Rapsódia no aeroporto

A cena passa-se num aeroporto, algures nos arredores da cidade de Aveiro, e procura descrever as actividades ligadas ao desembarque de passageiros que aí decorrem habitualmente. Há oito locais principais: o local de desembarque dos passageiros do avião, as zonas de recolha de bagagens e de armazenamento temporário (caso dos passageiros em trânsito), o *guichet* de reclamação de bagagens perdidas, as zonas de transferência de terminal, a saída do terminal de desembarque e a entrada do terminal de embarque.

Os intervenientes são de três tipos: os passageiros que terminam a sua viagem no aeroporto ou que estão em trânsito, o bagageiro que descarrega as bagagens do avião, quando este aterra, e as transporta para as zonas de recolha de bagagens ou de armazenamento temporário e o motorista do autocarro de transferência que transporta os passageiros em trânsito entre as zonas de transferência de terminal.

Vão realizar-se K chegadas de aviões, envolvendo cada uma delas o desembarque de N passageiros. Cada passageiro transporta no porão do avião de 0 a M peças de bagagem. O autocarro de transferência entre terminais tem uma lotação de T lugares.

As actividades estão organizadas, chegada a chegada de avião, da forma seguinte

- os passageiros deslocam-se do local de desembarque para a zona de recolha de bagagens, se terminarem aqui a sua viagem e tiverem bagagens a recolher; aqueles que não tiverem bagagens a recolher, deslocam-se imediatamente para a saída do terminal de desembarque e abandonam o aeroporto; os restantes passageiros, aqueles que estão em trânsito, deslocam-se para a zona de transferência de terminal;
- após o desembarque de todos os passageiros, o bagageiro recolhe as bagagens do porão do avião, uma a uma, e distribui-as pelas zonas de recolha de bagagens e de armazenamento temporário, conforme se trate de bagagens pertencentes a passageiros que terminam a sua viagem no aeroporto ou que estão em trânsito;
- na zona de recolha de bagagens, os passageiros aguardam a chegada das bagagens próprias; quando tomam posse delas, uma a uma, dirigem-se para a saída do terminal de desembarque e abandonam o aeroporto; aqueles que verificam que nem todas as suas bagagens foram transportadas, vão primeiro ao *guichet* de reclamação de bagagens perdidas para preencherem um formulário de reclamação, antes de se dirigirem para a saída do terminal de desembarque e abandonarem o aeroporto;
- na zona de transferência de terminal, os passageiros em trânsito aguardam a chegada do autocarro que os vai transportar à zona de transferência do terminal de embarque, de onde prosseguem para a entrada do terminal;
- o autocarro parte do local de transferência do terminal desembarque segundo um horário pré-estabelecido efectuando um percurso circular que tem apenas como outra paragem o local de transferência do terminal de embarque; contudo, se, quando estacionado na zona de transferência do terminal de desembarque, se verificar que todos os lugares estão ocupados, o motorista pode partir mais cedo.

No final das operações, é emitido um relatório completo das actividades desenvolvidas.

Assuma que há cinco chegadas de aviões, cada uma envolvendo o desembarque de seis passageiros, transportando um máximo de duas peças de bagagem no porão do avião, e que a lotação do autocarro de transferência entre terminais é de três passageiros.

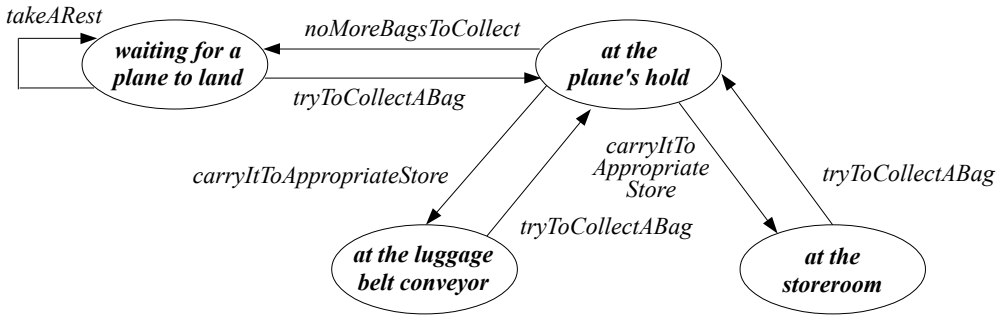
Construa uma simulação das actividades no aeroporto usando um dos modelos estudados de sincronização e de comunicação entre processos (*threads*): semáforos e memória partilhada, monitores ou passagem de mensagens.

A solução deve ser implementada em Linguagem C e ser passível de execução em Linux.

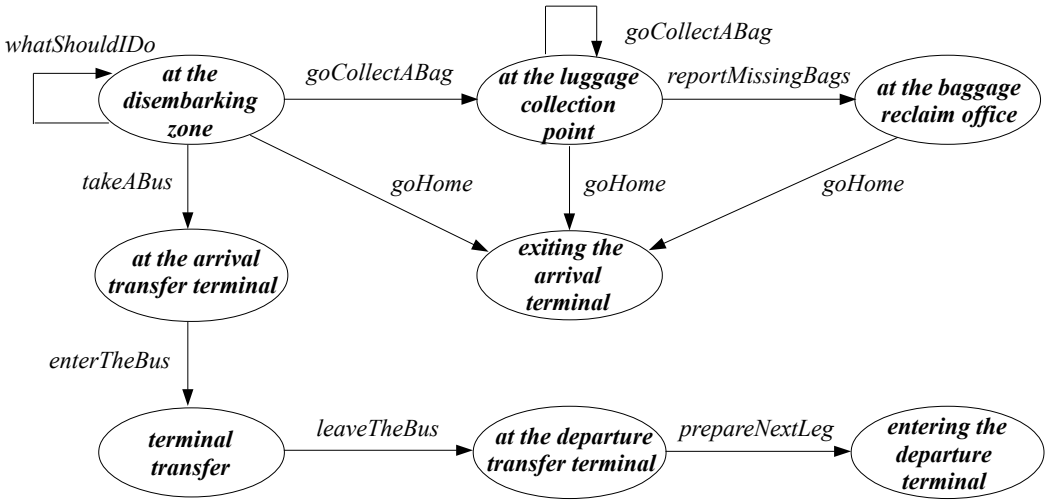
Incorpore um ficheiro de *logging* que descreva de um modo conciso, mas claro, a evolução do estado interno das diversas entidades envolvidas.

Sugestão de abordagem à solução

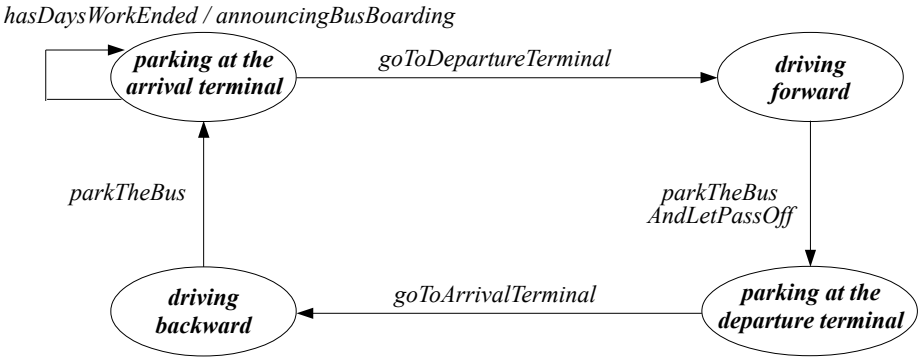
Ciclo de vida do bagageiro



Ciclo de vida do passageiro



Ciclo de vida do motorista



```
/* simulation parameters */

#define  K       5                              /* number of plane landings */
#define  N       6                          /* number of passengers per flight */
#define  M       2     /* maximum number of pieces of luggage per passenger */
#define  T       3                              /* number of seats in the bus */

/* porter process (thread) */

void main (void)
{
  unsigned int k;                                                /* flight number */
  BAG bag;                                 /* piece of luggage to be processed */

  for (k = 0; k < K; k++)
  { takeARest (k);                     /* the porter waits for a plane to land */
    while (tryToCollectABag (k, &bag))     /* the porter checks if there is
                                        still any luggage to collect at the
                                   plane's hold, if it is so he picks a piece */
      carryItToAppropriateStore (k, &bag);  /* the porter carries it to the
                         appropriate store (luggage conveyor belt / storeroom) */
    noMoreBagsToColect (k);        /* the porter goes back to the rest room */
  }
}

/* passengers processes (threads),   p = 0,1,...,N-1 */

void main (unsigned int p)
{
  unsigned int k;                                                /* flight number */
  unsigned int stat;                               /* status of the operation */

  for (k = 0; k < K; k++)
    switch (whatShouldIDo (k, p)) /* the passenger decides on her next move */
    { case FDBTC:           /* she has arrived to her final destination and
                                                          has bags to collect */
                    /* the passenger goes to the luggage collection point to
                                                        pick up her bags one by one */
        while ((stat = goCollectABag (k, p)) == NO);
        if (stat == MB)            /* the passenger checks for missing bags */
           reportMissingBags (k, p);     /* the passenger go to the baggage
                              reclaim office to fill the form for missing bags */
        goHome (k, p);                  /* the passenger leaves the airport */
        break;
      case FDNBTC:          /* she has arrived to her final destination and
                                                       has no bags to collect */
        goHome (k, p);                  /* the passenger leaves the airport */
        break;
      case INTRAN:                                      /* she is in transit */
        takeABus (k, p);      /* the passenger goes to the arrival transfer
               terminal to queue for taking a bus to the departure terminal */
        enterTheBus (k, p);      /* the passenger goes on board of the bus
                                                     as it starts its journey */
        leaveTheBus (k, p);           /* the passenger comes out of the bus
                                 as it reaches the departure transfer terminal */
        prepareNextLeg (k, p);        /* the passenger enters the departure
                         and does the check in for the next leg of the journey */
        break;
    }
}
```

```
/* bus driver process (thread) */

void main (void)
{
  while (!hasDaysWorkEnded ())
  { announcingBusBoarding ();             /* the driver invites the passengers
                forming the queue to board the bus up to it is packed full or
                                     there is at least one passenger waiting */
    goToDepartureTerminal ();                /* the driver takes the bus to
                                                    the departure terminal */
    parkTheBusAndLetPassOff ();          /* the driver parks the bus at the
                                       terminal and let the passengers leave */
    goToArrivalTerminal ();              /* the driver takes the bus back to
                                                     the arrival terminal */
    parkTheBus ();                       /* the driver parks at the terminal */
  }
}
```

Caracterização da interacção

Bagageiro

WAITING_FOR_A_PLANE_TO_LAND – estado de bloqueio (estado inicial / final)
o bagageiro é acordado pela operação *whatShouldIDo* do último dos passageiros a chegar ao local desembarque do avião
AT_THE_PLANES_HOLD – estado de transição
AT_THE_LUGGAGE_BELT_CONVEYOR –estado de transição
AT_THE_STOREROOM – estado de transição

Passageira

AT_THE_DISEMBARKING_ZONE – estado de transição (estado inicial)
AT_THE_LUGGAGE_COLLECTION_POINT – estado de bloqueio com transição eventual
a passageira é acordada pelas operações *carryItToAppropriateStore* e *tryToCollectABag* do bagageiro sempre que ele deposita no tapete rolante uma bagagem que lhe é destinada ou quando ele sinaliza que já não há mais bagagens no porão do avião e transita quando todas as bagagens próprias que pode recolher lhe foram disponibilizadas
AT_THE_BAGGAGE_RECLAIM_OFFICE – estado de transição
EXITING_THE_ARRIVAL_TERMINAL – estado de bloqueio com transição eventual (estado final)
a passageira é acordada pelas operações *goHome* ou *prepareNextLeg* da última passageira de cada voo a sair do terminal de chegada ou a entrar no terminal de partida
AT_THE_ARRIVAL_TRANSFER_TERMINAL – estado de bloqueio
antes de bloquear, a passageira acorda o motorista do autocarro se o seu lugar na fila de espera corresponder à lotação do autocarro e é acordada pela operação *announcingBus Boarding* do motorista para sinalizar a sua entrada no autocarro
TERMINAL_TRANSFER – estado de bloqueio
a passageira é acordada pela operação *parkTheBusAndLetPassOff* do motorista
AT_THE_DEPARTURE_TRANSFER_TERMINAL – estado de transição
ENTERING_THE_DEPARTURE_TERMINAL – estado de bloqueio com transição eventual (estado final)
a passageira é acordada pelas operações *goHome* ou *prepareNextLeg* da última passageira de cada voo a sair do terminal de chegada ou a entrar no terminal de partida

Motorista do autocarro

PARKING_AT_THE_ARRIVAL_TERMINAL – estado de duplo bloqueio (estado inicial / final)
o motorista é acordado do primeiro bloqueio pela operação *takeABus* de uma passageira que chega à zona de transferência de terminal de desembarque e verifica que o seu lugar na fila de espera corresponde à lotação do autocarro, ou quando o horário de partida foi atingido (só haverá transição entre pontos de bloqueio se pelo menos uma passageira estiver a aguardar viagem); o motorista é acordado do segundo bloqueio pela operação *enterTheBus* da última passageira a entrar no autocarro
DRIVING_FORWARD – estado de transição
PARKING_AT_THE_DEPARTURE_TERMINAL – estado de bloqueio
o motorista é acordado pela operação *leaveTheBus* da última passageira a sair do autocarro
DRIVING_BACKWARD – estado de transição

Comunicação e sincronização

Monitores

```
FULL_STAT fSt;                                  /* full state of the problem */
condition waitingFlight;   /* porter waiting for work synchronization point */
unsigned int nPassP;          /* number of passengers who have executed the
                              operation whatShouldIDo in each plane landing */
condition pass[N];  /* passengers synchronization point (one per passenger) */
unsigned int nCalls[N];         /* array of the number of calls made by the
                                      porter / bus driver to each passenger */
condition waitingDrive;        /* driver waiting for starting a new journey
                                                 synchronization point */
condition waitingPass;                    /* driver waiting for passengers
                                  to board / unboard synchronization point */
unsigned int nPassD;   /* number of passengers who have executed either the
                  operation enterTheBus or leaveTheBus in each bus transfer */
```

Semáforos e memória partilhada

```
shared FULL_STAT *pFSt;                         /* full state of the problem */
unsigned int access;                    /* identification of critical region
                                             protection semaphore - val = 1 */
unsigned int waitingFlight;              /* identification of porter waiting
                                             for work semaphore - val = 0 */
unsigned int nPassP;          /* number of passengers who have executed the
                              operation whatShouldIDo in each plane landing */
unsigned int pass[N];                       /* identification of passengers
                                  semaphores - val = 0 (one per passenger) */
unsigned int waitingDrive;           /* identification of driver waiting for
                                starting a new journey semaphore - val = 0 */
unsigned int waitingPass;            /* identification of driver waiting for
                                passengers to be ready semaphore - val = 0 */
unsigned int nPassD;   /* number of passengers who have executed either the
                  operation enterTheBus or leaveTheBus in each bus transfer */
```

Passagem de mensagens

```
message fSt;                              /* full state of the problem message */
message sync;                                  /* synchronization message */
mailbox access;                /* storage for the full state of the problem */
mailbox waitingFlight;                    /* storage for porter waiting for
                                                    work synchronization */
unsigned int nPassP;          /* number of passengers who have executed the
                              operation whatShouldIDo in each plane landing */
mailbox pass[N];                   /* storage for passenger synchronization
                                                   (one per passenger) */
mailbox waitingDrive;                     /* storage for driver waiting for
                                  starting a new journey synchronization */
mailbox waitingPass;                      /* storage for driver waiting for
                                  passengers to be ready synchronization */
unsigned int nPassD;   /* number of passengers who have executed either the
                  operation enterTheBus or leaveTheBus in each bus transfer */
```

```
/**
 *  Definition of state of the passenger data type.
 */

typedef struct
{ unsigned int stat;                                          /* internal state */
  unsigned int sit;   /* present situation (final destination / in transit) */
  unsigned int nBagsReal;    /* number of pieces of luggage she is supposed
                                                              to be carrying */
  unsigned int nBagsAct;       /* number of pieces of luggage she is really
                                                                  carrying */
} STAT_PASSENGER;
#define  FD       0  /* passenger has this airport as her final destination */
#define  TRT      1                                /* passenger is in transit */

/**
 *  Definition of state of the intervening entities data type.
 */

typedef struct
{ unsigned int porterStat;                              /* state of the porter */
  STAT_ PASSENGER passStat[K][N];                  /* state array of passengers */
  unsigned int driverStat;                              /* state of the driver */
} STAT;

/**
 *  Definition of piece of luggage data type.
 */

typedef struct
{ unsigned int id;                                   /* passenger identification */
} BAG;

/**
 *  Definition of plane load data type.
 */

typedef struct
{ unsigned int nBags;    /* number of pieces of luggage in the plane's hold */
  BAG bag[M*N];                                        /* plane's hold contents */
} LOAD;

/**
 *  Definition of cam of bags data type.
 */

typedef struct
{ BAG mem[M*N];                                             /* storage region */
  unsigned int n;                           /* number of bags presently stored */
} CAM;

/**
 *  Definition of queue of identification data type.
 */

typedef struct
{ unsigned int mem[N];                                      /* storage region */
  unsigned int ii;                                        /* insertion pointer */
  unsigned int ri;                                        /* retrieval pointer */
  bool full;                          /* flag signaling that the queue is full */
} QUEUE;
#define  EMPTYPOS    -1                              /* queue position is empty */
```

```
/**
 *  Definition of participant information for a transfer data type.
 */
typedef struct
{ int seat[T];               /* state of occupation of the seats in the bus
                                  (empty / identification of the passenger) */
  unsigned int nOccup;           /* number of seats presently occupied */
} TRANSF_INFO;
#define  EMPTYST     -1                                  /* seat is empty */

/**
 *  Definition of full state of the problem data type.
 */

typedef struct
{ unsigned int nLand;                              /* plane landing number */
  STAT st;                          /* state of the intervening entities */
  LOAD plHold[K];                /* array of manifests for the planes' hold */
  CAM convBelt;                                  /* luggage conveyor belt */
  QUEUE busQueue;                          /* queue for the transfer ride */
  TRANSF_INFO bus;                              /* bus occupation data */
  unsigned int nToTPassFD;       /* total number of passengers for whom the
                                        airport was their final destination */
  unsigned int nToTPassTST;       /* total number of passengers in transit */
  unsigned int nToTBagsPCB;             /* total number of bags placed in
                                                   the belt conveyor */
  unsigned int nToTBagsPSR;             /* total number of bags placed in
                                                        the storeroom */
  unsigned int nToTMBags;                 /* total number of missing bags */
  bool dayEnded;                           /* driver day's work has ended */
} FULL_STAT;
```