universidade de aveiro

Sistemas Operativos
Prof. Nuno Lau

# SIMULAÇÃO DE PONTE AÉREA

Trabalho Prático 2

**Catarina Costa - 103696**
**Guilherme Antunes - 103600**

**2021/2022**

# Índice

# Introdução

Na sequência do trabalho proposto na unidade curricular de Sistemas Operativos, este relatório tem como objetivo a explicação dos raciocínios utilizados para formular o código necessário para a correta execução e sincronização dos processos e threads fornecidas no código template. O tema do trabalho consiste numa ponte aérea que é servida por apenas 1 avião. Na ponte aérea coexistem três entidades que correspondem a processos independentes:

*Passenger* **-** chegam ao aeroporto em tempos aleatórios e são clientes do avião.

*Pilot* **-** informa a Hostess sempre que o avião está pronto para iniciar o processo de embarque, é informado pela mesma sempre que o embarque está completo e dá permissão aos *Passenger* para saírem do avião no destino. O *Pilot* só inicia o voo de regresso quando o último *Passenger* sair do avião.

*Hostess* **-** trata da verificação da identidade dos passageiros, dando autorização a cada passageiro para sair da fila de espera e entrar no avião.

Todo o código que será analisado neste relatório foi exclusivamente aquele que foi dado para completar, sendo que para que todo o programa funcione são necessários todos os outros ficheiros que não foram aqui referenciados.

# Tabela Modelo do Comportamento dos Semáforos

| **Semáforo** | **UP** | | | **DOWN** | | |
|---|---|---|---|---|---|---|
| | Quem? | Quando? | Quantos? | Quem? | Quando? | Quantos? |
| *Mutex* | *Passenger* | waitInQueue | 2 | *Passenger* | waitInQueue | 2 |
| | | waitUntilDestination | 1 | | waitUntilDestination | 1 |
| | *Pilot* | flight | 1 | *Pilot* | flight | 1 |
| | | signalReadyForBoarding | 1 | | signalReadyForBoarding | 1 |
| | | waitUntilReadyToFlight | 1 | | waitUntilReadyToFlight | 1 |
| | | dropPassengersAtTarget | 2 | | dropPassengersAtTarget | 2 |
| | *Hostess* | waitForNextFlight | 1 | *Hostess* | waitForNextFlight | 1 |
| | | waitForPassenger | 1 | | waitForPassenger | 1 |
| | | checkPassport | 2 | | checkPassport | 2 |
| | | signalReadyToFlight | 1 | | signalReadyToFlight | 1 |
| *passengersInQueue* | *Passenger* | waitInQueue | 1 | *Hostess* | waitForPassenger | 1 |
| *passengersWaitInQueue* | *Hostess* | checkPassport | 1 | *Passenger* | waitInQueue | 1 |
| *passengersWaitInFlight* | *Pilot* | dropPassengersAtTarget | nPassInFlight | *Passenger* | waitUntilDestination | 1 |
| *readyForBoarding* | *Pilot* | signalReadyForBoarding | 1 | *Hostess* | waitForNextFlight | 1 |
| *readyToFlight* | *Hostess* | signalReadyToFlight | 1 | *Pilot* | waitUntilReadyToFlight | 1 |
| *idShown* | *Passenger* | waitInQueue | 1 | *Hostess* | checkPassport | 1 |
| *planeEmpty* | *Passenger* | waitUntilDestination | 1 | *Pilot* | dropPassengersAtTarget | 1 |

# Hostess

## Função waitForNextFlight()

```
142  static void waitForNextFlight ()
143  {
144      if (semDown (semgid, sh->mutex) == -1){
145          perror ("error on the down operation for semaphore access (HT)");
146          exit (EXIT_FAILURE);
147      }
148
149      /* insert your code here */
150
151      sh->fSt.st.hostessStat = WAIT_FOR_FLIGHT;
152      saveState(nFic,&sh->fSt);
153
154      if (semUp (semgid, sh->mutex) == -1){
155          perror ("error on the up operation for semaphore access (HT)");
156          exit (EXIT_FAILURE);
157      }
158
159      /* insert your code here */
160
161      if (semDown (semgid, sh->readyForBoarding) == -1){
162          perror ("error on the down operation for semaphore access (HT)");
163          exit (EXIT_FAILURE);
164      }
165  }
```

Para a construção da função *waitForNextFlight()* nos dada a informação que a entidade *Hostess* deve esperar pelo próximo voo e que é necessário atualizar e guardar o seu estado e esperar que o avião esteja pronto para embarque.

A fim de colocar a função em análise a trabalhar segundo estas indicações decrementamos o semáforo *mutex*, entrando na região crítica do código, e procedemos, neste caso, à atualização do estado da entidade *Hostess* para WAIT_FOR_FLIGHT, guardando esta mudança. Após isto incrementamos o valor do semáforo *mutex* saíndo da região crítica. Antes de encerrar a função decrementamos o valor do semáforo *readyForBoarding* através de um *semDown()* para que *Hostess*, que estava à espera de indicação para poder iniciar o processo de embarque, receba e execute essa indicação.

## Função waitForPassenger()

```
174  static void waitForPassenger ()
175  {
176      if (semDown (semgid, sh->mutex) == -1){
177          perror ("error on the down operation for semaphore access (HT)");
178          exit (EXIT_FAILURE);
179      }
180
181      /* insert your code here */
182
183      sh->fSt.st.hostessStat = WAIT_FOR_PASSENGER;
184      saveState(nFic,&sh->fSt);
185
186      if (semUp (semgid, sh->mutex) == -1){
187          perror ("error on the up operation for semaphore access (HT)");
188          exit (EXIT_FAILURE);
189      }
190
191      /* insert your code here */
192
193      if (semDown (semgid, sh->passengersInQueue) == -1){
194          perror ("error on the down operation for semaphore access (HT)");
195          exit (EXIT_FAILURE);
196      }
197
198  }
```

Na função *waitForPassenger()* a entidade *Hostess* deve esperar pela entidade *Passenger*, isto é *Hostess* espera que os vários *Passenger* cheguem ao aeroporto. O estado de *Hostess* deve ser atualizado e guardado.

Para o correto funcionamento da função é executado um *semDown()* de *mutex* para entrar na região crítica. Ao entrar na região crítica atualizamos o estado de *Hostess* para WAIT_FOR_PASSENGER e guardamo-lo. Após incrementarmos o valor do semáforo *mutex* para fora da região crítica decrementamos o valor do semáforo *passengersInQueue* para que *Hostess* inicie o atendimento ao primeiro passageiro da fila.

## Função checkPassport()

```
213   static bool checkPassport()
214   {
215       bool last;
216
217       /* insert your code here */
218
219       if (semUp (semgid, sh->passengersWaitInQueue) == -1){
220           perror ("error on the up operation for semaphore access (HT)");
221           exit (EXIT_FAILURE);
222       }
223
224       if (semDown (semgid, sh->mutex) == -1){
225           perror ("error on the down operation for semaphore access (HT)");
226           exit (EXIT_FAILURE);
227       }
228
229       /* insert your code here */
230
231       sh->fSt.st.hostessStat = CHECK_PASSPORT;
232       saveState(nFic,&sh->fSt);
233
234       if (semUp (semgid, sh->mutex) == -1){
235           perror ("error on the up operation for semaphore access (HT)");
236           exit (EXIT_FAILURE);
237       }
238
239       /* insert your code here */
240
241       if (semDown (semgid, sh->idShown) == -1){
242           perror ("error on the down operation for semaphore access (HT)");
243           exit (EXIT_FAILURE);
244       }
245
246       if (semDown (semgid, sh->mutex) == -1){
247           perror ("error on the down operation for semaphore access (HT)");
248           exit (EXIT_FAILURE);
249       }
250
251       /* insert your code here */
252
253       sh->fSt.nPassInQueue--;
254       sh->fSt.nPassInFlight++;
255       sh->fSt.totalPassBoarded++;
256       savePassengerChecked(nFic, &sh->fSt);
257       saveState(nFic,&sh->fSt);
258
259       if (nPassengersInFlight() == MAXFC || (nPassengersInFlight() >= MINFC && nPassengersInQueue() == 0) || sh->fSt.totalPassBoarded == N)
260           last = true;
261       else
262           last = false;
263
264       if (semUp (semgid, sh->mutex) == -1){                                    /* exit critical region */
265           perror ("error on the up operation for semaphore access (HT)");
266           exit (EXIT_FAILURE);
267       }
268
269       /* insert your code here */
270
271       return last;
272   }
```

A função *checkPassport()* tem como objetivo simular a operação de validação do passaporte. Nesta função a *Hostess* verifica o passaporte do *Passenger* e espera que o mesmo o mostre. O estado de *Hostess* deve ser guardado e atualizado duas vezes. A função deve ainda retornar um valor booleano que deve ser true caso o avião reúna as condições necessária para decolar, ou false caso não reúna condições.

A função começa com um *semUp()* do semáforo *passengersWaitInQueue* para que a *Hostess* informe o primeiro *Passenger* da fila que esta está disponível para o atender e que a espera do mesmo pela disponibilidade da primeira terminou. Seguidamente é decrementado o *mutex* para entrar na região crítica e atualizar o estado de *Hostess* para CHECK_PASSPORT e em seguida guardá-lo. O *mutex* sai da região crítica com um *semUp()* e é dado um *semDown()* no semáforo *idShown* para que a *Hostess* receba o passaporte do *Passenger* e o valide. O *mutex* entra novamente na região crítica para decrementar a variável que contabiliza o número de *Passengers* na fila e para incrementar as variáveis do número de *Passengers* que estão no voo e o total deles que já embarcaram desde o início. Após isto é impressa a informação de qual passageiro foi verificado e entrou no voo. É guardado o estado novamente. Avalia-se agora se o avião confere as condições necessárias para decolar, sendo elas se este está totalmente lotado, se atingiu o número mínimo de *Passengers* que lhe permita decolar e não houver mais nenhum na fila para embarcar ou se já tiverem embarcado todos os *Passenger* que estavam previstos. Caso uma destas condições se verifique a função irá retornar no fim um valor true, caso contrário um valor false. Antes de terminar a função retiramos o *mutex* da região crítica novamente.

## Função signalReadyToFlight()

```
293   void  signalReadyToFlight()
294   {
295       if (semDown (semgid, sh->mutex) == -1) {
296           perror ("error on the  operation for semaphore access (HT)");
297           exit (EXIT_FAILURE);
298       }
299
300       /* insert your code here */
301
302       sh->fSt.st.hostessStat = READY_TO_FLIGHT;
303       saveState(nFic,&sh->fSt);
304
305       int i = 0;
306       int c = 0;
307       while(sh->fSt.nPassengersInFlight[i] > 0){
308           c+= sh->fSt.nPassengersInFlight[i];
309           i++;
310       }
311
312       sh->fSt.nPassengersInFlight[i] = sh->fSt.totalPassBoarded - c;
313       saveFlightDeparted(nFic,&sh->fSt);
314
315       if(sh->fSt.totalPassBoarded == N){
316           sh->fSt.finished = true;
317       }
318
319       if (semUp (semgid, sh->mutex) == -1){
320           perror ("error on the up operation for semaphore access (HT)");
321           exit (EXIT_FAILURE);
322       }
323
324       /* insert your code here */
325
326       if (semUp (semgid, sh->readyToFlight) == -1){
327           perror ("error on the up operation for semaphore access (HT)");
328           exit (EXIT_FAILURE);
329       }
330
331   }
```

Na função *signalReadyToFlight()*, como o nome sugere, é dado um sinal ao *Pilot* que pode iniciar o voo. Nela a *Hostess* irá atualizar o seu estado, contabilizar o número de passageiros no voo e verificar se a ponte aérea já foi finalizada, isto é, se todos os passageiros previstos já embarcaram. Depois disto informa o *Pilot* se este pode ou não decolar e o estado é guardado.

Inicia-se novamente com uma entrada na região crítica de *mutex* onde se atualiza o estado da *Hostess* para READY_TO_FLIGHT e posteriormente se guarda. De seguida a *Hostess* calcula quantos *Passenger* estão no voo que irá decolar pela diferença entre o número total dos que embarcaram na ponte e o número total dos que viajaram nos voos anteriores. É chamada uma função que dá "print" da informação de como o voo decolou. Caso o número total de *Passenger* que já embarcaram durante a ponte seja igual ao esperado para aquela ponte então a *Hostess* deverá informar que a mesma está finalizada. Por fim retira-se o *mutex* da região crítica com um incremento do seu valor e faz-se um *semUp()* do semáforo *readyToFlight* para que o *Pilot* receba a indicação de que pode decolar.

## Função flight()

```
137  static void flight (bool go)
138  {
139      if (semDown (semgid, sh->mutex) == -1){
140          perror ("error on the down operation for semaphore access (PT)");
141          exit (EXIT_FAILURE);
142      }
143
144      /* insert your code here */
145
146      if (go)
147          sh->fSt.st.pilotStat=FLYING;
148      else
149          sh->fSt.st.pilotStat=FLYING_BACK;
150      saveState(nFic, &sh->fSt);
151
152      if (semUp (semgid, sh->mutex) == -1){
153          perror ("error on the up operation for semaphore access (PT)");
154          exit (EXIT_FAILURE);
155      }
156
157      usleep((unsigned int) floor ((MAXFLIGHT * random ()) / RAND_MAX + 100.0));
158  }
```

A função *flight()* recebe como parâmetro de entrada a variável booleana *go* que serve de indicação para se o *Pilot* se encontra a voar em direção ao destino ou de volta ao aeroporto de partida. A função deve programar assim a ação do piloto relativamente ao seu trajeto e o seu estado deve ser salvo pelo menos uma vez.

Assim decrementamos o semáforo *mutex*, entrando na região crítica do código, e procedemos, neste caso, à atualização do estado da entidade *Pilot* para FLYING**,** se o voo que está a ser feito for para o destino, ou FLYING_BACK caso seja no sentido oposto, guardando depois esta mudança de estado. Após isto executamos um *semUp()* ao valor do semáforo *mutex*, saindo da região crítica. A função termina gerando um tempo de sleep aleatório baseado na macro definida de MAXFLIGHT que será usado para simular o tempo de chegada de cada passageiro ao aeroporto.

## Função signalReadyForBoarding()

```
168  static void signalReadyForBoarding ()
169  {
170      if (semDown (semgid, sh->mutex) == -1){
171          perror ("error on the down operation for semaphore access (PT)");
172          exit (EXIT_FAILURE);
173      }
174
175      /* insert your code here */
176
177      sh->fSt.st.pilotStat=READY_FOR_BOARDING;
178      sh->fSt.nFlight++;
179      saveState(nFic, &sh->fSt);
180      saveStartBoarding(nFic, &sh->fSt);
181
182      if (semUp (semgid, sh->mutex) == -1){
183          perror ("error on the up operation for semaphore ore access (PT)");
184          exit (EXIT_FAILURE);
185      }
186
187      /* insert your code here */
188
189      if (semUp (semgid, sh->readyForBoarding) == -1){
190          perror ("error on the up operation for semaphore ore access (PT)");
191          exit (EXIT_FAILURE);
192      }
193
194  }
```

Na função signalReadyForBoarding() a entidade *Pilot* deve informar a entidade *Hostess* que o voo está pronto para iniciar o processo de embarque. O estado de *Pilot* deve ser atualizado e guardado, assim como deve ser atualizado o número do voo.

Para o correto funcionamento da função é executado um *semDown()* de *mutex* para este entrar na região crítica. Ao entrar na região crítica atualizamos o estado de *Pilot* para READY_FOR_BOARDING e guardamo-lo de seguida. Também o número do voo *nFlight* é incrementado e guardamo-lo. Chamamos também a função que imprime a informação de que o voo vai começar a faze de embarque. Após incrementarmos o valor do semáforo *mutex* para fora da região crítica incrementamos o valor do semáforo *readyForBoarding* para que *Pilot* possa informar a *Hostess* que o avião está pronto para iniciar a fase de embarque.

## Função waitUntilReadyToFlight()

```
203  static void waitUntilReadyToFlight ()
204  {
205      if (semDown (semgid, sh->mutex) == -1){
206          perror ("error on the down operation for semaphore access (PT)");
207          exit (EXIT_FAILURE);
208      }
209
210      /* insert your code here */
211
212      sh->fSt.st.pilotStat=WAITING_FOR_BOARDING;
213      saveState(nFic, &sh->fSt);
214
215      if (semUp (semgid, sh->mutex) == -1){
216          perror ("error on the up operation for semaphore access (PT)");
217          exit (EXIT_FAILURE);
218      }
219
220      /* insert your code here */
221
222      if (semDown (semgid, sh->readyToFlight) == -1){
223          perror ("error on the down operation for semaphore access (PT)");
224          exit (EXIT_FAILURE);
225      }
226  }
```

Na função *waitUntilReadyToFlight()* a entidade *Pilot* deve esperar que o avião tenha condições para decolar. O estado de *Pilot* deve ser atualizado e guardado.

A função começa com um *semDown()* de *mutex*. Ao entrar na região crítica atualizamos o estado de *Pilot* para WAITING_FOR_BOARDING guardando-o. Saímos da região crítica do semáforo *mutex* e decrementamos o valor do semáforo *readyToFlight* para que *Pilot* confirme a receção da informação de que o avião está pronto a decolar.

## Função dropPassengersAtTarget()

```
236  static void dropPassengersAtTarget ()
237  {
238      if (semDown (semgid, sh->mutex) == -1){
239          perror ("error on the down operation for semaphore access (PT)");
240          exit (EXIT_FAILURE);
241      }
242
243      /* insert your code here */
244
245      saveFlightArrived(nFic, &sh->fSt);
246      sh->fSt.st.pilotStat=DROPING_PASSENGERS;
247      saveState(nFic, &sh->fSt);
248
249      for (int a = sh->fSt.nPassInFlight; a > 0; a--)
250      {
251          if (semUp (semgid, sh->passengersWaitInFlight) == -1){
252              perror ("error on the up operation for semaphore access (PT)");
253              exit (EXIT_FAILURE);
254          }
255      }
256
257      if (semUp (semgid, sh->mutex) == -1)  {
258          perror ("error on the up operation for semaphore access (PT)");
259          exit (EXIT_FAILURE);
260      }
261
262      /* insert your code here */
263
264      if (semDown (semgid, sh->planeEmpty) == -1) {
265          perror ("error on the down operation for semaphore access (PT)");
266          exit (EXIT_FAILURE);
267      }
268
269      if (semDown (semgid, sh->mutex) == -1) {
270          perror ("error on the down operation for semaphore access (PT)");
271          exit (EXIT_FAILURE);
272      }
273
274      /* insert your code here */
275
276      saveFlightReturning(nFic, &sh->fSt);
277
278      if (semUp (semgid, sh->mutex) == -1)  {
279          perror ("error on the up operation for semaphore access (PT)");
280          exit (EXIT_FAILURE);
281      }
282  }
```

A função *dropPassengersAtTarget()* é responsável por colocar em execução ações como o *Pilot* deixar os *Passenger* saírem do avião quando chegam ao seu destino. Apenas após todos

os passageiros deixarem o avião o piloto pode voltar para o outro lado da ponte aérea. O estado do piloto é alterado e guardado uma vez.

Para executar estas ações decrementamos o *mutex* para entrar na região crítica. Enquanto o seu valor se encontra a 0 a função chama outra função *saveFlightArrived()* que imprime a indicação de quando o avião chega ao destino seguida da informação referente à lotação do avião, que começa em seguida a esvaziar. O estado do *Pilot* é alterado e guardado em DROPPING_PASSENGERS e o semáforo *passengersWaitInFlight* é incrementado o número de vezes equivalente ao número de passageiros no voo para que todos os passageiros esperem pelo voo terminar para saírem do avião. Retiramos o *mutex* da região crítica e o *Pilot* espera avião esteja vazio, momento em que decrementa o semáforo *planeEmpty*. Volta a entrar na região crítica do *mutex*, desta vez para chamar uma função que imprime a indicação de quando o avião volta para o ponto de partida e respetivas atualizações de estado entre as entidades até se iniciar um novo embarque, caso ainda seja possível.

# Passenger

## Função waitInQueue()

```
141  static void waitInQueue (unsigned int passengerId)
142  {
143      if (semDown (semgid, sh->mutex) == -1){
144          perror ("error on the down operation for semaphore access (PG)");
145          exit (EXIT_FAILURE);
146      }
147
148      /* insert your code here */
149
150      sh->fSt.nPassInQueue++;
151      sh->fSt.st.passengerStat[passengerId]=IN_QUEUE;
152      saveState(nFic, &sh->fSt);
153
154      if (semUp (semgid, sh->mutex) == -1){
155          perror ("error on the up operation for semaphore access (PG)");
156          exit (EXIT_FAILURE);
157      }
158
159      /* insert your code here */
160
161      if (semUp (semgid, sh->passengersInQueue) == -1){
162          perror ("error on the up operation for semaphore access (PG)");
163          exit (EXIT_FAILURE);
164      }
165
166      if (semDown (semgid, sh->passengersWaitInQueue) == -1){
167          perror ("error on the down operation for semaphore access (PG)");
168          exit (EXIT_FAILURE);
169      }
170
171      if (semDown (semgid, sh->mutex) == -1){
172          perror ("error on the down operation for semaphore access (PG)");
173          exit (EXIT_FAILURE);
174      }
```

```
175
176     /* insert your code here */
177
178     sh->fSt.st.passengerStat[passengerId]=IN_FLIGHT;
179     saveState(nFic, &sh->fSt);
180
181     if (semUp (semgid, sh->mutex) == -1){
182         perror ("error on the up operation for semaphore access (PG)");
183         exit (EXIT_FAILURE);
184     }
185
186     /* insert your code here */
187
188     sh->fSt.passengerChecked=passengerId;
189
190     if (semUp (semgid, sh->idShown) == -1){
191         perror ("error on the up operation for semaphore access (PG)");
192         exit (EXIT_FAILURE);
193     }
194 }
```

Esta função é responsável por simular a ação de um *Passenger* esperar pela sua vez para ter a sua identificação validada pela *Hostess* e recebe como argumentos de entrada a variável *passengerId* que corresponde ao Id do passageiro a analidar. Nesta função *Passenger* deve assim atualizar o número de entidades do mesmo tipo na fila de espera e informar a *Hostess* que o mesmo está pronto para embarcar. O estado do mesmo deve ser guardado duas vezes.

Começando por entrar na região crítica de *mutex* o número de passageiros na fila *nPassInQueue* é incrementado e o estado de *Passenger* de Id *passengerId* é atualizado e guardado como IN_QUEUE. Incrementamos o semáforo *mutex*, incrementamos o semáforo *passengersInQueue* para que a H*ostess* receba a informação que já tem passageiros na fila para poder verificar. Posteriormente decrementamos os semáforos *passengerWaitInQueue*, pois *Passenger* já deixou de estar à espera de *Hostess* para ser atendido, e *mutex*, para reentrar na região crítica, por esta ordem. Atualizamos e guardamos o estado do *Passenger* de Id *passengerId* para IN_FLIGHT, incrementamos o *mutex*, atualizamos a variável de *passengerChecked* para o Id dado como argumento à função e termina o programa incrementando *idShown* para iniciar o processo de verificação do Id de *Passenger*.

## Função waitUntilDestination()

```
207  static void waitUntilDestination (unsigned int passengerId)
208  {
209
210      /* insert your code here */
211
212      if (semDown (semgid, sh->passengersWaitInFlight) == -1){
213          perror ("error on the down operation for semaphore access (PG)");
214          exit (EXIT_FAILURE);
215      }
216
217      if (semDown (semgid, sh->mutex) == -1){
218          perror ("error on the down operation for semaphore access (PG)");
219          exit (EXIT_FAILURE);
220      }
221
222      /* insert your code here */
223
224      sh->fSt.st.passengerStat[passengerId]=AT_DESTINATION;
225      sh->fSt.nPassInFlight--;
226      saveState(nFic, &sh->fSt);
227
228      if(sh->fSt.nPassInFlight == 0){
229          if (semUp (semgid, sh->planeEmpty) == -1){
230              perror ("error on the up operation for semaphore access (PG)");
231              exit (EXIT_FAILURE);
232          }
233      }
234
235      if (semUp (semgid, sh->mutex) == -1){
236          perror ("error on the up operation for semaphore access (PG)");
237          exit (EXIT_FAILURE);
238      }
239  }
```

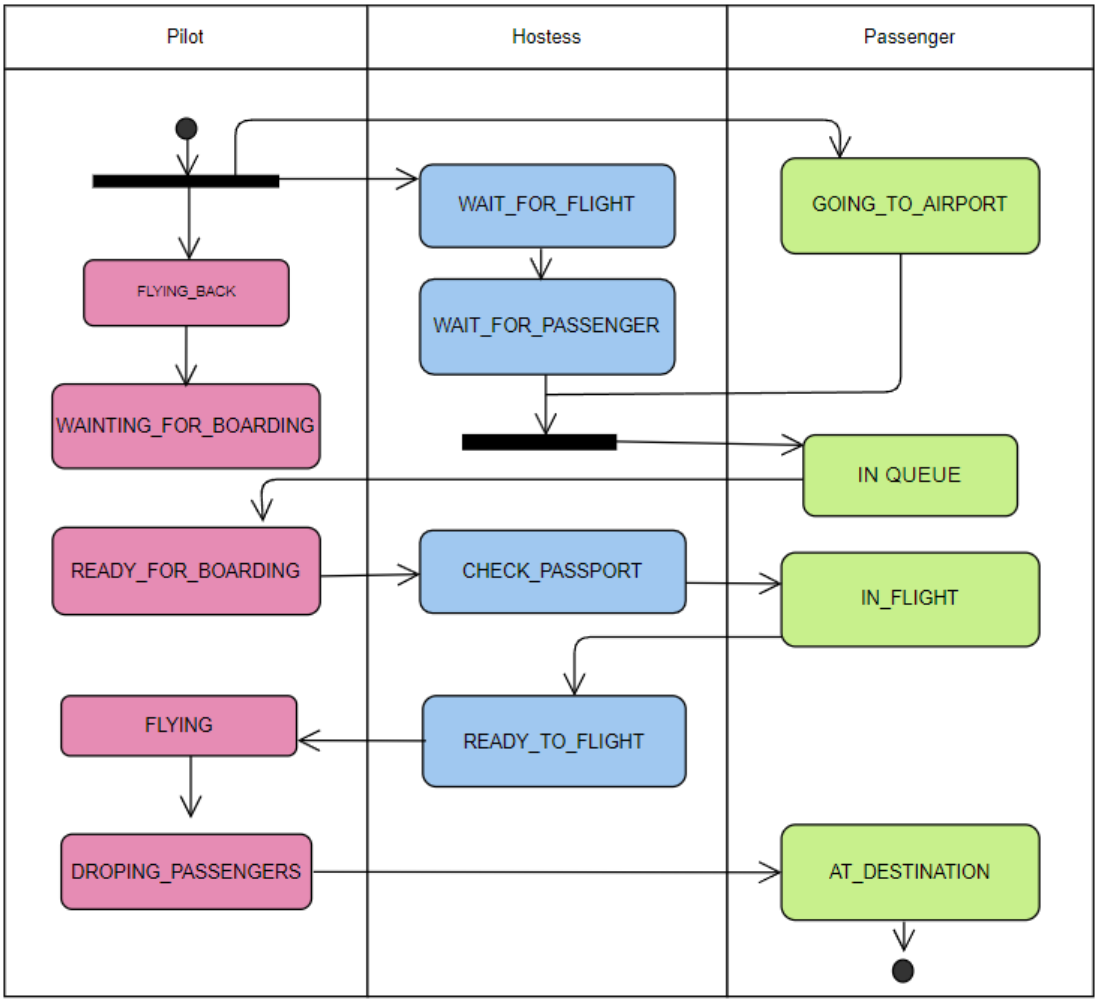
Na função *waitUntilDestination* deve ser simulada a situação em que os *Passenger* dentro do avião esperam que o voo termine e chegue ao destino. O *Passenger* deve esperar que o voo acabe, atualizar o número de Passenger e o último que sai do avião deve informar o Pilot que o avião está vazio. O estado é atualizado uma vez e a função recebe o Id do *Passenger* como argumento de entrada.

A função começa com um decremento do semáforo *passengersWaitInFlight* para o *Passenger* receber a informação de que o voo chegou ao destino. Também decrementamos o *mutex* e atualizamos o estado do *Passenger* de Id *passengerId* para AT_DESTINATION e guardamo-lo. Decrementamos o valor de *nPassInFlight* e se o valor de *nPassInFlight* for igual a 0, ou seja se o avião estiver vazio, fazemos *semUp()* ao semáforo *planeEmpty* para enviar a

informação ao *Pilot* de que o avião está vazio. A função termina com a saída da região crítica de *mutex.*

# Diagrama de Estados

O seguinte diagrama representa esquematicamente o funcionamento da ponte aérea através da atualização de estados de cada entidade.



# Validação de Resultados

```
                                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   0   0   0   0   0   0
  1  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
  2  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
  2  1    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  2  1    0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
  2  2    0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
  2  2    0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
Flight 1 : Passenger 8 checked
  2  2    0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1
  2  1    0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1
  2  1    0   0   0   0   1   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   1   1   1
  2  2    0   0   0   0   1   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   1   1   1
  2  2    0   0   0   0   2   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   1   1   1
Flight 1 : Passenger 4 checked
  2  2    0   0   0   0   2   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   2   2
  2  1    0   0   0   0   2   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   2   2
  2  1    0   0   0   0   2   0   0   0   2   0   0   0   0   0   0   0   0   0   1   0   0   1   2   2
  2  2    0   0   0   0   2   0   0   0   2   0   0   0   0   0   0   0   0   0   1   0   0   1   2   2
  2  2    0   0   0   0   2   0   0   0   2   0   0   0   0   0   0   0   0   0   2   0   0   1   2   2
Flight 1 : Passenger 18 checked
  2  2    0   0   0   0   2   0   0   0   2   0   0   0   0   0   0   0   0   0   2   0   0   0   3   3
  2  1    0   0   0   0   2   0   0   0   2   0   0   0   0   0   0   0   0   0   2   0   0   0   3   3
  2  1    1   0   0   0   2   0   0   0   2   0   0   0   0   0   0   0   0   0   2   0   0   1   3   3
  2  2    1   0   0   0   2   0   0   0   2   0   0   0   0   0   0   0   0   0   2   0   0   1   3   3
  2  2    2   0   0   0   2   0   0   0   2   0   0   0   0   0   0   0   0   0   2   0   0   1   3   3
Flight 1 : Passenger 0 checked
  2  2    2   0   0   0   2   0   0   0   2   0   0   0   0   0   0   0   0   0   2   0   0   0   4   4
  2  1    2   0   0   0   2   0   0   0   2   0   0   0   0   0   0   0   0   0   2   0   0   0   4   4
  2  1    2   0   0   0   2   0   0   1   2   0   0   0   0   0   0   0   0   0   2   0   0   1   4   4
  2  2    2   0   0   0   2   0   0   1   2   0   0   0   0   0   0   0   0   0   2   0   0   1   4   4
  2  2    2   0   0   0   2   0   0   2   2   0   0   0   0   0   0   0   0   0   2   0   0   1   4   4
Flight 1 : Passenger 7 checked
  2  2    2   0   0   0   2   0   0   2   2   0   0   0   0   0   0   0   0   0   2   0   0   0   5   5
  2  3    2   0   0   0   2   0   0   2   2   0   0   0   0   0   0   0   0   0   2   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    2   0   0   0   2   0   0   2   2   0   0   0   0   0   0   0   0   0   2   0   0   0   5   5
  3  0    2   0   0   0   2   0   0   2   2   0   0   0   0   0   0   0   0   0   2   0   0   0   5   5
  3  0    2   0   0   0   2   0   0   2   2   0   0   0   0   0   1   0   0   0   2   0   0   1   5   5
  3  0    2   0   0   1   2   0   0   2   2   0   0   0   0   0   1   0   0   0   2   0   0   2   5   5
  3  0    2   0   0   1   2   0   0   2   2   0   1   0   0   0   1   0   0   0   2   0   0   3   5   5
Flight 1 : Arrived
 PT HT  P00 P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20  InQ InF toB
  4  0    2   0   0   1   2   0   0   2   2   0   1   0   0   0   1   0   0   0   2   0   0   3   5   5
  4  0    2   0   0   1   2   0   0   2   2   0   1   0   0   0   1   0   0   1   2   0   0   4   5   5
  4  0    2   0   0   1   2   0   0   2   3   0   1   0   0   0   1   0   0   1   2   0   0   4   4   5
  4  0    2   0   0   1   3   0   0   2   3   0   1   0   0   0   1   0   0   1   2   0   0   4   3   5
  4  0    3   0   0   1   3   0   0   2   3   0   1   0   0   0   1   0   0   1   2   0   0   4   2   5
  4  0    3   0   0   1   3   0   0   2   3   0   1   0   0   0   1   0   0   1   3   0   0   4   1   5
  4  0    3   0   0   1   3   0   0   3   3   0   1   0   0   0   1   0   0   1   3   0   0   4   0   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    3   0   0   1   3   0   0   3   3   0   1   0   0   0   1   0   0   1   3   0   0   4   0   5
  1  0    3   0   0   1   3   0   0   3   3   0   1   0   0   0   1   0   0   1   3   0   0   4   0   5
  2  0    3   0   0   1   3   0   0   3   3   0   1   0   0   0   1   0   0   1   3   0   0   4   0   5
  2  1    3   0   0   1   3   0   0   3   3   0   1   0   0   0   1   0   0   1   3   0   0   4   0   5
  2  2    3   0   0   1   3   0   0   3   3   0   1   0   0   0   1   0   0   1   3   0   0   4   0   5
  2  2    3   0   0   1   3   0   0   3   3   0   1   0   0   1   1   0   0   1   3   0   0   5   0   5
  2  2    3   0   0   1   3   0   0   3   3   0   1   0   0   1   2   0   0   1   3   0   0   5   0   5
```

```
Flight 2 : Passenger 14 checked
  2  2    3   0   0   1   3   0   0   3   3   0   1   0   0   1   2   0   0   1   3   0   0    4   1   6
  2  1    3   0   0   1   3   0   0   3   3   0   1   0   0   1   2   0   0   1   3   0   0    4   1   6
  2  2    3   0   0   1   3   0   0   3   3   0   1   0   0   1   2   0   0   1   3   0   0    4   1   6
  2  2    3   0   0   2   3   0   0   3   3   0   1   0   0   1   2   0   0   1   3   0   0    4   1   6
Flight 2 : Passenger 3 checked
  2  2    3   0   0   2   3   0   0   3   3   0   1   0   0   1   2   0   0   1   3   0   0    3   2   7
  2  1    3   0   0   2   3   0   0   3   3   0   1   0   0   1   2   0   0   1   3   0   0    3   2   7
  2  2    3   0   0   2   3   0   0   3   3   0   1   0   0   1   2   0   0   1   3   0   0    3   2   7
  2  2    3   0   0   2   3   0   0   3   3   0   2   0   0   1   2   0   0   1   3   0   0    3   2   7
Flight 2 : Passenger 10 checked
  2  2    3   0   0   2   3   0   0   3   3   0   2   0   0   1   2   0   0   1   3   0   0    2   3   8
  2  2    3   0   0   2   3   0   0   3   3   0   2   0   0   1   2   0   0   1   3   0   1    3   3   8
  2  1    3   0   0   2   3   0   0   3   3   0   2   0   0   1   2   0   0   1   3   0   1    3   3   8
  2  2    3   0   0   2   3   0   0   3   3   0   2   0   0   1   2   0   0   1   3   0   1    3   3   8
  2  2    3   0   0   2   3   0   0   3   3   0   2   0   0   1   2   0   0   2   3   0   1    3   3   8
Flight 2 : Passenger 17 checked
  2  2    3   0   0   2   3   0   0   3   3   0   2   0   0   1   2   0   0   2   3   0   1    2   4   9
  2  1    3   0   0   2   3   0   0   3   3   0   2   0   0   1   2   0   0   2   3   0   1    2   4   9
  2  2    3   0   0   2   3   0   0   3   3   0   2   0   0   1   2   0   0   2   3   0   1    2   4   9
  2  2    3   0   0   2   3   0   0   3   3   0   2   0   0   2   2   0   0   2   3   0   1    2   4   9
Flight 2 : Passenger 13 checked
  2  2    3   0   0   2   3   0   0   3   3   0   2   0   0   2   2   0   0   2   3   0   1    1   5  10
  2  1    3   0   0   2   3   0   0   3   3   0   2   0   0   2   2   0   0   2   3   0   1    1   5  10
  2  2    3   0   0   2   3   0   0   3   3   0   2   0   0   2   2   0   0   2   3   0   1    1   5  10
  2  2    3   0   0   2   3   0   0   3   3   0   2   0   0   2   2   0   0   2   3   0   2    1   5  10
Flight 2 : Passenger 20 checked
  2  2    3   0   0   2   3   0   0   3   3   0   2   0   0   2   2   0   0   2   3   0   2    0   6  11
  2  3    3   0   0   2   3   0   0   3   3   0   2   0   0   2   2   0   0   2   3   0   2    0   6  11
Flight 2 : Departed with 6 passengers
 PT HT  P00 P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20  InQ InF toB
  2  0    3   0   0   2   3   0   0   3   3   0   2   0   0   2   2   0   0   2   3   0   2    0   6  11
  3  0    3   0   0   2   3   0   0   3   3   0   2   0   0   2   2   0   0   2   3   0   2    0   6  11
  3  0    3   0   1   2   3   0   0   3   3   0   2   0   0   2   2   0   0   2   3   0   2    1   6  11
  3  0    3   0   1   2   3   0   0   3   3   0   2   0   0   2   2   0   1   2   3   0   2    2   6  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
  4  0    3   0   1   2   3   0   0   3   3   0   2   0   0   2   2   0   1   2   3   0   2    2   6  11
  4  0    3   0   1   2   3   0   0   3   3   0   2   0   0   2   3   0   1   2   3   0   2    2   5  11
  4  0    3   0   1   3   3   0   0   3   3   0   2   0   0   2   3   0   1   2   3   0   2    2   4  11
  4  0    3   0   1   3   3   0   0   3   3   0   2   0   0   3   3   0   1   2   3   0   2    2   3  11
  4  0    3   0   1   3   3   0   0   3   3   0   2   0   0   3   3   0   1   3   3   0   2    2   2  11
  4  0    3   0   1   3   3   0   0   3   3   0   3   0   0   3   3   0   1   3   3   0   2    2   1  11
  4  0    3   0   1   3   3   0   0   3   3   0   3   0   0   3   3   0   1   3   3   0   3    2   0  11
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    3   0   1   3   3   0   0   3   3   0   3   0   0   3   3   0   1   3   3   0   3    2   0  11
  1  0    3   0   1   3   3   0   0   3   3   0   3   0   0   3   3   0   1   3   3   0   3    2   0  11
  2  0    3   0   1   3   3   0   0   3   3   0   3   0   0   3   3   0   1   3   3   0   3    2   0  11
  2  1    3   0   1   3   3   0   0   3   3   0   3   0   0   3   3   0   1   3   3   0   3    2   0  11
  2  2    3   0   1   3   3   0   0   3   3   0   3   0   0   3   3   0   1   3   3   0   3    2   0  11
  2  2    3   0   2   3   3   0   0   3   3   0   3   0   0   3   3   0   1   3   3   0   3    2   0  11
  2  2    3   0   2   3   3   1   0   3   3   0   3   0   0   3   3   0   1   3   3   0   3    3   0  11
Flight 3 : Passenger 2 checked
  2  2    3   0   2   3   3   1   0   3   3   0   3   0   0   3   3   0   1   3   3   0   3    2   1  12
  2  1    3   0   2   3   3   1   0   3   3   0   3   0   0   3   3   0   1   3   3   0   3    2   1  12
  2  2    3   0   2   3   3   1   0   3   3   0   3   0   0   3   3   0   1   3   3   0   3    2   1  12
  2  2    3   0   2   3   3   1   0   3   3   0   3   0   0   3   3   0   2   3   3   0   3    2   1  12
Flight 3 : Passenger 16 checked
  2  2    3   0   2   3   3   1   0   3   3   0   3   0   0   3   3   0   2   3   3   0   3    1   2  13
  2  1    3   0   2   3   3   1   0   3   3   0   3   0   0   3   3   0   2   3   3   0   3    1   2  13
  2  2    3   0   2   3   3   1   0   3   3   0   3   0   0   3   3   0   2   3   3   0   3    1   2  13
  2  2    3   0   2   3   3   1   0   3   3   0   3   0   1   3   3   0   2   3   3   0   3    2   2  13
  2  2    3   0   2   3   3   2   0   3   3   0   3   0   1   3   3   0   2   3   3   0   3    2   2  13
```

```
Flight 3 : Passenger 5 checked
  2  2    3   0   2   3   3   2   0   3   3   0   3   0   1   3   3   0   2   3   3   0   3    1   3  14
  2  1    3   0   2   3   3   2   0   3   3   0   3   0   1   3   3   0   2   3   3   0   3    1   3  14
  2  2    3   0   2   3   3   2   0   3   3   0   3   0   1   3   3   0   2   3   3   0   3    1   3  14
  2  2    3   0   2   3   3   2   0   3   3   0   3   0   2   3   3   0   2   3   3   0   3    1   3  14
Flight 3 : Passenger 12 checked
  2  2    3   0   2   3   3   2   0   3   3   0   3   0   2   3   3   0   2   3   3   0   3    0   4  15
  2  1    3   0   2   3   3   2   0   3   3   0   3   0   2   3   3   0   2   3   3   0   3    0   4  15
  2  1    3   0   2   3   3   2   0   3   3   0   3   0   2   3   3   0   2   3   3   1   3    1   4  15
  2  2    3   0   2   3   3   2   0   3   3   0   3   0   2   3   3   0   2   3   3   1   3    1   4  15
  2  2    3   0   2   3   3   2   0   3   3   0   3   0   2   3   3   0   2   3   3   2   3    1   4  15
Flight 3 : Passenger 19 checked
  2  2    3   0   2   3   3   2   0   3   3   0   3   0   2   3   3   0   2   3   3   2   3    0   5  16
  2  3    3   0   2   3   3   2   0   3   3   0   3   0   2   3   3   0   2   3   3   2   3    0   5  16
Flight 3 : Departed with 5 passengers
 PT HT   P00 P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20   InQ InF toB
  2  0    3   0   2   3   3   2   0   3   3   0   3   0   2   3   3   0   2   3   3   2   3    0   5  16
  3  0    3   0   2   3   3   2   0   3   3   0   3   0   2   3   3   0   2   3   3   2   3    0   5  16
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  0    3   0   2   3   3   2   0   3   3   0   3   0   2   3   3   0   2   3   3   2   3    0   5  16
  4  0    3   0   3   3   3   2   0   3   3   0   3   0   2   3   3   0   2   3   3   2   3    0   4  16
  4  0    3   0   3   3   3   2   0   3   3   0   3   0   2   3   3   0   3   3   3   2   3    0   3  16
  4  0    3   0   3   3   3   3   0   3   3   0   3   0   2   3   3   0   3   3   3   2   3    0   2  16
  4  0    3   0   3   3   3   3   0   3   3   0   3   0   3   3   3   0   3   3   3   2   3    0   1  16
  4  0    3   0   3   3   3   3   0   3   3   0   3   0   3   3   3   0   3   3   3   3   3    0   0  16
  4  0    3   1   3   3   3   3   0   3   3   0   3   0   3   3   3   0   3   3   3   3   3    1   0  16
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
  0  0    3   1   3   3   3   3   0   3   3   0   3   0   3   3   3   0   3   3   3   3   3    1   0  16
  0  0    3   1   3   3   3   3   0   3   3   0   3   0   3   3   3   1   3   3   3   3   3    2   0  16
  1  0    3   1   3   3   3   3   0   3   3   0   3   0   3   3   3   1   3   3   3   3   3    2   0  16
  2  0    3   1   3   3   3   3   0   3   3   0   3   0   3   3   3   1   3   3   3   3   3    2   0  16
  2  1    3   1   3   3   3   3   0   3   3   0   3   0   3   3   3   1   3   3   3   3   3    2   0  16
  2  2    3   1   3   3   3   3   0   3   3   0   3   0   3   3   3   1   3   3   3   3   3    2   0  16
  2  2    3   2   3   3   3   3   0   3   3   0   3   0   3   3   3   1   3   3   3   3   3    2   0  16
Flight 4 : Passenger 1 checked
  2  2    3   2   3   3   3   3   0   3   3   0   3   0   3   3   3   1   3   3   3   3   3    1   1  17
  2  1    3   2   3   3   3   3   0   3   3   0   3   0   3   3   3   1   3   3   3   3   3    1   1  17
  2  2    3   2   3   3   3   3   0   3   3   0   3   0   3   3   3   1   3   3   3   3   3    1   1  17
  2  2    3   2   3   3   3   3   0   3   3   0   3   0   3   3   3   2   3   3   3   3   3    1   1  17
Flight 4 : Passenger 15 checked
  2  2    3   2   3   3   3   3   0   3   3   0   3   0   3   3   3   2   3   3   3   3   3    0   2  18
  2  1    3   2   3   3   3   3   0   3   3   0   3   0   3   3   3   2   3   3   3   3   3    0   2  18
  2  1    3   2   3   3   3   3   0   3   3   0   3   1   3   3   3   2   3   3   3   3   3    1   2  18
  2  2    3   2   3   3   3   3   0   3   3   0   3   1   3   3   3   2   3   3   3   3   3    1   2  18
  2  2    3   2   3   3   3   3   0   3   3   0   3   2   3   3   3   2   3   3   3   3   3    1   2  18
Flight 4 : Passenger 11 checked
  2  2    3   2   3   3   3   3   0   3   3   0   3   2   3   3   3   2   3   3   3   3   3    0   3  19
  2  1    3   2   3   3   3   3   0   3   3   0   3   2   3   3   3   2   3   3   3   3   3    0   3  19
  2  1    3   2   3   3   3   3   1   3   3   0   3   2   3   3   3   2   3   3   3   3   3    1   3  19
  2  2    3   2   3   3   3   3   1   3   3   0   3   2   3   3   3   2   3   3   3   3   3    1   3  19
  2  2    3   2   3   3   3   3   2   3   3   0   3   2   3   3   3   2   3   3   3   3   3    1   3  19
Flight 4 : Passenger 6 checked
  2  2    3   2   3   3   3   3   2   3   3   0   3   2   3   3   3   2   3   3   3   3   3    0   4  20
  2  1    3   2   3   3   3   3   2   3   3   0   3   2   3   3   3   2   3   3   3   3   3    0   4  20
  2  1    3   2   3   3   3   3   2   3   3   1   3   2   3   3   3   2   3   3   3   3   3    1   4  20
  2  2    3   2   3   3   3   3   2   3   3   1   3   2   3   3   3   2   3   3   3   3   3    1   4  20
  2  2    3   2   3   3   3   3   2   3   3   2   3   2   3   3   3   2   3   3   3   3   3    1   4  20
Flight 4 : Passenger 9 checked
  2  2    3   2   3   3   3   3   2   3   3   2   3   2   3   3   3   2   3   3   3   3   3    0   5  21
  2  3    3   2   3   3   3   3   2   3   3   2   3   2   3   3   3   2   3   3   3   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   2   3   3   2   3   2   3   3   3   2   3   3   3   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   2   3   3   2   3   2   3   3   3   2   3   3   3   3   3    0   5  21
  4  3    3   3   3   3   3   3   2   3   3   2   3   2   3   3   3   2   3   3   3   3   3    0   4  21
  4  3    3   3   3   3   3   3   2   3   3   2   3   2   3   3   3   3   3   3   3   3   3    0   3  21
  4  3    3   3   3   3   3   3   3   3   3   2   3   2   3   3   3   3   3   3   3   3   3    0   2  21
  4  3    3   3   3   3   3   3   3   3   3   2   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  6 passengers
Flight 3 took  5 passengers
Flight 4 took  5 passengers
```

A figura anterior é o resultado de um teste realizado à ponte aérea implementada pelas funções apresentadas anteriormente. PT e HT correspondem aos estados de *Pilot* e *Hostess.* Os diferentes P's correspondem aos 21 passageiros que embarcam na ponte. O InQ e o InF correspondem ao número de *Passengers* na fila de espera e no voo respetivamente. toB representa a quantidade total de *Passengers* que embarcaram na ponte aérea.

A fim de testar o código relativamente a deadlocks o código foi corrido para 80 mil tentativas aleatórias, sendo esta uma delas. O programa passou nesse teste com sucesso.

```
36   /** \brief pilot flying to starting airport */
37   #define  FLYING_BACK                  0
38   /** \brief pilot signals ready for boarding */
39   #define  READY_FOR_BOARDING           1
40   /** \brief pilot wait for boarding to complete */
41   #define  WAITING_FOR_BOARDING         2
42   /** \brief pilot takes passengers to destination */
43   #define  FLYING                       3
44   /** \brief pilot drops passengers at destination */
45   #define  DROPING_PASSENGERS           4
46
47   /* Hostess state constants */
48
49   /** \brief hostess waits for plane to be ready for boarding */
50   #define  WAIT_FOR_FLIGHT              0
51   /** \brief hostess waits for passenger to arrive */
52   #define  WAIT_FOR_PASSENGER           1
53   /** \brief hostess checks passenger passport */
54   #define  CHECK_PASSPORT               2
55   /** \brief hostess signals boarding is complete */
56   #define  READY_TO_FLIGHT              3
57
58   /* Passenger state constants */
59
60   /** \brief passenger is going to the airport */
61   #define  GOING_TO_AIRPORT             0
62   /** \brief passenger is waiting in queue */
63   #define  IN_QUEUE                     1
64   /** \brief passenger is flying */
65   #define  IN_FLIGHT                    2
66   /** \brief passenger arrives at destination */
67   #define  AT_DESTINATION               3
68
69   #endif /* PROBCONST_H_ */
70
```

Sendo que cada número do lado direito corresponde a um estado específico da entidade comentada em cabeçalho para cada grupo de macros podemos observar no resultado obtido que todas as entidades são atualizadas para os estados seguintes na altura correta, mediante aquilo que foi descrito nas funções. Também as informações sobre o voo são impressas corretamente.

No final do primeiro teste, que está colocado acima, podemos observar que o avião decolou com o número mínimo de passageiros, e com outros valores diferentes da lotação máxima, isto é possível pois o valor mínimo foi atingido e a fila de passageiros para embarque

se encontrava vazia. O número de passageiros na fila pode ser encontrado na coluna InQ. Também foram obtidos resultados de aviões que decolaram com a lotação máxima, como por exemplo no teste número 178.

```
38369    AirLift result
38370    AirLift used 4 Flights
38371    Flight 1 took  5 passengers
38372    Flight 2 took 10 passengers
38373    Flight 3 took  5 passengers
38374    Flight 4 took  1 passengers
```

Neste teste observamos que tanto o avião partiu com lotação máxima, como abaixo da mínima, uma vez que já não havia mais passageiros para embarcar para além do que embarcou no último voo da ponte.

# Conclusão

Este trabalho possibilitou-nos desenvolver conhecimentos sobre os mecanismos associados à execução e sincronização de processos e threads. Posto isto, o trabalho foi desenvolvido de forma adequada e de acordo com todos os requisitos propostos, visto que os objetivos propostos foram alcançados.

De um modo geral, a maior dificuldade foi entender o modo de formulação das funções necessárias para a implementação do projeto, visto que todo o código teria que ter uma estrutura e ordem exigente para o seu correto funcionamento. Durante a fase de testes todos os que realizados foram bem-sucedidos.