_BACK). É apresentada a informação de que o avião está a regressar, e o estado não é guardado.

O estado não é guardado, conforme é solicitado no comentário

que explica o que a função deve fazer. No entanto, é chamada a função saveFlightReturning(nFic, &sh→fSt), de forma a que se possa obter resultados mesmo idênticos entre a solução pré-compilada e esta.
}
E com a compleição desta função, o piloto ficou operacional e funcional com a solução pré-compilada ("make pt"). Após olhar e ruminar sobre o código da hostess e do passenger, o passenger pareceu-me mais simples e direto de atacar (só era necessários completar duas funções, embora uma delas fosse algo extensa e complexa), e o código do passenger foi o próximo a ser completado.
Nota: todas as entidades nesta solução funcionam bem umas com as outras, bem como com as suas versões pré-compiladas.

Completar o Passageiro (semSharedMemPassenger.c)

-Função waitInQueue(unsigned int passengerId) {
Esta função é a segunda no ciclo de vida do passenger, mas é a primeira que precisa de ser completada, e também é a mais complexa. Um dos objetivos que esta função tem que atingir é o de verificar o passaporte e embarcar, algo que requer bastante interação com a hostess, existindo 3 semáforos diferentes que têm de ser sincronizados de modo a obter uma interação correta entre a hostess e o passenger (tendo em atenção que existem N processos passenger, e que todos vão ter de interagir com a hostess e todos vão alterar dados da zona de memória partilhada).
Houve alguma confusão inicialmente acerca dos vários semáforos a usar e os seus papéis (por exemplo, não me foi imediata e diferença entre os semáforos "passengersInQueue" e "passengersWaitInQueue"), mas após ler com mais atenção as descrições dos semáforos, reparei que a primeira entidade a ser referida no comentário de cada semáforo, é referida como a entidade que espera, e que bloqueia (faz Down()) o semáforo.

Nota: por observação dos semáforos em "sharedDataSync.h" e das suas descrições, e confirmado depois na utilização dos mesmos, os semáforos revelam um padrão de descrição do género "identification of semaphore used by entity1 to wait for…", que indica que essa entidade ("entity1") bloqueia o semáforo. Os predicados refletem as entidades que vão desbloquear esse semáforo.

1) O passageiro muda para o estado IN_QUEUE, e incrementa o nPassInQueue. O estado é guardado;
O estado dos passageiros é representado num array de unsigned ints ao invés de um unsigned int isolado, sendo necessário usar o valor do passengerId aceder ao index correspondente no array para mudar o estado do passageiro em questão.

2) O passageiro informa a hospedeira que há passageiros na fila;
Através da versão pré-compilada, pode-se observar que a hospedeira só muda para o estado "CHECK_PASSPORT" se houverem passageiros na fila. Portanto, o processo dela bloqueia se não houverem passageiro à espera de embarcarem (situação prevista na descrição do semáforo "passengersInQueue"), e como tal, o passageiro deve desbloquear o processo dela ao entrar para a fila de espera. Cada passageiro faz um Up() ao semáforo "passengersInQueue". Por sua vez, existe um ciclo "do while" no ciclo de vida da hospedeira que fará esta fazer um Down() por cada passageiro, até não haverem mais passageiros para embarcar. Ela bloqueia quando não houverem mais passageiros para embarcar, mas ainda não tiverem sido todos transportados.

3) O passageiro espera que a hostess o aborde;
Ao chegar à fila de espera, o passageiro tem de lá ficar até que a hospedeira lhe verificar o passaporte. Portanto, o processo do passageiro tem de passar para o estado "Waiting" aqui e só deve voltar a executar o restante código quando o processo da hospedeira o desbloquear. O semáforo usado para conseguir isto é "passengersWaitInQueue", ao qual cada passageiro faz um Down() neste passo.

4) O passageiro dá permissão à hostess para lhe verificar o ID. O Id é guardado e impresso. O passageiro muda do estado IN_QUEUE para o estado IN_FLIGHT e é feito um saveState;
O passageiro faz Up() no semáforo "idShown", que é usado pela hospedeira para se bloquear enquanto espera que o passaporte do passageiro seja verificado. Isto é importante, pois o Id do último passageiro a ser verificado é atualizado aqui, e que é necessário para imprimir um log de "PassengerChecked", mas sobretudo para o passageiro mudar de estado, e oo números de passageiros na fila e em voo e total de embarcados possam ser atualizados e usados pela hospedeira para verificar as condições de embarque. Caso se verifiquem, a hospedeira não desbloqueia mais nenhum passenger que esteja bloqueado no passo acima (bloqueado com o Down() de "passengersWaitInQueue") pois haverá voo e ela sairá do ciclo de espera pelos passageiros e de verificação de passaportes.}

-Função waitUntilDestination (unsigned int passengerId) {
Esta função completa o ciclo de vida do passageiro; o processo acaba com o término desta função. Esta função serve para os passageiros bloquearem até o voo terminar, e quando o voo terminar, os passageiros desbloqueiam, mudam de estado, e terminam o processo. Antes de o fazerem, verificam se são o último passageiro ou não, de forma a avisar o piloto que pode regressar.

1) O passageiro espera que o voo termine;
Cada passageiro faz Down() ao semáforo "passengersWaitInFlight", bloqueando aqui, até que o piloto faça Up();

2) O passageiro muda de estado para AT_DESTINATION, e decrementa o número de passageiros em voo. O estado é guardado;
Como já referido numa nota anterior, sempre que ocorrem mudanças de estado, ou alteração do número de passageiros em voo ou de outros valores de memória partilhada, estas operações são feitas dentro da região crítica.

3) O passageiro verifica se é o último, e informa piloto caso seja.
Cada passageiro verifica o número de passageiros em voo. Se este valor for zero, significa que o processo a fazer esta verificação é o último a sair, e portanto faz um Up() ao semáforo "planeEmpty" (tudo isto também é feito dentro da região crítica). O processo do passageiro termina aqui.
}

Completar a hospedeira (semSharedMemHostess.c)

A hospedeira tem, discutivelmente, o ciclo de vida mais complexo de todas as entidades, visto que possui um ciclo "do while" dentro de um ciclo "while". Ela existe enquanto existirem processos passageiros, e executa um número de tarefas e funções repetidamente até verificar que existem condições de voo.

-Função waitForNextFlight() {

Esta é a primeira função da hospedeira. Ela atualiza o seu estado e bloqueia enquanto (o piloto) voa, sendo desbloqueada quando o piloto aterrar na origem e declarar que avião está pronto para o embarque.

1) A hostess muda para o estado WAIT_FOR_FLIGHT. O estado é guardado;

2) A hostess espera até que o piloto sinalize que o avião está pronto para boarding.
A hospedeira faz Down() no semáforo "readyForBoarding", bloqueando até que o piloto a desbloqueie.
}

-Função waitForPassenger() {

Nesta função, a hospedeira simplesmente bloqueia até haverem passageiros na fila.

1) A hostess muda o seu estado para WAIT_FOR_PASSENGER;

2) A hostess espera pela chegada dos passageiros;
A hospedeira faz Down() ao semáforo "passengersInQueue" (ao qual os passageiros fazem Up() ao chegarem ao aeroporto).
}

-Função bool checkPassport() {
Nesta função, a hospedeira tem como objetivo principal embarcar os passageiros, e verificar, à medida que os vai embarcando (como é demonstrado pelo ciclo "do while" que termina quando esta função retornar True), as condições de embarque.

1) A hostess atende um passageiro na fila de espera;
A hospedeira faz Up() ao semáforo "passengersWaitInQueue", desbloqueando um processo passenger que aí estivesse em estado "waiting".

2) A hostess muda para o estado CHECK_PASSPORT. O estado é guardado (saveState);

3) A hostess espera que o passageiro lhe dê o passaporte para ela verificar;

A hospedeira faz Down() no semáforo “idShown”, bloqueando assim o seu processo até que o passageiro acima desbloqueado lhe retribua a ação e a desbloqueie (e atualize o seu estado e atualize o Id do último passageiro verificado e faça o respetivo saveState).
Nota: todas as entidades mudam o seu próprio estado, e não o estado de outras entidades. Cada passageiro também muda apenas o seu próprio estado.

4) A hospedeira atualiza o número de passageiros na fila de espera, no avião, e total de embarcados, e faz o devido registo (savePassengerChecked)

5) A hospedeira faz a verificação das condições de voo, faz um saveSate que atualiza no “stdout” as alterações anteriores e, caso as condições se verifiquem, regista o número de passageiros neste voo, e atualiza o valor que esta função vai retornar.
Esta operação e a operação acima (4) são feitas dentro da mesma região crítica, não dando possibilidade de haver inconsistências entre os dados impressos, ou pior.
}

-Função signalReadyToFlight() {

Esta é a última função da hospedeira. Nesta função, ela informa o piloto que o embarque terminou, e verifica se este é o último voo para o destino ou não. Caso seja, e todos os N passageiros já tiverem sido embarcados, ela já não volta a repetir o seu ciclo de vida e termina o seu processo, o que explica o seu estado ser diferente no voo final do que nos restantes voos (em que se pode observar no “stdout” que está no estado 0, e no final está no estado 3).

1) A hostess muda para o estado READY_TO_FLIGHT, regista o número de passageiros no voo, e o estado é guardado;
Para que o output entre este programa e o pré-compilado sejam mais idênticos, o log do “FlightDeparted” é feito aqui.

2) A hostess verifica se todos os N passageiros já foram transportados;
O ciclo de vida do piloto não acaba sem este passo.

3) A hostess informa o piloto que embarque terminou.
Se este não for o último voo, a hospedeira repete o seu ciclo de vida. Caso contrário, o seu processo termina aqui.
}

Resultados

```
joao@joao-HP-Laptop-17-cp0xxx:~/Desktop/Trabalho Semáforos Voo/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   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0    0   0   0
  0  0    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1    1   0   0
  1  0    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1    1   0   0
Flight 1 : Boarding Started
 PT HT  P00 P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20  InQ InF toB
  2  0    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1    1   0   0
  2  1    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1    1   0   0
  2  2    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1    1   0   0
  2  2    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2    1   0   0
Flight 1 : Passenger 20 checked
  2  2    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2    0   1   1
  2  1    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2    0   1   1
  2  1    1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2    1   1   1
  2  2    1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2    1   1   1
  2  2    2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2    1   1   1
Flight 1 : Passenger 0 checked
  2  2    2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2    0   2   2
  2  1    2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2    0   2   2
  2  1    2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   2    1   2   2
  2  2    2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   2    1   2   2
  2  2    2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   2    1   2   2
Flight 1 : Passenger 18 checked
  2  2    2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   2    0   3   3
  2  1    2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   2    0   3   3
  2  1    2   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   2   0   2    1   3   3
  2  2    2   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   2   0   2    1   3   3
  2  2    2   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   2   0   2    1   3   3
Flight 1 : Passenger 4 checked
  2  2    2   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   2   0   2    0   4   4
  2  1    2   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   2   0   2    0   4   4
  2  1    2   0   0   0   2   0   0   0   0   0   1   0   0   0   0   0   0   2   0   2    1   4   4
  2  2    2   0   0   0   2   0   0   0   0   0   1   0   0   0   0   0   0   2   0   2    1   4   4
  2  2    2   0   0   0   2   0   0   0   0   0   2   0   0   0   0   0   0   2   0   2    1   4   4
Flight 1 : Passenger 10 checked
  2  2    2   0   0   0   2   0   0   0   0   0   2   0   0   0   0   0   0   2   0   2    0   5   5
  2  3    2   0   0   0   2   0   0   0   0   0   2   0   0   0   0   0   0   2   0   2    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    2   0   0   0   2   0   0   0   0   0   2   0   0   0   0   0   0   2   0   2    0   5   5
  3  0    2   0   0   0   2   0   0   0   0   0   2   0   0   0   0   0   0   2   0   2    0   5   5
  3  0    2   0   0   0   2   0   0   0   0   0   2   0   0   0   1   0   0   2   0   2    1   5   5
```

Fig.2: Excerto inicial da execução do programa (compilação make all).

```
  2  2    3   2   3   3   3   2   3   1   2   3   3   1   2   3   3   1   2   3   3   1   3   4   5  17
  2  1    3   2   3   3   3   2   3   1   2   3   3   1   2   3   3   1   2   3   3   1   3   4   5  17
  2  2    3   2   3   3   3   2   3   1   2   3   3   1   2   3   3   1   2   3   3   1   3   4   5  17
  2  2    3   2   3   3   3   2   3   1   2   3   3   1   2   3   3   1   2   3   3   2   3   4   5  17
Flight 3 : Passenger 19 checked
  2  2    3   2   3   3   3   2   3   1   2   3   3   1   2   3   3   1   2   3   3   2   3   3   6  18
  2  1    3   2   3   3   3   2   3   1   2   3   3   1   2   3   3   1   2   3   3   2   3   3   6  18
  2  2    3   2   3   3   3   2   3   1   2   3   3   1   2   3   3   1   2   3   3   2   3   3   6  18
  2  2    3   2   3   3   3   2   3   1   2   3   3   2   2   3   3   1   2   3   3   2   3   3   6  18
Flight 3 : Passenger 11 checked
  2  2    3   2   3   3   3   2   3   1   2   3   3   2   2   3   3   1   2   3   3   2   3   2   7  19
  2  1    3   2   3   3   3   2   3   1   2   3   3   2   2   3   3   1   2   3   3   2   3   2   7  19
  2  2    3   2   3   3   3   2   3   1   2   3   3   2   2   3   3   1   2   3   3   2   3   2   7  19
  2  2    3   2   3   3   3   2   3   1   2   3   3   2   2   3   3   2   2   3   3   2   3   2   7  19
Flight 3 : Passenger 15 checked
  2  2    3   2   3   3   3   2   3   1   2   3   3   2   2   3   3   2   2   3   3   2   3   1   8  20
  2  1    3   2   3   3   3   2   3   1   2   3   3   2   2   3   3   2   2   3   3   2   3   1   8  20
  2  2    3   2   3   3   3   2   3   1   2   3   3   2   2   3   3   2   2   3   3   2   3   1   8  20
  2  2    3   2   3   3   3   2   3   2   2   3   3   2   2   3   3   2   2   3   3   2   3   1   8  20
Flight 3 : Passenger 7 checked
  2  2    3   2   3   3   3   2   3   2   2   3   3   2   2   3   3   2   2   3   3   2   3   0   9  21
  2  3    3   2   3   3   3   2   3   2   2   3   3   2   2   3   3   2   2   3   3   2   3   0   9  21
Flight 3 : Departed with 9 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   2   3   3   3   2   3   2   2   3   3   2   2   3   3   2   2   3   3   2   3   0   9  21
Flight 3 : 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  3    3   2   3   3   3   2   3   2   2   3   3   2   2   3   3   2   2   3   3   2   3   0   9  21
  4  3    3   2   3   3   3   2   3   2   2   3   3   2   3   3   3   2   2   3   3   2   3   0   8  21
  4  3    3   2   3   3   3   3   3   2   2   3   3   2   3   3   3   2   2   3   3   2   3   0   7  21
  4  3    3   3   3   3   3   3   3   2   2   3   3   2   3   3   3   2   2   3   3   2   3   0   6  21
  4  3    3   3   3   3   3   3   3   2   3   3   3   2   3   3   3   2   2   3   3   2   3   0   5  21
  4  3    3   3   3   3   3   3   3   2   3   3   3   2   3   3   3   2   3   3   3   2   3   0   4  21
  4  3    3   3   3   3   3   3   3   2   3   3   3   3   3   3   3   2   3   3   3   2   3   0   3  21
  4  3    3   3   3   3   3   3   3   2   3   3   3   3   3   3   3   3   3   3   3   2   3   0   2  21
  4  3    3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   2   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 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
AirLift result
AirLift used 3 Flights
Flight 1 took  5 passengers
Flight 2 took  7 passengers
Flight 3 took  9 passengers
```

Fig.3: Excerto final da execução do programa (compilação make all).

```
joao@joao-HP-Laptop-17-cp0xxx:~/Desktop/Sistemas Operativos/Trabalho 2/semaphore_airLift/run$ ./filter.sh
                        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
  .  .    .   .   1   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   0   0
  1  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   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  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   0   0
  .  1    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   0   0
  .  2    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   0   0
  .  .    .   .   2   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   0   0
Flight 1 : Passenger 2 checked
  .  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   1   1
  .  1    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   1   1
  .  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   1   .    1   1   1
  .  2    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   1   1
  .  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   2    1   1   1
Flight 1 : Passenger 20 checked
  .  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   2   2
  .  1    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   2   2
  .  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   1   .    1   2   2
  .  2    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   2   2
  .  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   2   .    1   2   2
Flight 1 : Passenger 19 checked
  .  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   3   3
  .  1    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   3   3
  .  .    .   .   .   .   .   1   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   3   3
  .  2    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   3   3
  .  .    .   .   .   .   .   2   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   3   3
Flight 1 : Passenger 4 checked
  .  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   4   4
  .  1    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   4   4
  .  .    .   .   .   .   .   .   .   .   .   .   .   1   .   .   .   .   .   .   .   .   .    1   4   4
  .  2    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   4   4
  .  .    .   .   .   .   .   .   .   .   .   .   .   2   .   .   .   .   .   .   .   .   .    1   4   4
Flight 1 : Passenger 11 checked
  .  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   5   5
  .  3    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    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
  .  0    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   5   5
  3  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   5   5
  .  .    .   .   .   .   .   .   .   .   1   .   .   .   .   .   .   .   .   .   .   .   .    1   5   5
```

Fig.4: Excerto inicial da execução do script filter.sh (compilação make all).

```
Flight 4 : Passenger 10 checked
  .  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   5  20
  .  3    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   5  20
Flight 4 : 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
  .  0    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   5  20
  3  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   5  20
  .  .    .   .   .   .   1   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   5  20
Flight 4 : 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  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   5  20
  .  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   3   .   .   .   .   .   .    1   4  20
  .  .    3   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   3  20
  .  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   3   .   .    1   2  20
  .  .    .   .   .   .   .   .   .   .   .   .   3   .   .   .   .   .   .   .   .   .   .    1   1  20
  .  .    .   .   .   .   .   .   .   .   .   .   .   .   .   3   .   .   .   .   .   .   .    1   0  20
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
  0  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   0  20
  1  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   0  20
Flight 5 : 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  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   0  20
  .  1    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   0  20
  .  2    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   0  20
  .  .    .   .   .   .   2   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    1   0  20
Flight 5 : Passenger 3 checked
  .  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   1  21
  .  3    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   1  21
Flight 5 : Departed with 1 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  .    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   1  21
Flight 5 : 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   1  21
  .  .    .   .   .   .   3   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .    0   0  21
Flight 5 : 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 5 Flights
Flight 1 took  5 passengers
Flight 2 took  5 passengers
Flight 3 took  5 passengers
Flight 4 took  5 passengers
Flight 5 took  1 passengers
```

Fig.5: Excerto final da execução do script filter.sh (compilação make all).

```
  2  2    3   2   3   3   3   3   3   3   2   3   3   2   2   3   3   3   3   3   1   3   3    1   4  20
  2  2    3   2   3   3   3   3   3   3   2   3   3   2   2   3   3   3   3   3   2   3   3    1   4  20
Flight 4 : Passenger 18 checked
  2  2    3   2   3   3   3   3   3   3   2   3   3   2   2   3   3   3   3   3   2   3   3    0   5  21
  2  3    3   2   3   3   3   3   3   3   2   3   3   2   2   3   3   3   3   3   2   3   3    0   5  21
Flight 4 : 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
  3  3    3   2   3   3   3   3   3   3   2   3   3   2   2   3   3   3   3   3   2   3   3    0   5  21
Flight 4 : 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  3    3   2   3   3   3   3   3   3   2   3   3   2   2   3   3   3   3   3   2   3   3    0   5  21
  4  3    3   3   3   3   3   3   3   3   2   3   3   2   2   3   3   3   3   3   2   3   3    0   4  21
  4  3    3   3   3   3   3   3   3   3   2   3   3   2   3   3   3   3   3   3   2   3   3    0   3  21
  4  3    3   3   3   3   3   3   3   3   2   3   3   3   3   3   3   3   3   3   2   3   3    0   2  21
  4  3    3   3   3   3   3   3   3   3   2   3   3   3   3   3   3   3   3   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  5 passengers
Flight 3 took  6 passengers
Flight 4 took  5 passengers

Run n.º 1000
                              Air Lift - Description of the internal state

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

Fig.6: Excerto da execução do script run.sh (compilação make all).

```
 PT HT   P00 P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20 P21 P22 P23 P24 P25 P26 P27 P28  InQ InF toB
  0  0    3   3   3   3   0   3   3   3   3   3   3   3   3   3   3   3   3   3   0   3   3   3   3   3   3   0   3   3   3    0   0  26
  0  0    3   3   3   3   0   3   3   3   3   3   3   3   3   3   3   3   3   3   0   3   3   3   3   3   3   1   3   3   3    1   0  26
  1  0    3   3   3   3   0   3   3   3   3   3   3   3   3   3   3   3   3   3   0   3   3   3   3   3   3   1   3   3   3    1   0  26
Flight 6 : 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 P21 P22 P23 P24 P25 P26 P27 P28  InQ InF toB
  2  0    3   3   3   3   0   3   3   3   3   3   3   3   3   3   3   3   3   3   0   3   3   3   3   3   3   1   3   3   3    1   0  26
  2  0    3   3   3   3   0   3   3   3   3   3   3   3   3   3   3   3   3   3   1   3   3   3   3   3   3   1   3   3   3    2   0  26
  2  0    3   3   3   3   1   3   3   3   3   3   3   3   3   3   3   3   3   3   1   3   3   3   3   3   3   1   3   3   3    3   0  26
  2  1    3   3   3   3   1   3   3   3   3   3   3   3   3   3   3   3   3   3   1   3   3   3   3   3   3   1   3   3   3    3   0  26
  2  2    3   3   3   3   1   3   3   3   3   3   3   3   3   3   3   3   3   3   1   3   3   3   3   3   3   1   3   3   3    3   0  26
  2  2    3   3   3   3   1   3   3   3   3   3   3   3   3   3   3   3   3   3   1   3   3   3   3   3   3   2   3   3   3    3   0  26
Flight 6 : Passenger 25 checked
  2  2    3   3   3   3   1   3   3   3   3   3   3   3   3   3   3   3   3   3   1   3   3   3   3   3   3   2   3   3   3    2   1  27
  2  1    3   3   3   3   1   3   3   3   3   3   3   3   3   3   3   3   3   3   1   3   3   3   3   3   3   2   3   3   3    2   1  27
  2  2    3   3   3   3   1   3   3   3   3   3   3   3   3   3   3   3   3   3   1   3   3   3   3   3   3   2   3   3   3    2   1  27
  2  2    3   3   3   3   1   3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   3   3   3   2   3   3   3    2   1  27
Flight 6 : Passenger 18 checked
  2  2    3   3   3   3   1   3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   3   3   3   2   3   3   3    1   2  28
  2  1    3   3   3   3   1   3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   3   3   3   2   3   3   3    1   2  28
  2  2    3   3   3   3   1   3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   3   3   3   2   3   3   3    1   2  28
  2  2    3   3   3   3   2   3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   3   3   3   2   3   3   3    1   2  28
Flight 6 : Passenger 4 checked
  2  2    3   3   3   3   2   3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   3   3   3   2   3   3   3    0   3  29
  2  3    3   3   3   3   2   3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   3   3   3   2   3   3   3    0   3  29
Flight 6 : 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 P21 P22 P23 P24 P25 P26 P27 P28  InQ InF toB
  3  3    3   3   3   3   2   3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   3   3   3   2   3   3   3    0   3  29
Flight 6 : Arrived
 PT HT   P00 P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20 P21 P22 P23 P24 P25 P26 P27 P28  InQ InF toB
  4  3    3   3   3   3   2   3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   3   3   3   2   3   3   3    0   3  29
  4  3    3   3   3   3   2   3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   3   3   3   3   3   3   3    0   2  29
  4  3    3   3   3   3   2   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3    0   1  29
  4  3    3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3    0   0  29
Flight 6 : Returning
 PT HT   P00 P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20 P21 P22 P23 P24 P25 P26 P27 P28  InQ InF toB
AirLift result
AirLift used 6 Flights
Flight 1 took  5 passengers
Flight 2 took  6 passengers
Flight 3 took  5 passengers
Flight 4 took  5 passengers
Flight 5 took  5 passengers
Flight 6 took  3 passengers
```

Fig.7: Excerto final da execução do programa com N=29 (compilação make all).

Conclusão

Penso que a solução a que cheguei é uma solução apropriada para o problema proposto, pois esta solução gera resultados aparentemente corretos, consistentes, semelhantes ao que o programa pré-compilado produz (tentei adaptar o código que escrevi de modo a replicar de maneira muita próxima os resultados emitidos pelo programa pré-compilado).

O objetivo principal da sincronização dos processos parece-me ter sido conseguido, tendo em especial atenção 2 metas implícitas mas não faladas no guião: as de impossibilitar a ocorrência de deadlocks e de condições de corrida no programa. Após a execução do script run.sh providenciado várias vezes, e não ter encontrado um único deadlock, deduzo que este fenómeno foi impossibilitado com sucesso. É de mencionar que tentei escrever dentro das regiões críticas limitadas pelos mutexes (e pelos comentários que as assinalavam) apenas o estritamente necessário (mudanças de estado e alterações dos valores de estado). Também foi posto de fora da região crítica um ciclo "for" em que é feito Up() ao semáforo "passengersWaitInFlight", através da criação de uma nova variável int destinada à fixação do número de passageiros em voo antes de ser executado o ciclo.

As mensagens imprimidas no "stdout" correspondentes à execução de funções "saveState" apresentam consistência, tanto quanto à validade e razoabilidade das mudanças de estado (não observei nenhuma ocorrência de alguma entidade mudar para um estado que não deve, ou "saltar" estados, com a exceção prevista de mudar do estado final para o inicial; nenhuma entidade muda do estado 0 para o estado 2, por exemplo), tanto quanto ao número de mudanças apresentadas em cada mensagem (escrevi o código de modo a que cada "saveState" apresentasse uma e uma só alteração de estado para o estado anterior, com a exceção prevista do número de passageiros em fila, InQ, e o número de passageiros em voo, InF, alterarem em simultâneo, como apresentado no programa pré-compilado). Considerando isto, e o facto de todas as execuções do programa terminarem com os estados e valores previstos, também cismo que a ocorrência de fenómenos de condições de corrida foi impossibilitada.

Tive cuidado em tentar comentar todas as alterações que fiz ao código base (o objetivo das alterações está escrito na linha acima delas), e de não ter entidades a mudar o estado de mais nenhuma a não ser elas próprias. Todas as entidades nesta solução funcionam bem independentemente de terem que se sincronizar com outras entidades que completei ou com as suas versões pré-compiladas (usei os comandos "make pt", "make pg" e "make ht" antes de usar "make all").

Este trabalho também me ajudou imenso a diferenciar mutexes de semáforos, como explicado na última função do piloto, em que aprendi que podia fazer vários Ups() ao mesmo semáforo consecutivamente com propósito, visto que eles alteram o valor do semáforo (reconheço que é algo que já devia de saber, mas aprendi aqui). A descrição dos semáforos também me ajudou imenso a saber onde aplicá-los, e a antever quando deveriam ser (des)bloqueados. Por observação dos semáforos, e das suas descrições, e confirmado na utilização dos mesmos, os semáforos que são descritos como "used by entity1…" são sempre bloqueados por essa sentidade. Os predicados refletem as entidades que vão desbloquear esses semáforos. Sinto-me menos espantado pelo conceito de semáforos e mais em paz com o seu uso, que me enleava antes de resolver este problema.