UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Dipartimento di Ingegneria “Enzo Ferrari”

Corso di Laurea Magistrale in Ingegneria Informatica (LM-32)

Curriculum “Cloud and Cybersecurity”

Elaborato di Progettazione di Sistemi Operativi

Docenti:

Prof.ssa Letizia Leonardi

Prof.ssa Silvia Cascianelli

Candidato:

Michele Mosca

Matricola 184249

ANNO ACCADEMICO 2022/2023

Indice

Elenco dei Codici

Introduzione

In questo elaborato verranno affrontati, descritti e discussi quattro problemi relativi alla programmazione concorrente mediante l'utilizzo della libreria *Pthread*. I quattro problemi sono tratti da testi di esami di anni precedenti, i cui testi originali sono presenti all'interno della sitografia. L'obiettivo dell'elaborato è quello di riuscire a costruire una soluzione per ognuno dei quattro problemi sincronizzando i processi concorrenti, proteggendo le strutture dati in comune e le varie sezioni critiche giungendo ad un risultato consistente. I quattro problemi selezionati in questo elaborato sono i seguenti:

- Nel **Capitolo 1** verrà analizzato il problema della **Banca**
- Nel **Capitolo 2** verrà analizzato il problema dell' **Elicottero**
- Nel **Capitolo 3** verrà analizzato il problema dell' **Officina**
- Nel **Capitolo 4** verrà analizzato il problema della **Vetrina Online**

Per ognuno di questi problemi verrà effettuata: una descrizione del problema, mostrando il testo dell'esercizio; una descrizione delle strutture, analizzando di quali variabili necessita la soluzione per la sua esecuzione; verrà fornita l'implementazione della soluzione, analizzando quali sono le procedure entry di cui la soluzione fa uso; verrà fornito un esempio di funzionamento, mostrando il relativo output; infine verrà discussa l'eventuale presenza di situazioni di Deadlock o Starvation all'interno della soluzione proposta.

1. Banca

1.1 Descrizione del Problema

In una **banca** è presente un'**area riservata** con **cassette di sicurezza**, e vi lavorano B **bancari**, addetti all'apertura delle cassette di sicurezza. Per motivi di sicurezza, nell'area riservata non possono essere presenti più di MAX clienti contemporaneamente (B $\ll$ MAX). Nell'area riservata si recano **clienti** che desiderano accedere alla propria cassetta di sicurezza[1]. Ogni cliente, per accedere alla propria cassetta di sicurezza, deve essere accompagnato da un bancario che apre la cassetta. Una volta aperta la cassetta (in un tempo random), il bancario lascia il cliente libero di visionare il contenuto (sempre in un tempo random). Quindi, il cliente lascia l'area riservata e torna a casa. I clienti possono essere clienti normali o clienti VIP. I clienti VIP hanno priorità nell'essere accompagnati dai bancari alle proprie cassette di sicurezza.

Si implementi una soluzione usando il costrutto *monitor* per modellare la banca, i *processi* per modellare i **clienti**, e i **bancari**, e si implementino **l'area riservata** come *risorsa*. Nella soluzione si massimizzi l'utilizzo delle risorse. Si discuta se la soluzione proposta può presentare starvation e in caso positivo per quali processi, e si propongano modifiche e/o aggiunte per evitare la starvation.

[1] Per semplicità non ci interessa associare i clienti a una specifica cassetta di sicurezza. Si supponga che se un cliente si reca in banca, sia presente una sua "generica" cassetta e che il bancario garantisca l'apertura della sua cassetta.

1.2 Descrizione delle strutture dati

1.2.1 Array di clienti

Per risolvere il problema, utilizzeremo un array di **Clienti** di tipo *int*, di dimensione pari al numero totale di clienti, dunque comprende sia i clienti VIP che i clienti normali, scelto per l'esecuzione del programma. Sul quale andremo a memorizzare l'*id* del thread *Bancario* che si occuperà di accompagnare il cliente nell'area riservata e successivamente aprire la sua cassetta di sicurezza.

```
1    int *clienti;
```

L'array di **Clienti** verrà inizializzato con il valore -1, indicando che il cliente non è ancora stato servito, mentre al termine dell'apertura della cassetta di sicurezza da parte di un thread *Bancario* avrà valore -2, indicando che tale cliente è stato servito. A tal proposito sono state realizzate due macro per rappresentare lo stato di ognuno dei clienti:

```
1    #define CLIENTE_NON_SERVITO -1
2    #define CLIENTE_SERVITO -2
```

1.2.2 Realizzazione delle code

Ogni cliente non appena si reca all'interno della banca, viene inserito in una delle due code messe a disposizione, in base alla tipologia di cliente che si presenta all'ingresso.

L'implementazione delle due code è stata realizzata mediante due array, **coda_clienti_vip** e **coda_clienti_normali**, di tipo *int* e di dimensioni pari al numero di utenti scelto per l'esecuzione del programma.

Inizializzati a -1, per indicare l'assenza di clienti all'interno della coda, e successivamente contenenti l'*id* del thread *Cliente* in attesa che venga accompagnato all'interno dell'area riservata e che gli venga aperta la cassetta di sicurezza. Verranno, inoltre, utilizzati due rispettivi contatori, **contatore_clienti_vip** e **contatore_clienti_normali**, inizializzati a 0, per indicare il numero di thread *Clienti* attualmente in coda.

```
1    int *coda_clienti_normali;
2    int contatore_clienti_normali;
3
```

```
4     int *coda_clienti_vip;
5     int contatore_clienti_vip;
```

1.2.3 Clienti all'interno dell'area riservata

Per limitare l'accesso dei thread *Clienti* all'interno dell'area riservata, è stato utilizzato un contatore di tipo *int* nominato **numero_clienti_area_riservata**. Il quale terrà traccia del numero dei *Clienti* che sono attualmente all'interno dell'area. Sarà compito dei thread *Bancari* controllare il valore di tale contatore prima di far accedere un thread *Cliente* all'interno dell'area e nel caso il contatore raggiunga il numero massimo di clienti consentiti, il thread *Bancario* dovrà attendere che un thread *Cliente* esca dall'area.

```
1     int numero_clienti_area_riservata;
```

1.2.4 Implementazione del costrutto monitor

La protezione di tali strutture dati, verrà effettuata dal costrutto *monitor* realizzato mediante l'utilizzo del seguente semaforo binario di mutua esclusione.

```
1     pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
```

1.2.5 Variabili condizione

Per la comunicazione tra i thread di tipo *Clienti* e i thread di tipo *Bancari*, utilizzeremo le seguenti variabili condizione:

```
1     pthread_cond_t clienti_in_coda = PTHREAD_COND_INITIALIZER;
2     pthread_cond_t attesa_ingresso = PTHREAD_COND_INITIALIZER;
3     pthread_cond_t attesa_apertura = PTHREAD_COND_INITIALIZER;
```

Dove **clienti_in_coda** rappresenta la variabile condizione in cui i thread *Bancari* si bloccano in attesa che un cliente si inserisca in una delle due code, pronto per poter essere servito da un bancario. Mentre la variabile condizione **attesa_ingresso** serve ai thread *Clienti* per bloccarsi in attesa che un thread *Bancario* lo selezioni per accompagnarlo all'interno dell'area riservata. Infine, **attesa_apertura** è la variabile condizione che serve ai thread *Clienti*, una volta che un thread

Bancario lo ha selezionato per accompagnarlo all'interno dell'area riservata, per bloccarsi in attesa che il bancario abbia terminato l'apertura della cassetta di sicurezza.

1.3 Implementazione della soluzione

1.3.1 Schema del thread Clienti

Lo schema del thread *Cliente*, analogo sia per clienti VIP che clienti normali, è composto dalla chiamata della procedure entry **ENTRA_BANCA**, seguita dalla generazione random della durata della visone del contenuto della cassetta di sicurezza con consecutiva visione del contenuto ed infine dalla chiamata della procedure entry **ESCI_BANCA**.

```
1     /* entro all'interno della banca */
2     ENTRA_BANCA(*pi, CLIENTE_VIP);
3
4     /* genero la durata della visione del contenuto della cassetta di
          sicurezza */
5     durata = mia_random(MAX_DURATA_VISIONE_CASSETTA);
6
7     printf("CLIENTE_VIP-[Thread%d e identificatore %lu] inizio la visione
          della cassetta di sicurezza (durata: %d secondi)\n", *pi, pthread_self
          (), durata);
8
9     /* simulo l'azione di visione del contenuto della cassetta di sicurezza
          mediante una sleep */
10    sleep(durata);
11
12    /* notifico di aver appena terminato la visione del contenuto della
          cassetta di sicurezza */
13    ESCI_BANCA(*pi, CLIENTE_VIP);
```

All'interno della procedure entry **ENTRA_BANCA**, realizzata mediante l'utilizzo del semaforo di mutua esclusione **mutex**, il cliente inserisce il proprio *id* all'interno della rispettiva coda di clienti, in base alla tipologia di cliente a cui appartiene. A tal proposito sono state definite le seguenti macro per la rappresentazione della tipologia di cliente:

```
#define CLIENTE_NORMALE 0
#define CLIENTE_VIP 1
```

Una volta inserito il proprio *id* all'interno dell'apposita coda dei clienti, il cliente effettua una *signal* in *broadcast* sulla variabile condizione **clienti_in_coda**, risvegliando tutti i thread *Bancari* in attesa.

Successivamente, viene verificata la condizione di selezione da parte di un thread *Bancario* e in caso contrario si sospende effettuando una *wait* sulla variabile condizione **attesa_ingresso**, in attesa che un bancario lo selezioni per farlo entrare all'interno dell'area riservata.

Risvegliato da parte di un thread *Bancario*, il cliente aumenta di uno il **numero_clienti_area_riservata**, indicando di essere appena stato accompagnato all'interno dell'area riservata e stampa in output che un bancario, specificando anche l'*id*, lo ha appena fatto entrare. Dopodiché si sospende in attesa che il bancario termini l'apertura della sua cassetta di sicurezza, mediante una *wait* sulla variabile condizione **attesa_apertura**.

Risvegliato nuovamente da parte del thread *Bancario*, il cliente adesso può visionare la sua cassetta di sicurezza, dunque termina la procedure entry per eseguire i passi successivi.

```
void ENTRA_BANCA(int id, int tipo_cliente)
{
    pthread_mutex_lock(&mutex);

    if (tipo_cliente == CLIENTE_VIP)
    {
        printf("CLIENTE_VIP-[Thread%d e identificatore %lu] sono appena
            entrato all'interno della banca, mi metto nella coda dei
            clienti VIP\n", id, pthread_self());

        /* mi metto nella coda dei clienti VIP */
        coda_clienti_vip[contatore_clienti_vip] = id;
        contatore_clienti_vip++;
    }
    else
    {
        printf("CLIENTE_NORMALE-[Thread%d e identificatore %lu] sono
            appena entrato all'interno della banca, mi metto nella coda dei
```

```
                clienti normali\n", id, pthread_self());
16
17              /* mi metto nella coda dei clienti normali */
18              coda_clienti_normali[contatore_clienti_normali] = id;
19              contatore_clienti_normali++;
20          }
21
22          /* sveglio i bancari in attesa per poter essere servito */
23          pthread_cond_broadcast(&clienti_in_coda);
24
25          while(clienti[id - NUM_THREADS_BANCARI] == CLIENTE_NON_SERVITO)
26          {
27              /* mi metto in attesa che un bancario mi faccia accedere all'area
                    riservata per potermi aprire la cassetta di sicurezza */
28              pthread_cond_wait(&attesa_ingresso, &mutex);
29          }
30
31          /* aumento di uno il numero di utenti all'interno dell'area riservata
                */
32          numero_clienti_area_riservata++;
33
34          if (tipo_cliente == CLIENTE_VIP)
35              printf("CLIENTE_VIP-[Thread%d e identificatore %lu] il bancario
                    con id %d mi ha appena accompagnato nell'area riservata,
                    attendo che apra la cassetta di sicurezza (clienti all'interno
                    dell'area riservata: %d)\n", id, pthread_self(), clienti[id -
                    NUM_THREADS_BANCARI], numero_clienti_area_riservata);
36          else
37              printf("CLIENTE_NORMALE-[Thread%d e identificatore %lu] il
                    bancario con id %d mi ha appena accompagnato nell'area
                    riservata, attendo che apra la cassetta di sicurezza (clienti
                    all'interno dell'area riservata: %d)\n", id, pthread_self(),
                    clienti[id - NUM_THREADS_BANCARI],
                    numero_clienti_area_riservata);
38
39          while(clienti[id - NUM_THREADS_BANCARI] != CLIENTE_SERVITO)
```

```
40          {
41              /* mi metto in attesa che il bancario termini l'apertura della
                    cassetta di sicurezza */
42              pthread_cond_wait(&attesa_apertura, &mutex);
43          }
44
45          pthread_mutex_unlock(&mutex);
46      }
```

Codice 1.1: Procedure Entry ENTRA_BANCA del problema della Banca

All'interno della procedure entry **ESCI_BANCA**, realizzata mediante l'utilizzo del semaforo di mutua esclusione **mutex**, il cliente una volta completata la visione del contenuto della cassetta di sicurezza, lo notifica stampando un messaggio significativo su standard output. Dopodiché diminuisce di uno il numero di clienti all'interno dell'area riservata e notifica i bancari, mediante l'utilizzo di una *signal* in *broadcast* sulla variabile condizione **clienti_in_coda**, di aver liberato un posto all'interno dell'area. Terminando così la procedure entry **ESCI_BANCA** e la sua esecuzione.

1.3.2 Scheda del thread Bancari

Lo schema del thread *Bancario* è composto da un ciclo infinito in cui il corriere ripete le seguenti operazioni:

- **INIZIO_LAVORO**, procedure entry in cui il bancario si mette a disposizione dei clienti per accompagnarli all'interno dell'area riservata
- simulazione dell'apertura della cassetta di sicurezza mediante una sleep
- **FINE_LAVORO**, procedure entry un cui il bancario notifica il cliente di aver terminato l'apertura della cassetta di sicurezza

```
1       while(1)
2       {
3           /* mi metto a disposizione per l'apertura della cassetta di sicurezza
                di un utente */
4           INIZIO_LAVORO(*pi, &id_cliente, &durata);
```

```
5
6           /* simulo l'azione di apertura della cassetta di sicurezza mediante
                una sleep */
7           sleep(durata);
8
9           /* notifico all'utente che la sua cassetta di sicurezza e' pronta per
                essere visionata */
10          FINE_LAVORO(*pi, id_cliente);
11      }
```

All'interno della procedure entry **INIZIO_LAVORO**, realizzata mediante l'utilizzo del semaforo di mutua esclusione **mutex**, il bancario si sospende, mediante una *wait* sulla variabile condizione **clienti_in_coda**, in attesa che un thread *Cliente* si inserisca in una delle due code e che ci siano abbastanza posti disponibili all'interno dell'area riservata.

Risvegliato da parte di un thread *Cliente*, controlla prima se ci sono clienti VIP in coda ed in caso contrario controlla la coda dei clienti normali. Successivamente inserisce il proprio *id* all'interno dell'array **clienti** con indice l'*id* del thread *Cliente*. Rimuove il cliente dall'array dei clienti in coda e calcola la durata del tempo impiegato per l'apertura della cassetta di sicurezza. Infine notifica il cliente di averlo selezionato per entrare all'interno dell'area riservata, mediante una *signal* in *broadcast* sulla variabile condizione **attesa_ingresso**. Terminando così la procedure entry **INIZIO_LAVORO**.

```
1       void INIZIO_LAVORO(int id, int *id_cliente, int *durata)
2       {
3           pthread_mutex_lock(&mutex);
4
5           int i;  /* variabile contatore utilizzata per scorrere le code dei
                clienti */
6
7           /* attendo che ci sia un cliente in coda e che non siano esauriti i
                posti all'interno dell'area riservata */
8           while((contatore_clienti_vip == 0 && contatore_clienti_normali == 0)
                || numero_clienti_area_riservata >= MAX_CLIENTI_CONTEMPORANEAMENTE)
9           {
10              if (contatore_clienti_vip == 0 && contatore_clienti_normali == 0)
```

```
11                  printf("BANCARIO-[Thread%d e identificatore %lu] nessun
                        cliente e' in coda, mi sospendo\n", id, pthread_self());
12              else
13                  printf("BANCARIO-[Thread%d e identificatore %lu] ci sono dei
                        clienti in coda ma il numero di persone all'interno dell'
                        area riservato ha raggiunto il numero massimo, mi sospendo\
                        n", id, pthread_self());
14
15              pthread_cond_wait(&clienti_in_coda, &mutex);
16          }
17
18          /* se ci sono clienti VIP in coda do la precedenza a loro */
19          if (contatore_clienti_vip > 0)
20          {
21              printf("BANCARIO-[Thread%d e identificatore %lu] cliente VIP in
                    coda, lo rimuovo dalla coda\n", id, pthread_self());
22
23              /* seleziono il primo cliente VIP che e' entrato nella coda */
24              *id_cliente = coda_clienti_vip[0];
25              clienti[*id_cliente - NUM_THREADS_BANCARI] = id;
26
27              /* rimuovo dalla coda il cliente VIP selezionato */
28              for (i = 0; i < contatore_clienti_vip - 1; i++)
29                  coda_clienti_vip[i] = coda_clienti_vip[i+1];
30
31              coda_clienti_vip[contatore_clienti_vip -1] = -1;
32              contatore_clienti_vip--;
33
34              /* calcolo la durata del tempo impiegato per l'apertura della
                    cassetta di sicurezza */
35              *durata = mia_random(MAX_DURATA_APERTURA_CASSETTA);
36
37              printf("BANCARIO-[Thread%d e identificatore %lu] faccio entrare il
                     cliente VIP con id %d all'interno dell'area riservata e apro
                    la sua cassetta di sicurezza (tempo di apertura: %d secondi)\n"
                    , id, pthread_self(), *id_cliente, *durata);
```

```
38          }
39          else    /* in coda ci sono solo clienti normali */
40          {
41              printf("BANCARIO-[Thread%d e identificatore %lu] cliente normale
                    in coda, lo rimuovo dalla coda\n", id, pthread_self());
42
43              /* seleziono il primo cliente normale che e' entrato nella coda */
44              *id_cliente = coda_clienti_normali[0];
45              clienti[*id_cliente - NUM_THREADS_BANCARI] = id;
46
47              /* rimuovo dalla coda il cliente normale selezionato */
48              for (i = 0; i < contatore_clienti_normali - 1; i++)
49                  coda_clienti_normali[i] = coda_clienti_normali[i+1];
50
51              coda_clienti_normali[contatore_clienti_normali -1] = -1;
52              contatore_clienti_normali--;
53
54              /* calcolo la durata del tempo impiegato per l'apertura della
                    cassetta di sicurezza */
55              *durata = mia_random(MAX_DURATA_APERTURA_CASSETTA);
56
57              printf("BANCARIO-[Thread%d e identificatore %lu] faccio entrare il
                     cliente normale con id %d all'interno dell'area riservata e
                    apro la sua cassetta di sicurezza (tempo di apertura: %d
                    secondi)\n", id, pthread_self(), *id_cliente, *durata);
58          }
59
60          /* notifico il cliente di averlo selezionato per entrare all'interno
                dell'area riservata */
61          pthread_cond_broadcast(&attesa_ingresso);
62
63          pthread_mutex_unlock(&mutex);
64      }
```

Codice 1.2: Procedure Entry INIZIO_LAVORO del problema della Banca

All'interno della procedure entry **FINE_LAVORO**, realizzata mediante l'utilizzo del semaforo

di mutua esclusione **mutex**, il bancario modifica lo stato del cliente, all'interno dell'array **clienti**, in *CLIENTE_SERVITO*. Dopodiché notifica l'utente di aver appena aperto la sua cassetta di sicurezza, mediante l'uso di una *signal* in *broadcast* sulla variabile condizione **attesa_apertura** e termina la procedure entry, lasciando l'utente libero di visionarne il contenuto.

```
1   void FINE_LAVORO(int id, int id_cliente)
2   {
3       pthread_mutex_lock(&mutex);
4
5       printf("BANCARIO-[Thread%d e identificatore %lu] apertura della
            cassetta di sicurezza del cliente avente id %d completata\n",id,
            pthread_self(), id_cliente);
6
7       /* modifico lo stato del cliente in CLIENTE_SERVITO (-2) */
8       clienti[id_cliente - NUM_THREADS_BANCARI] = CLIENTE_SERVITO;
9
10      /* notifico il cliente di aver terminato l'apertura della sua cassetta
            di sicurezza, pertanto puo' procedere a visionarla */
11      pthread_cond_broadcast(&attesa_apertura);
12
13      pthread_mutex_unlock(&mutex);
14  }
```

Codice 1.3: Procedure Entry FINE_LAVORO del problema della Banca

1.4 Esempio di Funzionamento

L'esecuzione del programma avviene mediante l'utilizzo di quattro argomenti, rispettivamente sono:

1. *NUM_BANCARI*
2. *MAX_CLIENTI_CONTEMPORANEAMENTE*
3. *NUM_CLIENTI_NORMALI*
4. *NUM_CLIENTI_VIP*

Per implementare il vincolo che il numero massimo di clienti all'interno dell'area riservata sia molto maggiore del numero dei bancari (B $\ll$ MAX), ho imposto che il numero dei clienti **MAX_CLIENTI_CONTEMPORANEAMENTE** debba essere almeno il doppio di **NUM_BANCARI** per la corretta esecuzione del programma.

Una tipica invocazione del programma è la seguente:

```
$ ./banca 2 4 6 4
```

Codice 1.4: Esempio di invocazione del programma del problema della Banca

In questo caso avremo due bancari all'interno della banca, il numero dei clienti contemporaneamente ammessi all'interno dell'area riservata sarà di quattro clienti e avremo un numero totale di clienti pari a dieci, suddiviso in sei clienti normali e quattro clienti VIP.

```
Numero totale di BANCARI: 2
Numero MAX_CLIENTI_CONTEMPORANEAMENTE: 4
Numero CLIENTI_NORMALI: 6
Numero CLIENTI_VIP: 4
Numero totale di CLIENTI: 10
Sto per creare il thread BANCARIO 0-esimo
SONO IL MAIN e ho creato il Pthread BANCARIO 0-esimo con id =140349876512512
Sto per creare il thread BANCARIO 1-esimo
BANCARIO-[Thread0 e identificatore 140349876512512] STO ARRIVANDO
BANCARIO-[Thread0 e identificatore 140349876512512] nessun cliente e' in coda, mi sospendo
SONO IL MAIN e ho creato il Pthread BANCARIO 1-esimo con id =140349800445696
BANCARIO-[Thread1 e identificatore 140349800445696] STO ARRIVANDO
BANCARIO-[Thread1 e identificatore 140349800445696] nessun cliente e' in coda, mi sospendo
Sto per creare il thread CLIENTE_VIP 2-esimo
SONO IL MAIN e ho creato il Pthread CLIENTE_VIP 2-esimo con id =140349792052992
Sto per creare il thread CLIENTE_NORMALE 3-esimo
SONO IL MAIN e ho creato il Pthread CLIENTE_NORMALE 3-esimo con id =140349783660288
Sto per creare il thread CLIENTE_VIP 4-esimo
```

```
19  CLIENTE_VIP-[Thread2 e identificatore 140349792052992] STO ARRIVANDO
20  SONO IL MAIN e ho creato il Pthread CLIENTE_VIP 4-esimo con id
        =140349775267584
21  Sto per creare il thread CLIENTE_VIP 5-esimo
22  CLIENTE_VIP-[Thread2 e identificatore 140349792052992] sono appena entrato
        all'interno della banca, mi metto nella coda dei clienti VIP
23  SONO IL MAIN e ho creato il Pthread CLIENTE_VIP 5-esimo con id
        =140349766874880
24  Sto per creare il thread CLIENTE_VIP 6-esimo
25  CLIENTE_NORMALE-[Thread3 e identificatore 140349783660288] STO ARRIVANDO
26  CLIENTE_NORMALE-[Thread3 e identificatore 140349783660288] sono appena
        entrato all'interno della banca, mi metto nella coda dei clienti
        normali
27  CLIENTE_VIP-[Thread6 e identificatore 140349758482176] STO ARRIVANDO
28  SONO IL MAIN e ho creato il Pthread CLIENTE_VIP 6-esimo con id
        =140349758482176
29  Sto per creare il thread CLIENTE_NORMALE 7-esimo
30  CLIENTE_VIP-[Thread5 e identificatore 140349766874880] STO ARRIVANDO
31  CLIENTE_VIP-[Thread5 e identificatore 140349766874880] sono appena entrato
        all'interno della banca, mi metto nella coda dei clienti VIP
32  CLIENTE_VIP-[Thread4 e identificatore 140349775267584] STO ARRIVANDO
33  CLIENTE_VIP-[Thread4 e identificatore 140349775267584] sono appena entrato
        all'interno della banca, mi metto nella coda dei clienti VIP
34  BANCARIO-[Thread0 e identificatore 140349876512512] cliente VIP in coda,
        lo rimuovo dalla coda
35  BANCARIO-[Thread0 e identificatore 140349876512512] faccio entrare il
        cliente VIP con id 2 all'interno dell'area riservata e apro la sua
        cassetta di sicurezza (tempo di apertura: 4 secondi)
36  CLIENTE_VIP-[Thread2 e identificatore 140349792052992] il bancario con id
        0 mi ha appena accompagnato nell'area riservata, attendo che apra la
        cassetta di sicurezza (clienti all'interno dell'area riservata: 1)
37  SONO IL MAIN e ho creato il Pthread CLIENTE_NORMALE 7-esimo con id
        =140349750089472
38  Sto per creare il thread CLIENTE_NORMALE 8-esimo
39  BANCARIO-[Thread1 e identificatore 140349800445696] cliente VIP in coda,
        lo rimuovo dalla coda
```

```
40  BANCARIO-[Thread1 e identificatore 140349800445696] faccio entrare il
        cliente VIP con id 5 all'interno dell'area riservata e apro la sua
        cassetta di sicurezza (tempo di apertura: 3 secondi)
41  CLIENTE_NORMALE-[Thread7 e identificatore 140349750089472] STO ARRIVANDO
42  CLIENTE_NORMALE-[Thread7 e identificatore 140349750089472] sono appena
        entrato all'interno della banca, mi metto nella coda dei clienti
        normali
43  CLIENTE_VIP-[Thread5 e identificatore 140349766874880] il bancario con id
        1 mi ha appena accompagnato nell'area riservata, attendo che apra la
        cassetta di sicurezza (clienti all'interno dell'area riservata: 2)
44  SONO IL MAIN e ho creato il Pthread CLIENTE_NORMALE 8-esimo con id
        =140349397792512
45  Sto per creare il thread CLIENTE_NORMALE 9-esimo
46  SONO IL MAIN e ho creato il Pthread CLIENTE_NORMALE 9-esimo con id
        =140349389399808
47  Sto per creare il thread CLIENTE_NORMALE 10-esimo
48  CLIENTE_NORMALE-[Thread8 e identificatore 140349397792512] STO ARRIVANDO
49  CLIENTE_NORMALE-[Thread8 e identificatore 140349397792512] sono appena
        entrato all'interno della banca, mi metto nella coda dei clienti
        normali
50  SONO IL MAIN e ho creato il Pthread CLIENTE_NORMALE 10-esimo con id
        =140349381007104
51  Sto per creare il thread CLIENTE_NORMALE 11-esimo
52  CLIENTE_NORMALE-[Thread9 e identificatore 140349389399808] STO ARRIVANDO
53  CLIENTE_VIP-[Thread6 e identificatore 140349758482176] sono appena entrato
         all'interno della banca, mi metto nella coda dei clienti VIP
54  CLIENTE_NORMALE-[Thread10 e identificatore 140349381007104] STO ARRIVANDO
55  CLIENTE_NORMALE-[Thread10 e identificatore 140349381007104] sono appena
        entrato all'interno della banca, mi metto nella coda dei clienti
        normali
56  CLIENTE_NORMALE-[Thread9 e identificatore 140349389399808] sono appena
        entrato all'interno della banca, mi metto nella coda dei clienti
        normali
57  SONO IL MAIN e ho creato il Pthread CLIENTE_NORMALE 11-esimo con id
        =140349372614400
58  CLIENTE_NORMALE-[Thread11 e identificatore 140349372614400] STO ARRIVANDO
```

```
59 CLIENTE_NORMALE-[Thread11 e identificatore 140349372614400] sono appena
       entrato all'interno della banca, mi metto nella coda dei clienti
       normali
60 BANCARIO-[Thread1 e identificatore 140349800445696] apertura della
       cassetta di sicurezza del cliente avente id 5 completata
61 BANCARIO-[Thread1 e identificatore 140349800445696] cliente VIP in coda,
       lo rimuovo dalla coda
62 BANCARIO-[Thread1 e identificatore 140349800445696] faccio entrare il
       cliente VIP con id 4 all'interno dell'area riservata e apro la sua
       cassetta di sicurezza (tempo di apertura: 1 secondi)
63 CLIENTE_VIP-[Thread5 e identificatore 140349766874880] inizio la visione
       della cassetta di sicurezza (durata: 2 secondi)
64 CLIENTE_VIP-[Thread4 e identificatore 140349775267584] il bancario con id
       1 mi ha appena accompagnato nell'area riservata, attendo che apra la
       cassetta di sicurezza (clienti all'interno dell'area riservata: 3)
65 BANCARIO-[Thread0 e identificatore 140349876512512] apertura della
       cassetta di sicurezza del cliente avente id 2 completata
66 BANCARIO-[Thread0 e identificatore 140349876512512] cliente VIP in coda,
       lo rimuovo dalla coda
67 BANCARIO-[Thread0 e identificatore 140349876512512] faccio entrare il
       cliente VIP con id 6 all'interno dell'area riservata e apro la sua
       cassetta di sicurezza (tempo di apertura: 2 secondi)
68 CLIENTE_VIP-[Thread2 e identificatore 140349792052992] inizio la visione
       della cassetta di sicurezza (durata: 3 secondi)
69 CLIENTE_VIP-[Thread6 e identificatore 140349758482176] il bancario con id
       0 mi ha appena accompagnato nell'area riservata, attendo che apra la
       cassetta di sicurezza (clienti all'interno dell'area riservata: 4)
70 BANCARIO-[Thread1 e identificatore 140349800445696] apertura della
       cassetta di sicurezza del cliente avente id 4 completata
71 BANCARIO-[Thread1 e identificatore 140349800445696] ci sono dei clienti in
        coda ma il numero di persone all'interno dell'area riservato ha
       raggiunto il numero massimo, mi sospendo
72 CLIENTE_VIP-[Thread4 e identificatore 140349775267584] inizio la visione
       della cassetta di sicurezza (durata: 4 secondi)
73 CLIENTE_VIP-[Thread5 e identificatore 140349766874880] ho appena terminato
        di visionare la cassetta di sicurezza, ESCO dall'area riservata (
```

```
    clienti all'interno dell'area riservata: 3)
74  CLIENTE_VIP-[Thread5 e identificatore 140349766874880] VADO A CASA
75  BANCARIO-[Thread1 e identificatore 140349800445696] cliente normale in
    coda, lo rimuovo dalla coda
76  BANCARIO-[Thread1 e identificatore 140349800445696] faccio entrare il
    cliente normale con id 3 all'interno dell'area riservata e apro la sua
    cassetta di sicurezza (tempo di apertura: 3 secondi)
77  CLIENTE_NORMALE-[Thread3 e identificatore 140349783660288] il bancario con
     id 1 mi ha appena accompagnato nell'area riservata, attendo che apra
    la cassetta di sicurezza (clienti all'interno dell'area riservata: 4)
78  BANCARIO-[Thread0 e identificatore 140349876512512] apertura della
    cassetta di sicurezza del cliente avente id 6 completata
79  BANCARIO-[Thread0 e identificatore 140349876512512] ci sono dei clienti in
     coda ma il numero di persone all'interno dell'area riservato ha
    raggiunto il numero massimo, mi sospendo
80  CLIENTE_VIP-[Thread6 e identificatore 140349758482176] inizio la visione
    della cassetta di sicurezza (durata: 4 secondi)
81  CLIENTE_VIP-[Thread2 e identificatore 140349792052992] ho appena terminato
     di visionare la cassetta di sicurezza, ESCO dall'area riservata (
    clienti all'interno dell'area riservata: 3)
82  CLIENTE_VIP-[Thread2 e identificatore 140349792052992] VADO A CASA
83  BANCARIO-[Thread0 e identificatore 140349876512512] cliente normale in
    coda, lo rimuovo dalla coda
84  BANCARIO-[Thread0 e identificatore 140349876512512] faccio entrare il
    cliente normale con id 7 all'interno dell'area riservata e apro la sua
    cassetta di sicurezza (tempo di apertura: 4 secondi)
85  Pthread 2-esimo restituisce 2
86  CLIENTE_NORMALE-[Thread7 e identificatore 140349750089472] il bancario con
     id 0 mi ha appena accompagnato nell'area riservata, attendo che apra
    la cassetta di sicurezza (clienti all'interno dell'area riservata: 4)
87  CLIENTE_VIP-[Thread4 e identificatore 140349775267584] ho appena terminato
     di visionare la cassetta di sicurezza, ESCO dall'area riservata (
    clienti all'interno dell'area riservata: 3)
88  CLIENTE_VIP-[Thread4 e identificatore 140349775267584] VADO A CASA
89  BANCARIO-[Thread1 e identificatore 140349800445696] apertura della
    cassetta di sicurezza del cliente avente id 3 completata
```

```
90  BANCARIO-[Thread1 e identificatore 140349800445696] cliente normale in
        coda, lo rimuovo dalla coda
91  BANCARIO-[Thread1 e identificatore 140349800445696] faccio entrare il
        cliente normale con id 8 all'interno dell'area riservata e apro la sua
        cassetta di sicurezza (tempo di apertura: 3 secondi)
92  CLIENTE_NORMALE-[Thread3 e identificatore 140349783660288] inizio la
        visione della cassetta di sicurezza (durata: 5 secondi)
93  CLIENTE_NORMALE-[Thread8 e identificatore 140349397792512] il bancario con
         id 1 mi ha appena accompagnato nell'area riservata, attendo che apra
        la cassetta di sicurezza (clienti all'interno dell'area riservata: 4)
94  CLIENTE_VIP-[Thread6 e identificatore 140349758482176] ho appena terminato
         di visionare la cassetta di sicurezza, ESCO dall'area riservata (
        clienti all'interno dell'area riservata: 3)
95  CLIENTE_VIP-[Thread6 e identificatore 140349758482176] VADO A CASA
96  BANCARIO-[Thread0 e identificatore 140349876512512] apertura della
        cassetta di sicurezza del cliente avente id 7 completata
97  BANCARIO-[Thread0 e identificatore 140349876512512] cliente normale in
        coda, lo rimuovo dalla coda
98  BANCARIO-[Thread0 e identificatore 140349876512512] faccio entrare il
        cliente normale con id 10 all'interno dell'area riservata e apro la sua
         cassetta di sicurezza (tempo di apertura: 3 secondi)
99  CLIENTE_NORMALE-[Thread10 e identificatore 140349381007104] il bancario
        con id 0 mi ha appena accompagnato nell'area riservata, attendo che
        apra la cassetta di sicurezza (clienti all'interno dell'area riservata:
         4)
100 CLIENTE_NORMALE-[Thread7 e identificatore 140349750089472] inizio la
        visione della cassetta di sicurezza (durata: 5 secondi)
101 BANCARIO-[Thread1 e identificatore 140349800445696] apertura della
        cassetta di sicurezza del cliente avente id 8 completata
102 BANCARIO-[Thread1 e identificatore 140349800445696] ci sono dei clienti in
         coda ma il numero di persone all'interno dell'area riservato ha
        raggiunto il numero massimo, mi sospendo
103 CLIENTE_NORMALE-[Thread8 e identificatore 140349397792512] inizio la
        visione della cassetta di sicurezza (durata: 1 secondi)
104 CLIENTE_NORMALE-[Thread8 e identificatore 140349397792512] ho appena
        terminato di visionare la cassetta di sicurezza, ESCO dall'area
```

```
    riservata (clienti all'interno dell'area riservata: 3)
105 CLIENTE_NORMALE-[Thread8 e identificatore 140349397792512] VADO A CASA
106 BANCARIO-[Thread1 e identificatore 140349800445696] cliente normale in
    coda, lo rimuovo dalla coda
107 BANCARIO-[Thread1 e identificatore 140349800445696] faccio entrare il
    cliente normale con id 9 all'interno dell'area riservata e apro la sua
    cassetta di sicurezza (tempo di apertura: 4 secondi)
108 CLIENTE_NORMALE-[Thread9 e identificatore 140349389399808] il bancario con
     id 1 mi ha appena accompagnato nell'area riservata, attendo che apra
    la cassetta di sicurezza (clienti all'interno dell'area riservata: 4)
109 CLIENTE_NORMALE-[Thread3 e identificatore 140349783660288] ho appena
    terminato di visionare la cassetta di sicurezza, ESCO dall'area
    riservata (clienti all'interno dell'area riservata: 3)
110 CLIENTE_NORMALE-[Thread3 e identificatore 140349783660288] VADO A CASA
111 Pthread 3-esimo restituisce 3
112 Pthread 4-esimo restituisce 4
113 Pthread 5-esimo restituisce 5
114 Pthread 6-esimo restituisce 6
115 BANCARIO-[Thread0 e identificatore 140349876512512] apertura della
    cassetta di sicurezza del cliente avente id 10 completata
116 BANCARIO-[Thread0 e identificatore 140349876512512] cliente normale in
    coda, lo rimuovo dalla coda
117 BANCARIO-[Thread0 e identificatore 140349876512512] faccio entrare il
    cliente normale con id 11 all'interno dell'area riservata e apro la sua
     cassetta di sicurezza (tempo di apertura: 1 secondi)
118 CLIENTE_NORMALE-[Thread11 e identificatore 140349372614400] il bancario
    con id 0 mi ha appena accompagnato nell'area riservata, attendo che
    apra la cassetta di sicurezza (clienti all'interno dell'area riservata:
     4)
119 CLIENTE_NORMALE-[Thread10 e identificatore 140349381007104] inizio la
    visione della cassetta di sicurezza (durata: 8 secondi)
120 BANCARIO-[Thread0 e identificatore 140349876512512] apertura della
    cassetta di sicurezza del cliente avente id 11 completata
121 BANCARIO-[Thread0 e identificatore 140349876512512] nessun cliente e' in
    coda, mi sospendo
122 CLIENTE_NORMALE-[Thread11 e identificatore 140349372614400] inizio la
```

```
    visione della cassetta di sicurezza (durata: 6 secondi)
123 CLIENTE_NORMALE-[Thread7 e identificatore 140349750089472] ho appena
    terminato di visionare la cassetta di sicurezza, ESCO dall'area
    riservata (clienti all'interno dell'area riservata: 3)
124 CLIENTE_NORMALE-[Thread7 e identificatore 140349750089472] VADO A CASA
125 BANCARIO-[Thread0 e identificatore 140349876512512] nessun cliente e' in
    coda, mi sospendo
126 Pthread 7-esimo restituisce 7
127 Pthread 8-esimo restituisce 8
128 BANCARIO-[Thread1 e identificatore 140349800445696] apertura della
    cassetta di sicurezza del cliente avente id 9 completata
129 BANCARIO-[Thread1 e identificatore 140349800445696] nessun cliente e' in
    coda, mi sospendo
130 CLIENTE_NORMALE-[Thread9 e identificatore 140349389399808] inizio la
    visione della cassetta di sicurezza (durata: 7 secondi)
131 CLIENTE_NORMALE-[Thread11 e identificatore 140349372614400] ho appena
    terminato di visionare la cassetta di sicurezza, ESCO dall'area
    riservata (clienti all'interno dell'area riservata: 2)
132 CLIENTE_NORMALE-[Thread11 e identificatore 140349372614400] VADO A CASA
133 BANCARIO-[Thread1 e identificatore 140349800445696] nessun cliente e' in
    coda, mi sospendo
134 BANCARIO-[Thread0 e identificatore 140349876512512] nessun cliente e' in
    coda, mi sospendo
135 CLIENTE_NORMALE-[Thread10 e identificatore 140349381007104] ho appena
    terminato di visionare la cassetta di sicurezza, ESCO dall'area
    riservata (clienti all'interno dell'area riservata: 1)
136 CLIENTE_NORMALE-[Thread10 e identificatore 140349381007104] VADO A CASA
137 BANCARIO-[Thread1 e identificatore 140349800445696] nessun cliente e' in
    coda, mi sospendo
138 BANCARIO-[Thread0 e identificatore 140349876512512] nessun cliente e' in
    coda, mi sospendo
139 CLIENTE_NORMALE-[Thread9 e identificatore 140349389399808] ho appena
    terminato di visionare la cassetta di sicurezza, ESCO dall'area
    riservata (clienti all'interno dell'area riservata: 0)
140 CLIENTE_NORMALE-[Thread9 e identificatore 140349389399808] VADO A CASA
141 BANCARIO-[Thread1 e identificatore 140349800445696] nessun cliente e' in
```

```
    coda, mi sospendo
142 BANCARIO-[Thread0 e identificatore 140349876512512] nessun cliente e' in
    coda, mi sospendo
143 Pthread 9-esimo restituisce 9
144 Pthread 10-esimo restituisce 10
145 Pthread 11-esimo restituisce 11
146 Contatore coda clienti VIP: 0
147 Contatore coda clienti normali: 0
148 Numero clienti all'interno dell'area riservata: 0
149 Stato clienti: [ -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 ]
```

Codice 1.5: Output del programma del problema della Banca

1.5 Possibili problemi di Starvation o Deadlock

La soluzione presentata è stata realizzata evitando qualunque problema di Deadlock, dunque non sono presenti problemi di questo genere. Per quanto riguarda i problemi di Starvation, si presenta tale problema, per i thread di tipo *Clienti_Normali*, nel caso in cui continuino ad arrivare sempre clienti VIP i quali, avendo la precedenza nell'essere serviti, faranno sì che i clienti normali rimangano sempre all'interno della coda. É possibile evitare tale problema con l'aggiunta di un vincolo sul numero di clienti VIP che vengono serviti consecutivamente, mediante l'utilizzo di un contatore di clienti VIP serviti in successione ed un controllo nella scelta della coda da servire da parte, da parte dei thread *Bancari*.

2. Elicottero

2.1 Descrizione del Problema

Una **compagnia aerea** dispone di un unico **elicottero** che effettua, ogni giorno, **V** voli turistici sulla città: le partenze avvengono ad orari ben specifici e ogni volo ha una certa durata. L'elicottero dispone di **N** posti e come personale a bordo ha solo il **pilota**. La compagnia accetta sia **passeggeri singoli** che in **gruppo** (di **n** persone con $2 \leq n \leq N$). I gruppi di passeggeri hanno priorità rispetto ai passeggeri singoli. Dopo che i passeggeri/gruppi sono *saliti* sull'elicottero, all'orario prestabilito, il **pilota** chiude l'imbarco e parte per la destinazione. Al termine del volo, dopo l'atterraggio, i passeggeri/gruppi *scendono* dall'elicottero e quindi il pilota, dopo lo sbarco dei passeggeri, libera i posti sull'aeromobile.

Si implementi una soluzione usando il costrutto *monitor* per modellare la **compagnia aerea** e i *processi* per modellare i **passeggeri/gruppi** e il **pilota**. Nella soluzione si massimizzi l'utilizzo delle risorse. Si discuta se la soluzione proposta può presentare starvation e in caso positivo per quali processi, e si propongano modifiche e/o aggiunte per evitare la starvation.

2.2 Descrizione delle strutture dati

2.2.1 Array di passeggeri

Per risolvere il problema, utilizzeremo un array di **passeggeri** di tipo *int*, di dimensione pari al numero totale di passeggeri, dunque comprende sia i passeggeri singoli che quelli in gruppo, scelto per l'esecuzione del programma. Sul quale andremo a memorizzare l'*id* del thread *pilota* una volta che il passeggero/gruppo di passeggeri e' stato selezionato per salire sull'elicottero.

```
1    int *passeggeri;
```

L'array di **passeggeri** verrà inizializzato con il valore -1, indicando che il passeggero/gruppo di passeggeri non è ancora stato servito, mentre al termine del volo avrà valore -2, indicando che tale

passeggero/gruppo di passeggeri è stato servito. A tal proposito sono state realizzate due macro per rappresentare lo stato di ognuno dei passeggeri/gruppo di passeggeri:

```
1    #define PASSEGGERO_NON_SERVITO -1
2    #define PASSEGGERO_SERVITO -2
```

2.2.2 Realizzazione delle code

Ogni passeggero/gruppo di passeggeri non appena si reca dalla compagnia aerea, viene inserito in una delle due code messe a disposizione, in base al numero di persone che si presentano alla compagnia. Una coda per i passeggeri singoli ed una coda per i passeggeri in gruppo.

L'implementazione delle due code è stata realizzata mediane due array, **coda_passeggeri_singoli** e **coda_passeggeri_gruppo**. In cui, la prima è di tipo **int**, mentre per la seconda è stato definito un nuovo tipo di dato basato su una *struct*, contenente l'**id** del thread *Passeggeri_gruppo* e il **numero delle persone** di cui è composto il gruppo.

```
1    struct definizione_gruppo{
2        int id;
3        int num_persone;
4    };
5
6    typedef struct definizione_gruppo gruppo;
```

Entrambi i due array hanno dimensione pari al numero di passeggeri singoli e di gruppi di passeggeri scelto per l'esecuzione del programma.

Inizializzati a -1, per indicare l'assenza di passeggeri all'interno della coda, e successivamente contenenti l'id del thread *Passeggero*, in attesa che il pilota lo faccia salire sull'elicottero. Verranno, inoltre, utilizzati due rispettivi contatori, **contatore_passeggeri_singoli** e **contatore_passeggeri_gruppo**, inizializzati a 0, per indicare il numero di thread *Passeggeri* attualmente in coda.

```
1    int *coda_passeggeri_singoli;
2    int contatore_passeggeri_singoli;
3
4    gruppo *coda_passeggeri_gruppo;
5    int contatore_passeggeri_gruppo;
```

2.2.3 Implementazione del costrutto monitor

La protezione di tali strutture dati, verrà effettuata dal costrutto *monitor* realizzato mediante l'utilizzo del seguente semaforo binario di mutua esclusione.

```
1    pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
```

2.2.4 Variabili condizione

Per la comunicazione tra i thread di tipo *Passeggeri* (sia singoli che in gruppo) e i thread di tipo *Corrieri*, utilizzeremo le seguenti variabili condizione:

```
1    pthread_cond_t attesa_imbarco = PTHREAD_COND_INITIALIZER;
2    pthread_cond_t attesa_termine = PTHREAD_COND_INITIALIZER;
```

Dove **attesa_imbarco** rappresenta la variabile condizione in cui i thread *Passeggeri* (sia singoli che in gruppo) si bloccano in attesa che il thread *Pilota* li faccia salire sull'elicottero e proceda con la partenza dell'elicottero. Mentre la variabile condizione **attesa_termine** serve sempre ai thread *Passeggeri* (sia singoli che in gruppo), una volta che il thread *Pilota* li ha selezionati per salire sull'elicottero e dunque sono in attesa che il volo termini.

2.3 Implementazione della soluzione

2.3.1 Schema thread Passeggero

Lo schema del thread *Passeggero*, che può essere sia un passeggero singolo che un gruppo di passeggeri, è composto dalla chiamata della procedure entry **PRENOTA** e da un controllo per il caso in cui i voli per la giornata odierna sono finiti e dunque il passeggero è costretto a tornare a casa.

```
1    /* effettuo la prenotazione per un passeggero singolo */
2    PRENOTA(*pi, 1, &voli_finiti);
3
4    if (voli_finiti)
```

```
5           printf("PASSEGGERO_SINGOLO-[Thread%d e identificatore %lu] il pilota
                con ha terminato i voli per oggi, TORNO A CASA\n", *pi,
                pthread_self());
6       else
7           printf("PASSEGGERO_SINGOLO-[Thread%d e identificatore %lu] volo
                terminato, VADO A CASA\n", *pi, pthread_self());
```

Nel caso di un gruppo di passeggeri viene anche generato in modo random il numero di persone di cui è composto il gruppo, accertandosi che il numero sia maggiore uguale a 2 e minore uguale a N.

```
1       /* calcolo in modo random il numero di persone che compongono il gruppo */
2       num_persone = mia_random(NUM_POSTI);
3
4       /* mi assicuro che il numero di persone che compongono il gruppo non sia
            di una sola persona */
5       if (num_persone == 1)
6           num_persone++;
7
8       /* effettuo la prenotazione per un gruppo di passeggeri */
9       PRENOTA(*pi, num_persone, &voli_finiti);
10
11      if (voli_finiti)
12          printf("PASSEGGERI_GRUPPO-[Thread%d e identificatore %lu] il pilota ha
                terminato i voli per oggi, TORNIAMO A CASA\n", *pi, pthread_self()
               );
13      else
14          printf("PASSEGGERI_GRUPPO-[Thread%d e identificatore %lu] volo
                terminato, ANDIAMO A CASA\n", *pi, pthread_self());
```

All'interno della procedure entry **PRENOTA**, realizzata mediante l'utilizzo di un semaforo di mutua esclusione **mutex**, in base al numero di persone di cui si vuole effettuare la prenotazione dei posti, valore passato come parametro della funzione, il thread *Passeggero* inserisce il proprio *id* nella rispettiva coda di attesa.

Successivamente, viene verificata la condizione di selezione da parte del thread *Pilota* e in caso contrario si sospende effettuando una *wait* sulla variabile condizione **attesa_imbarco**, in attesa che

il pilota lo selezioni per salire sull'elicottero. Viene inoltre controllato che il pilota non abbia finito i voli previsti per la giornata odierna e nel caso in cui i voli sono terminati viene notificato l'accaduto, stampando su standard output una frase significativa, e il processo termina la sua esecuzione.

Risvegliato da parte del thread *Pilota*, il passeggero singolo/in gruppo stampa su standard output che il pilota, specificando anche l'*id*, lo ha selezionato per salire sull'elicottero. Dopodiché si sospende in attesa che il pilota termini il volo, mediante una *wait* sulla variabile condizione **attesa_termine**.

Risvegliato nuovamente da parte del thread *Pilota*, stampa su standard output la conferma del termine del volo e termina la sua esecuzione.

```
1   void PRENOTA(int id, int num_persone, Boolean *voli_finiti)
2   {
3       pthread_mutex_lock(&mutex);
4
5       /* mi inserisco nella coda in base al numero di posti che sto
           prenotando */
6       if (num_persone == 1)   /* prenoto per un passeggero singolo */
7       {
8           printf("PASSEGGERO_SINGOLO-[Thread%d e identificatore %lu] mi
               inserisco nella coda dei passeggeri singoli \n", id,
               pthread_self());
9
10          /* inserisco il mio id nella coda dei passeggeri singoli e
               incremento il numero di persone singole in attesa */
11          coda_passeggeri_singoli[contatore_passeggeri_singoli] = id;
12          contatore_passeggeri_singoli++;
13      }
14      else    /* prenoto per un gruppo di passeggeri */
15      {
16          printf("PASSEGGERI_GRUPPO-[Thread%d e identificatore %lu] siamo un
                gruppo di passeggeri di %d persone, ci inseriamo nella coda
               dei passeggeri in gruppo \n", id, pthread_self(), num_persone);
17
18          /* inserisco il mio id nella coda dei passeggeri in gruppo e
               incremento il numero di passeggeri in gruppo in attesa */
```

```
19              coda_passeggeri_gruppo[contatore_passeggeri_gruppo].id = id;
20              coda_passeggeri_gruppo[contatore_passeggeri_gruppo].num_persone =
                    num_persone;
21              contatore_passeggeri_gruppo++;
22          }
23
24          while (passeggeri[id - 1] == PASSEGGERO_NON_SERVITO)
25          {
26              if (voli_eseguiti == NUM_VOLI)  /* se i voli sono terminati torno
                    a casa */
27              {
28                  *voli_finiti = true; /* indico che i voli sono stati terminati
                        , dunque i passeggeri tornano a casa */
29                  pthread_mutex_unlock(&mutex);
30                  return;
31              }
32
33              /* mi metto in attesa che il pilota mi/ci faccia salire sull'
                    elicottero */
34              pthread_cond_wait(&attesa_imbarco, &mutex);
35          }
36
37          if (num_persone == 1)
38              printf("PASSEGGERO_SINGOLO-[Thread%d e identificatore %lu] il
                    pilota con id %d mi ha fatto salire sull'elicottero\n", id,
                    pthread_self(), passeggeri[id - 1]);
39          else
40              printf("PASSEGGERI_GRUPPO-[Thread%d e identificatore %lu] il
                    pilota con id %d ci ha fatto salire sull'elicottero\n", id,
                    pthread_self(), passeggeri[id - 1]);
41
42          while(passeggeri[id - 1] != PASSEGGERO_SERVITO)
43          {
44              /* l'elicottero e' appena partito, attendo il termine del volo */
45              pthread_cond_wait(&attesa_termine, &mutex);
46          }
```

```
47
48          pthread_mutex_unlock(&mutex);
49      }
```

Codice 2.1: Procedure Entry PRENOTA del problema dell'Elicottero

2.3.2 Schema thread Pilota

Lo schema del thread *Pilota* è composto da un ciclo che viene effettuato V volte, pari al numero di voli che il pilota deve effettuare nella giornata odierna, in cui ripete le seguenti operazioni:

- attende che arrivi l'ora per la partenza, mediante una sleep
- procede con l'imbarco e successiva partenza dell'elicottero mediante la procedure entry **IMBARCO**
- simulazione del volo mediante una sleep
- termina il volo e fa scendere i passeggeri mediante la procedure entry **VOLO_TERMINATO**

```
1   /* inizialmente il numero di posti disponibili sara' pari a NUM_POSTI */
2   num_posti_disponibili = NUM_POSTI;
3
4   /* eseguo i voli di questa giornata */
5   for (voli_eseguiti = 0; voli_eseguiti < NUM_VOLI; voli_eseguiti++)
6   {
7       /* attendo che arrivi l'ora per la partenza, mediante una sleep di 2
            secondi */
8       sleep(2);
9
10      /* faccio salire i passeggeri sull'elicottero */
11      printf("PILOTA-[Thread%d e identificatore %lu] e' arrivata l'ora di
            partire, effettuo l'imbarco per il %d-esimo volo\n", *pi,
            pthread_self(), voli_eseguiti);
12      IMBARCO(*pi, &num_posti_disponibili, &id_selezionati, &
            contatore_selezionati);
13
```

```
14          /* simulo il volo mediante una sleep di 4 secondi */
15          sleep(4);
16
17          /* termino il volo e faccio scendere i passeggeri */
18          VOLO_TERMINATO(*pi, &num_posti_disponibili, &id_selezionati, &
                contatore_selezionati);
19      }
```

All'interno della procedure entry **IMBARCO**, realizzata mediante l'utilizzo del semaforo di mutua esclusione **mutex**, il pilota effettua, come da specifica, prima il controllo se ci sono gruppi di passeggeri in coda, facendoli così salire per primi, e successivamente se rimangono ancora posti disponibili sull'elicottero controllerà se ci sono passeggeri singoli in coda. Nel caso siano presenti dei gruppi di passeggeri in coda, il pilota occuperà i posti rimanenti facendo entrare un gruppo alla volta, in base al loro ordine di arrivo. Nel caso in cui non ci sono abbastanza posti disponibili per far entrare il gruppo, tale gruppo rimarrà nella coda in attesa del prossimo volo e si procederà con il controllo di tutti i restanti gruppi in coda.

Per ogni gruppo selezionato per salire in elicottero, verranno eseguite le seguenti operazioni:

- inserire l'*id* del thread *Pilota* all'interno dell'array di **passeggeri** con indice l'*id* del thread *Passeggeri_Gruppo*, modificando così il suo stato.
- inserire l'*id* del thread *Passeggeri_Gruppo* nell'array **id_selezionati**, usato dal thread *Pilota* per tenere traccia dei passeggeri/gruppi di passeggeri che sono stati scelti per salire sull'elicottero
- aumentare il contatore dei passeggeri/gruppi di passeggeri selezionati
- diminuire il numero di posti disponibili in base al numero di posti che il gruppo occupa
- rimuovere il gruppo dalla coda
- diminuire di uno il contatore dei gruppi in coda

Una volta occupati più posti possibili all'interno dell'elicottero facendo salire i gruppi, nel caso siano rimasti dei posti e ci sono passeggeri singoli in coda, si procede occupando i posti rimanenti con i passeggeri singoli.

Per ogni passeggero singolo selezionato per salire in elicottero, verranno eseguite le seguenti operazioni:

- inserire l'*id* del thread *Pilota* all'interno dell'array di **passeggeri** con indice l'*id* del thread *Passeggero_Singolo*, modificando così il suo stato.
- inserire l'*id* del thread *Passeggero_Singolo* nell'array **id_selezionati**, usato dal thread *Pilota* per tenere traccia dei passeggeri/gruppi di passeggeri che sono stati scelti per salire sull'elicottero
- aumentare il contatore dei passeggeri/gruppi di passeggeri selezionati
- diminuire il numero di posti disponibili di uno
- rimuovere il passeggero singolo dalla coda
- diminuire di uno il contatore dei passeggeri singoli

Terminata la scelta dei passeggeri/gruppi di passeggeri che saliranno sull'elicottero, il thread *Pilota* notificherà che la fase di imbarco è stata terminata e pertanto si prepara alla partenza, mediante l'utilizzo di una *signal* in *broadcast* sulla variabile condizione **attesa_imbarco**. Terminando così la procedure entry **IMBARCO**.

```
1  void IMBARCO(int id, int *num_posti_disponibili, int **id_selezionati, int
       *contatore_selezionati)
2  {
3      pthread_mutex_lock(&mutex);
4
5      int i;  /* variabile contatore usata per la selezione del gruppo di
           passeggeri */
6      int j;  /* variabile contatore usata per spostare i passeggeri singoli
           /in gruppo all'interno della coda */
7      Boolean trovato;    /* variabile booleana che indica se e' stato
           trovato un gruppo di passeggeri che puo' riempire i posti
           disponibili in elicottero */
8
9      /* controllo se ci sono gruppi di passeggeri in attesa */
```

```
10          if (contatore_passeggeri_gruppo > 0)
11          {
12              trovato = true; /* imposto 'trovato' a true per entrare nel ciclo
                    */
13              while (trovato == true)
14              {
15                  /* imposto 'trovato' a false cose' che se la ricerca non ha
                        avuto esito positivo usciro' dal ciclo */
16                  trovato = false;
17
18                  for (i = 0; i < contatore_passeggeri_gruppo; i++)
19                  {
20                      /* occupo i posti disponibili inserendo prima i passeggeri
                            in gruppo */
21                      if (*num_posti_disponibili - coda_passeggeri_gruppo[i].
                            num_persone >= 0)
22                      {
23                          /* faccio salire il gruppo di passeggeri in elicottero
                                */
24                          passeggeri[coda_passeggeri_gruppo[i].id - 1] = id;
25                          (*id_selezionati)[*contatore_selezionati] =
                                coda_passeggeri_gruppo[i].id;
26                          printf("PILOTA-[Thread%d e identificatore %lu] faccio
                                salire il gruppo di passeggeri con id %d composto
                                da %d persone. Posti ancora disponibili: %d\n", id,
                                 pthread_self(), coda_passeggeri_gruppo[i].id,
                                coda_passeggeri_gruppo[i].num_persone, *
                                num_posti_disponibili - coda_passeggeri_gruppo[i].
                                num_persone);
27
28                          /* aumento di uno il numero dei passeggeri/gruppi di
                                passeggeri selezionati per salire sull'elicottero
                                */
29                          (*contatore_selezionati)++;
30
31                          /* diminuisco il numero di posti disponibili */
```

```
32                          *num_posti_disponibili -= coda_passeggeri_gruppo[i].
                                num_persone;
33
34                          /* rimuovo dalla coda il gruppo di passeggeri
                                selezionato */
35                          for (j = i; j < contatore_passeggeri_gruppo-1; j++)
                                /* sposto tutti gli altri gruppi di passeggeri di
                                una posizione in coda */
36                               coda_passeggeri_gruppo[j] = coda_passeggeri_gruppo
                                     [j+1];
37
38                          /* resetto l'ultima posizione della coda */
39                          (coda_passeggeri_gruppo[j]).id = -1;
40                          (coda_passeggeri_gruppo[j]).num_persone = 0;
41
42                          /* diminuisco di uno il numero di gruppi di passeggeri
                                in coda */
43                          contatore_passeggeri_gruppo--;
44
45                          /* dato che ho trovato un gruppo che riempie i posti
                                rimanenti imposto 'trovato' a true */
46                          trovato = true;
47
48                          /* termino la ricerca del gruppo di passeggeri */
49                          break;
50                      }
51                  }
52              }
53          }
54
55          while (contatore_passeggeri_singoli > 0 && *num_posti_disponibili > 0)
56          {
57              /* faccio salire il passeggero singolo in elicottero */
58              passeggeri[coda_passeggeri_singoli[0] - 1] = id;
59              (*id_selezionati)[*contatore_selezionati] =
                    coda_passeggeri_singoli[0];
```

```
60              printf("PILOTA-[Thread%d e identificatore %lu] faccio salire il
                    passeggero singolo con id %d. Posti ancora disponibili: %d\n",
                    id, pthread_self(), coda_passeggeri_singoli[0], *
                    num_posti_disponibili - 1);
61
62              /* aumento di uno il numero dei passeggeri/gruppi di passeggeri
                    selezionati per salire sull'elicottero */
63              (*contatore_selezionati)++;
64
65              /* diminuisco il numero di posti disponibili */
66              (*num_posti_disponibili)--;
67
68              /* rimuovo dalla coda il passeggero selezionato */
69              for (j = 0; j < contatore_passeggeri_singoli - 1; j++)  /* sposto
                    tutti gli altri passeggeri di una posizione in coda */
70                  coda_passeggeri_singoli[j] = coda_passeggeri_singoli[j+1];
71
72              /* resetto l'ultima posizione della coda */
73              coda_passeggeri_singoli[j] = -1;
74
75              /* diminuisco di uno il numero dei passeggeri singoli in coda */
76              contatore_passeggeri_singoli--;
77          }
78
79          printf("PILOTA-[Thread%d e identificatore %lu] elicottero in partenza,
                    numero passeggeri: %d\n", id, pthread_self(), NUM_POSTI - *
                    num_posti_disponibili);
80
81          /* notifico i passeggeri che l'elicottero e' in partenza */
82          pthread_cond_broadcast(&attesa_imbarco);
83
84          pthread_mutex_unlock(&mutex);
85      }
```

Codice 2.2: Procedure Entry IMBARCO del problema dell'Elicottero

All'interno della procedure entry **VOLO_TERMINATO**, realizzata mediante l'utilizzo del se-

maforo di mutua esclusione **mutex**, il thread *Pilota* per ogni passeggero/gruppo di passeggeri modifica il suo stato, all'interno dell'array **passeggeri**, in *PASSEGGERO_SERVITO*. Rimuove il passeggero/gruppo di passeggeri dall'array *id_selezionati* inserendo al suo posto il valore -1 e diminuisce il contatore dei passeggeri/gruppi di passeggeri selezionati di uno. Infine ripristina il numero di posti disponibili e notifica tutti i passeggeri che il volo è terminato, mediante l'utilizzo di una *signal* in *broadcast* sulla variabile condizione *attesa_termine*. Terminando così la procedure entry *VOLO_TERMINATO*.

```
1   void VOLO_TERMINATO(int id, int *num_posti_disponibili, int **
        id_selezionati, int *contatore_selezionati)
2   {
3       pthread_mutex_lock(&mutex);
4
5       printf("PILOTA-[Thread%d e identificatore %lu] volo %d-esimo terminato
            , faccio scendere i passeggeri\n", id, pthread_self(),
            voli_eseguiti);
6
7       while (*contatore_selezionati > 0)
8       {
9           /* imposto lo stato dei passeggeri in PASSEGGERO_SERVITO (-2) per
                indicare che il loro volo e' terminato */
10          passeggeri[(*id_selezionati)[*contatore_selezionati -1] - 1] =
                PASSEGGERO_SERVITO;
11
12          /* libero il posto in elicottero */
13          (*id_selezionati)[*contatore_selezionati - 1] = -1;
14
15          /* diminuisco il contatore dell'array id_selezionati */
16          (*contatore_selezionati)--;
17      }
18
19      /* ripristino il numero di posti disponibili in elicottero pari a
            NUM_POSTI */
20      *num_posti_disponibili = NUM_POSTI;
21
22      /* notifico i passeggeri che il volo e' terminato */
```

```
23          pthread_cond_broadcast(&attesa_termine);
24
25          pthread_mutex_unlock(&mutex);
26      }
```

Codice 2.3: Procedure Entry VOLO_TERMINATO del problema dell'Elicottero

Una volta terminati i voli previsti, il thread *Pilota* controlla se ci sono ancora passeggeri in coda, ed in caso positivo li notifica del termine dei voli mediante l'uso di una *signal* in *broadcast* sulla variabile condizione **attesa_imbarco**. I thread *Passeggeri* (sia singoli che in gruppo), si accorgeranno che i voli sono finiti e termineranno la loro esecuzione.

```
1       if (contatore_passeggeri_singoli > 0 || contatore_passeggeri_gruppo > 0)
2       {
3           printf("PILOTA-[Thread%d e identificatore %lu] i voli sono finiti ma
                ci sono ancora passeggeri in coda, li notifico e VADO A CASA\n", *
                pi, pthread_self());
4
5           /* notifico i passeggeri in attesa */
6           pthread_cond_broadcast(&attesa_imbarco);
7       }
8       else
9           printf("PILOTA-[Thread%d e identificatore %lu] voli terminati, VADO A
                CASA\n", *pi, pthread_self());
```

2.4 Esempio di Funzionamento

L'esecuzione del programma avviene mediante l'utilizzo di quattro argomenti, rispettivamente sono:

1. *NUMERO_VOLI*
2. *NUMERO_POSTI*
3. *NUMERO_PASSEGGERI_SINGOLI*
4. *NUMERO_PASSEGGERI_GRUPPO*

Per consentire una distinzione tra passeggeri singoli e passeggeri in gruppo è stato inserito un vincolo aggiuntivo sul *NUMERO_POSTI*. In quanto tale numero influisce sulla scelta in modo random del numero di persone di cui ogni gruppo è composto. Tale numero dovrà pertanto essere maggiore o uguale a 2 per consentire una corretta suddivisione delle categorie, evitando cosi' di generare gruppi formati da una sola persona.

Una tipica invocazione del programma è la seguente:

```
$ ./elicottero 6 5 6 4
```

Codice 2.4: Esempio di invocazione del programma del problema dell'Elicottero

In questo caso verranno effettuati sei sei nella giornata odierna, il numero di posti disponibili sull'elicottero sarà pari a cinque, ci saranno sei passeggeri singoli e quattro passeggeri in gruppo. Il totale delle prenotazione dei passeggeri sarà dunque di dieci.

```
Numero VOLI da effettuare: 6
Numero POSTI sull'elicottero: 5
Numero PASSEGGERI_SINGOLI: 6
Numero PASSEGGERI_GRUPPO: 4
Numero totale di PASSEGGERI: 10
Sto per creare il thread PILOTA 0-esimo
SONO IL MAIN e ho creato il Pthread PILOTA 0-esimo con id=139627529303808
Sto per creare il thread PASSEGGERI_GRUPPO 1-esimo
PILOTA-[Thread0 e identificatore 139627529303808] STO ARRIVANDO
SONO IL MAIN e ho creato il Pthread PASSEGGERI_GRUPPO 1-esimo con id
    =139627386693376
PASSEGGERI_GRUPPO-[Thread1 e identificatore 139627386693376] STIAMO
    ARRIVANDO
PASSEGGERI_GRUPPO-[Thread1 e identificatore 139627386693376] siamo un
    gruppo di passeggeri di 2 persone, ci inseriamo nella coda dei
    passeggeri in gruppo
Sto per creare il thread PASSEGGERI_GRUPPO 2-esimo
SONO IL MAIN e ho creato il Pthread PASSEGGERI_GRUPPO 2-esimo con id
    =139627520911104
PASSEGGERI_GRUPPO-[Thread2 e identificatore 139627520911104] STIAMO
    ARRIVANDO
```

```
16  PASSEGGERI_GRUPPO-[Thread2 e identificatore 139627520911104] siamo un
        gruppo di passeggeri di 2 persone, ci inseriamo nella coda dei
        passeggeri in gruppo
17  Sto per creare il thread PASSEGGERI_GRUPPO 3-esimo
18  PILOTA-[Thread0 e identificatore 139627529303808] e' arrivata l'ora di
        partire, effettuo l'imbarco per il 0-esimo volo
19  PILOTA-[Thread0 e identificatore 139627529303808] faccio salire il gruppo
        di passeggeri con id 1 composto da 2 persone. Posti ancora disponibili:
         3
20  PILOTA-[Thread0 e identificatore 139627529303808] faccio salire il gruppo
        di passeggeri con id 2 composto da 2 persone. Posti ancora disponibili:
         1
21  PILOTA-[Thread0 e identificatore 139627529303808] elicottero in partenza,
        numero passeggeri: 4
22  PASSEGGERI_GRUPPO-[Thread1 e identificatore 139627386693376] il pilota con
         id 0 ci ha fatto salire sull'elicottero
23  SONO IL MAIN e ho creato il Pthread PASSEGGERI_GRUPPO 3-esimo con id
        =139627440633600
24  PASSEGGERI_GRUPPO-[Thread2 e identificatore 139627520911104] il pilota con
         id 0 ci ha fatto salire sull'elicottero
25  PASSEGGERI_GRUPPO-[Thread3 e identificatore 139627440633600] STIAMO
        ARRIVANDO
26  PASSEGGERI_GRUPPO-[Thread3 e identificatore 139627440633600] siamo un
        gruppo di passeggeri di 2 persone, ci inseriamo nella coda dei
        passeggeri in gruppo
27  Sto per creare il thread PASSEGGERO_SINGOLO 4-esimo
28  SONO IL MAIN e ho creato il Pthread PASSEGGERO_SINGOLO 4-esimo con id
        =139627432240896
29  PASSEGGERO_SINGOLO-[Thread4 e identificatore 139627432240896] STO
        ARRIVANDO
30  PASSEGGERO_SINGOLO-[Thread4 e identificatore 139627432240896] mi inserisco
         nella coda dei passeggeri singoli
31  Sto per creare il thread PASSEGGERO_SINGOLO 5-esimo
32  SONO IL MAIN e ho creato il Pthread PASSEGGERO_SINGOLO 5-esimo con id
        =139627423848192
33  PASSEGGERO_SINGOLO-[Thread5 e identificatore 139627423848192] STO
```

```
    ARRIVANDO
34  PASSEGGERO_SINGOLO-[Thread5 e identificatore 139627423848192] mi inserisco
        nella coda dei passeggeri singoli
35  Sto per creare il thread PASSEGGERI_GRUPPO 6-esimo
36  SONO IL MAIN e ho creato il Pthread PASSEGGERI_GRUPPO 6-esimo con id
    =139627415455488
37  PASSEGGERI_GRUPPO-[Thread6 e identificatore 139627415455488] STIAMO
    ARRIVANDO
38  PASSEGGERI_GRUPPO-[Thread6 e identificatore 139627415455488] siamo un
    gruppo di passeggeri di 2 persone, ci inseriamo nella coda dei
    passeggeri in gruppo
39  PILOTA-[Thread0 e identificatore 139627529303808] volo 0-esimo terminato,
    faccio scendere i passeggeri
40  PASSEGGERI_GRUPPO-[Thread1 e identificatore 139627386693376] volo
    terminato, ANDIAMO A CASA
41  PASSEGGERI_GRUPPO-[Thread2 e identificatore 139627520911104] volo
    terminato, ANDIAMO A CASA
42  Sto per creare il thread PASSEGGERO_SINGOLO 7-esimo
43  SONO IL MAIN e ho creato il Pthread PASSEGGERO_SINGOLO 7-esimo con id
    =139627407062784
44  PASSEGGERO_SINGOLO-[Thread7 e identificatore 139627407062784] STO
    ARRIVANDO
45  PASSEGGERO_SINGOLO-[Thread7 e identificatore 139627407062784] mi inserisco
        nella coda dei passeggeri singoli
46  Sto per creare il thread PASSEGGERO_SINGOLO 8-esimo
47  SONO IL MAIN e ho creato il Pthread PASSEGGERO_SINGOLO 8-esimo con id
    =139627398670080
48  PASSEGGERO_SINGOLO-[Thread8 e identificatore 139627398670080] STO
    ARRIVANDO
49  PASSEGGERO_SINGOLO-[Thread8 e identificatore 139627398670080] mi inserisco
        nella coda dei passeggeri singoli
50  PILOTA-[Thread0 e identificatore 139627529303808] e' arrivata l'ora di
    partire, effettuo l'imbarco per il 1-esimo volo
51  PILOTA-[Thread0 e identificatore 139627529303808] faccio salire il gruppo
    di passeggeri con id 3 composto da 2 persone. Posti ancora disponibili:
        3
```

```
52  PILOTA-[Thread0 e identificatore 139627529303808] faccio salire il gruppo
        di passeggeri con id 6 composto da 2 persone. Posti ancora disponibili:
         1
53  PILOTA-[Thread0 e identificatore 139627529303808] faccio salire il
        passeggero singolo con id 4. Posti ancora disponibili: 0
54  PILOTA-[Thread0 e identificatore 139627529303808] elicottero in partenza,
        numero passeggeri: 5
55  PASSEGGERI_GRUPPO-[Thread6 e identificatore 139627415455488] il pilota con
         id 0 ci ha fatto salire sull'elicottero
56  PASSEGGERO_SINGOLO-[Thread4 e identificatore 139627432240896] il pilota
        con id 0 mi ha fatto salire sull'elicottero
57  PASSEGGERI_GRUPPO-[Thread3 e identificatore 139627440633600] il pilota con
         id 0 ci ha fatto salire sull'elicottero
58  Sto per creare il thread PASSEGGERO_SINGOLO 9-esimo
59  SONO IL MAIN e ho creato il Pthread PASSEGGERO_SINGOLO 9-esimo con id
        =139627037980416
60  PASSEGGERO_SINGOLO-[Thread9 e identificatore 139627037980416] STO
        ARRIVANDO
61  PASSEGGERO_SINGOLO-[Thread9 e identificatore 139627037980416] mi inserisco
         nella coda dei passeggeri singoli
62  Sto per creare il thread PASSEGGERO_SINGOLO 10-esimo
63  SONO IL MAIN e ho creato il Pthread PASSEGGERO_SINGOLO 10-esimo con id
        =139627029587712
64  PASSEGGERO_SINGOLO-[Thread10 e identificatore 139627029587712] STO
        ARRIVANDO
65  PASSEGGERO_SINGOLO-[Thread10 e identificatore 139627029587712] mi
        inserisco nella coda dei passeggeri singoli
66  PILOTA-[Thread0 e identificatore 139627529303808] volo 1-esimo terminato,
        faccio scendere i passeggeri
67  PASSEGGERI_GRUPPO-[Thread6 e identificatore 139627415455488] volo
        terminato, ANDIAMO A CASA
68  PASSEGGERO_SINGOLO-[Thread4 e identificatore 139627432240896] volo
        terminato, VADO A CASA
69  PASSEGGERI_GRUPPO-[Thread3 e identificatore 139627440633600] volo
        terminato, ANDIAMO A CASA
70  PILOTA-[Thread0 e identificatore 139627529303808] e' arrivata l'ora di
```

```
    partire, effettuo l'imbarco per il 2-esimo volo
71  PILOTA-[Thread0 e identificatore 139627529303808] faccio salire il
    passeggero singolo con id 5. Posti ancora disponibili: 4
72  PILOTA-[Thread0 e identificatore 139627529303808] faccio salire il
    passeggero singolo con id 7. Posti ancora disponibili: 3
73  PILOTA-[Thread0 e identificatore 139627529303808] faccio salire il
    passeggero singolo con id 8. Posti ancora disponibili: 2
74  PILOTA-[Thread0 e identificatore 139627529303808] faccio salire il
    passeggero singolo con id 9. Posti ancora disponibili: 1
75  PILOTA-[Thread0 e identificatore 139627529303808] faccio salire il
    passeggero singolo con id 10. Posti ancora disponibili: 0
76  PILOTA-[Thread0 e identificatore 139627529303808] elicottero in partenza,
    numero passeggeri: 5
77  PASSEGGERO_SINGOLO-[Thread5 e identificatore 139627423848192] il pilota
    con id 0 mi ha fatto salire sull'elicottero
78  PASSEGGERO_SINGOLO-[Thread8 e identificatore 139627398670080] il pilota
    con id 0 mi ha fatto salire sull'elicottero
79  PASSEGGERO_SINGOLO-[Thread9 e identificatore 139627037980416] il pilota
    con id 0 mi ha fatto salire sull'elicottero
80  PASSEGGERO_SINGOLO-[Thread7 e identificatore 139627407062784] il pilota
    con id 0 mi ha fatto salire sull'elicottero
81  PASSEGGERO_SINGOLO-[Thread10 e identificatore 139627029587712] il pilota
    con id 0 mi ha fatto salire sull'elicottero
82  PILOTA-[Thread0 e identificatore 139627529303808] volo 2-esimo terminato,
    faccio scendere i passeggeri
83  PASSEGGERO_SINGOLO-[Thread7 e identificatore 139627407062784] volo
    terminato, VADO A CASA
84  PASSEGGERO_SINGOLO-[Thread5 e identificatore 139627423848192] volo
    terminato, VADO A CASA
85  PASSEGGERO_SINGOLO-[Thread9 e identificatore 139627037980416] volo
    terminato, VADO A CASA
86  PASSEGGERO_SINGOLO-[Thread10 e identificatore 139627029587712] volo
    terminato, VADO A CASA
87  PASSEGGERO_SINGOLO-[Thread8 e identificatore 139627398670080] volo
    terminato, VADO A CASA
88  PILOTA-[Thread0 e identificatore 139627529303808] e' arrivata l'ora di
```

```
    partire, effettuo l'imbarco per il 3-esimo volo
89  PILOTA-[Thread0 e identificatore 139627529303808] elicottero in partenza,
    numero passeggeri: 0
90  PILOTA-[Thread0 e identificatore 139627529303808] volo 3-esimo terminato,
    faccio scendere i passeggeri
91  PILOTA-[Thread0 e identificatore 139627529303808] e' arrivata l'ora di
    partire, effettuo l'imbarco per il 4-esimo volo
92  PILOTA-[Thread0 e identificatore 139627529303808] elicottero in partenza,
    numero passeggeri: 0
93  PILOTA-[Thread0 e identificatore 139627529303808] volo 4-esimo terminato,
    faccio scendere i passeggeri
94  PILOTA-[Thread0 e identificatore 139627529303808] e' arrivata l'ora di
    partire, effettuo l'imbarco per il 5-esimo volo
95  PILOTA-[Thread0 e identificatore 139627529303808] elicottero in partenza,
    numero passeggeri: 0
96  PILOTA-[Thread0 e identificatore 139627529303808] volo 5-esimo terminato,
    faccio scendere i passeggeri
97  PILOTA-[Thread0 e identificatore 139627529303808] voli terminati, VADO A
    CASA
98  Pthread 0-esimo restituisce 0
99  Pthread 1-esimo restituisce 1
100 Pthread 2-esimo restituisce 2
101 Pthread 3-esimo restituisce 3
102 Pthread 4-esimo restituisce 4
103 Pthread 5-esimo restituisce 5
104 Pthread 6-esimo restituisce 6
105 Pthread 7-esimo restituisce 7
106 Pthread 8-esimo restituisce 8
107 Pthread 9-esimo restituisce 9
108 Pthread 10-esimo restituisce 10
109 Contatore coda passeggeri singoli: 0
110 Contatore coda passeggeri in gruppo: 0
111 Stato passeggeri: [ -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 ]
```

Codice 2.5: Output del programma del problema dell'Elicottero

2.5 Possibili problemi di Starvation o Deadlock

La soluzione presentata è stata realizzata evitando qualunque problema di Deadlock, dunque non sono presenti problemi di questo genere. Per quanto riguarda i problemi di Starvation, si presenta tale problema, per i thread di tipo *Passeggeri_Singoli*, nel caso in cui continuino ad arrivare sempre passeggeri in gruppo che riescono ad occupare tutti i posti dell'elicottero. Non lasciando così alcun posto disponibile per i passeggeri singoli, i quali rimarranno continuamente in coda. Per ovviare a questo tipo di problema senza creare un ulteriore problema per il caso opposto, eventuale Starvation dei thread *Passeggeri_Gruppo*, bisognerebbe prima far salire un gruppo di passeggeri, così da rispettare il vincolo di precedenza, dopodiché alternare la scelta dei passeggeri singoli a quelli in gruppo, ove possibile, in modo tale da evitare che entrambe le tipologie di thread vadano in situazioni di Starvation.

3. Officina

3.1 Descrizione del Problema

In un'**officina**, arrivano **automobili** per effettuare due tipi di controllo: il bollino blu per i gas di scarico (B) o il tagliando (T). Nell'officina ci sono N **operai** che effettuano i controlli: $2 * \frac{N}{3}$ di "tipo 0" e i rimanenti $\frac{N}{3}$ di "tipo 1". Gli operai di "tipo 0" sono autorizzati ad effettuare i controlli per il bollino blu, ed i tagliandi. Essendo il controllo per il bollino blu più breve, ha priorità sui tagliandi. Gli operai di "tipo1" possono effettuare solo i tagliandi. Ogni automobile attende un operaio libero, a seconda del controllo che deve fare. L'operaio, effettuato il controllo di durata variabile random *s* secondi, lascia libera l'auto (la durata dell'operazione è stabilita dall'operaio stesso).

Si implementi una soluzione usando il costrutto *monitor* per modellare l'**officina** e i *processi* per modellare le **auto** e gli **operai**. Nella soluzione si massimizzi l'utilizzo delle risorse. Si discuta se la soluzione proposta può presentare starvation e in caso positivo per quali processi, e si propongano modifiche e/o aggiunte per evitare la starvation.

3.2 Descrizione delle strutture dati

3.2.1 Array di automobili

Per risolvere il problema, utilizzeremo un array di **automobili** di tipo *int*, di dimensione pari al numero di auto scelto per l'esecuzione del programma. Sul quale andremo a memorizzare l'id del thread *operaio* che si occuperà di effettuare il controllo sull'automobile.

```
1    int *automobili;
```

L'array di **automobili** verrà inizializzato con il valore -1, indicando che tale auto non è ancora stata servita, mentre al termine del controllo avrà valore -2, indicando che tale automobile è stata servita. A tal proposito sono state realizzate le seguenti due macro per rappresentare lo stato di ognuna delle automobili:

```
1    #define AUTO_NON_SERVITA -1
```

```
2    #define AUTO_SERVITA -2
```

3.2.2 Realizzazione delle code

Ogni auto non appena si reca in officina, viene inserita in una delle due code messe a disposizione, in base al tipo di controllo che deve effettuare. Una coda per le auto che devono effettuare un controllo per il bollino blu ed un'altra per le auto che devono effettuare il controllo per il tagliando.

L'implementazione delle due code è stata realizzata mediante due array, **coda_bollino_blu** e **coda_tagliando**, di tipo *int* e di dimensione pari al numero di auto scelte per l'esecuzione del programma.

Inizializzati a -1, per indicare l'assenza di automobili all'interno della coda, e successivamente contenenti l'id del thread *Auto* in attesa che un operaio effettui il controllo sull'auto. Verranno, inoltre, utilizzati due rispettivi contatori, **contatore_bollino_blu** e **contatore_tagliando**, inizializzati a 0, per indicare il numero di thread *Auto* attualmente in coda.

```
1    int *coda_bollino_blu;
2    int contatore_bollino_blu;
3
4    int *coda_tagliando;
5    int contatore_tagliando;
```

3.2.3 Implementazione del costrutto monitor

La protezione di tali strutture dati, verrà effettuata dal costrutto *monitor* realizzato mediante l'utilizzo del seguente semaforo binario di mutua esclusione.

```
1    pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
```

3.2.4 Variabili condizione

Per la comunicazione tra i thread di tipo *Auto* e i thread di tipo *Operai*, utilizzeremo le seguenti variabili condizione:

```
1    pthread_cond_t auto_in_coda = PTHREAD_COND_INITIALIZER;
2    pthread_cond_t attesa_controllo = PTHREAD_COND_INITIALIZER;
```

```
3     pthread_cond_t attesa_termine = PTHREAD_COND_INITIALIZER;
```

Dove **auto_in_coda** rappresenta la variabile condizione in cui i thread *Operai* si bloccano in attesa che un automobile arrivi in officina e si inserisca in una delle due code, pronta per poter essere servita da un operaio. Mentre la variabile condizione **attesa_controllo** serve ai thread *Auto* per bloccarsi in attesa che un thread *Operaio* la selezioni per effettuare il controllo. Infine, **attesa_termine** è la variabile condizione che serve ai thread *Auto*, una volta che un thrad *Operaio* ha selezionato l'auto e ha appena iniziato il controllo, per bloccarsi in attesa che l'operaio termini il lavoro.

3.3 Implementazione della soluzione

3.3.1 Schema thread Auto

Lo schema del thread *Auto* è composto dalla generazione random del tipo di controllo che vuole effettuare e successivamente dalla chiamata della procedure entry **AUTO_ENTRA**.

```
1     /* effettuo la scelta del tipo di operazione da effettuare */
2     tipo_operazione = generazione_random_tipo_operazione();
3
4     AUTO_ENTRA(*pi, tipo_operazione);
```

All'interno della procedure entry **AUTO_ENTRA**, realizzata mediante l'utilizzo del semaforo di mutua esclusione **mutex**, in base al tipo di controllo che l'auto deve effettuare (controllo per il bollino blu o controllo per il tagliando), il thread inserisce il proprio *id* nella rispettiva coda per il controllo da effettuare ed effettua una *signal* in *broadcast* sulla variabile condizione **auto_in_coda**, risvegliando tutti i thread *Operaio* in attesa.

Successivamente, viene verificata la condizione di selezione da parte di un thread *Operaio* e in caso contrario si sospende effettuando una *wait* sulla variabile condizione **attesa_controllo**, in attesa che un operaio la selezioni per effettuare il controllo.

Risvegliato da parte di un thread *Operaio*, l'auto stampa su standard output che un operaio, specificando anche l'*id*, lo ha selezionato per effettuare il controllo. Dopodiché si sospende in attesa che l'operaio termini il controllo, mediane una *wait* sulla variabile condizione **attesa_termine**.

Risvegliato nuovamente da parte del thread *Operaio*, stampa su standard output la conferma del termine dell'operazione di controllo e termina la sua esecuzione.

```
1   void AUTO_ENTRA(int id, int tipo_operazione)
2   {
3       pthread_mutex_lock(&mutex);
4
5       if (tipo_operazione == 0)
6       {
7           printf("AUTO-[Thread%d e identificatore %lu] devo effettuare il
                controllo per il BOLLINO BLU \n", id, pthread_self());
8
9           /* mi metto in coda per i controlli dei bollini blu  */
10          coda_bollino_blu[contatore_bollino_blu] = id;
11          contatore_bollino_blu++;
12      }
13      else
14      {
15          printf("AUTO-[Thread%d e identificatore %lu] devo effettuare il
                controllo per il TAGLIANDO \n", id, pthread_self());
16
17          /* mi metto in coda per i controlli dei tagliandi  */
18          coda_tagliando[contatore_tagliando] = id;
19          contatore_tagliando++;
20      }
21
22      /* sveglio gli operai in attesa per poter essere servito */
23      pthread_cond_broadcast(&auto_in_coda);
24
25      while(automobili[id - NUM_THREADS_OPERAI] == AUTO_NON_SERVITA)
26      {
27          /* mi metto in attesa che un operaio effettui il controllo */
28          pthread_cond_wait(&attesa_controllo, &mutex);
29      }
30
31      printf("AUTO-[Thread%d e identificatore %lu] l'operario con id %d ha
            appena preso in carico il mio controllo\n", id, pthread_self(),
```

```
            automobili[id - NUM_THREADS_OPERAI]);
32
33          while (automobili[id - NUM_THREADS_OPERAI] != AUTO_SERVITA)
34          {
35              /* il controllo dell'auto e' appena iniziato, attendo il termine
                    */
36              pthread_cond_wait(&attesa_termine, &mutex);
37          }
38
39          pthread_mutex_unlock(&mutex);
40      }
```

Codice 3.1: Procedure Entry AUTO_ENTRA del problema dell'Officina

3.3.2 Schema thread Operaio

Lo schema del thread *Operaio* è composto da un ciclo infinito in cui l'operaio ripete le seguenti operazioni:

- **INIZIA_CONTROLLO**, procedure entry in cui l'operaio si mette a disposizione per eseguire un controllo
- simulazione dell'operazione di controllo mediante una sleep
- **FINE_CONTROLLO**, procedure entry in cui l'operaio notifica l'auto che ha terminato il controllo

```
1       while(1)
2       {
3           INIZIA_CONTROLLO(*pi, &durata, 1, &id_auto);
4
5           /* effettuo il controllo */
6           sleep(durata);
7
8           FINE_CONTROLLO(*pi, 1, &id_auto);
9       }
```

All'interno della procedure entry **INIZIA_CONTROLLO**, realizzata mediante l'utilizzo del semaforo di mutua esclusione **mutex**, l'operaio si sospende, mediante una *wait* sulla variabile condizione **auto_in_coda**, in attesa che un thread *Auto* si inserisca in una delle due code.

Risvegliato da parte di un thread *Auto*, in base alla sua tipologia di operaio, valore che viene passato come parametro alla funzione, può effettuare due operazioni:

- **OPERAIO_TIPO_0**: controlla prima se è presente un auto nella coda dedicata ai controlli per il bollino blu ed in caso contrario controlla la coda dei controlli per il tagliando.
- **OPERAIO_TIPO_1**: controlla, soltanto, se è presente un auto nella coda dedicata ai controlli per il tagliando.

Successivamente inserisce il proprio *id* all'interno dell'array **automobili** con indice l'*id* del thread *Auto*. Calcola la durata dell'operazione di controllo in modo random e notifica l'auto di aver appena iniziato il controllo, mediante una *signal* in *broadcast* sulla variabile condizione **attesa_controllo**. Terminando così la procedure entry **INIZIO_CONTROLLO**.

```
1   void INIZIA_CONTROLLO(int id, int *durata, int tipo, int *id_auto)
2   {
3       pthread_mutex_lock(&mutex);
4
5       int i; /* variabile contatore utilizzata per scorrere le code delle
            auto */
6
7       if (tipo == 0)
8       {
9           /* attendo che ci sia un auto in coda */
10          while (contatore_bollino_blu == 0 && contatore_tagliando == 0) /*
                e' necessario per il corretto funzionamento di questa
               soluzione, utilizzare un while per verificare nuovamente la
               condizione */
11          {
12              printf("OPERAIO_TIPO_%d-[Thread%d e identificatore %lu] nessun
                    auto e' in coda, mi sospendo\n", tipo, id, pthread_self())
                   ;
13              pthread_cond_wait(&auto_in_coda, &mutex);
```

```
14              }
15
16              /* verifico il tipo di controllo che devo effettuare, dando
                   precedenza al controllo per il bollino blu */
17              if (contatore_bollino_blu > 0)
18              {
19                  printf("OPERAIO_TIPO_%d-[Thread%d e identificatore %lu] auto
                       con tipologia BOLLINO BLU in coda, la rimuovo dalla coda\n"
                       , tipo, id, pthread_self());
20
21                  /* prendo in carica la prima auto che c'e' in coda */
22                  *id_auto = coda_bollino_blu[0];
23                  automobili[*id_auto - NUM_THREADS_OPERAI] = id;
24
25                  /* rimuovo l'auto dalla coda */
26                  for (i = 0; i < contatore_bollino_blu - 1; i++)
27                      coda_bollino_blu[i] = coda_bollino_blu[i+1];
28
29                  coda_bollino_blu[contatore_bollino_blu -1] = -1;
30                  contatore_bollino_blu--;
31
32                  /* calcolo la durata del controllo */
33                  *durata = mia_random(MAX_DURATA_OPERAZIONE/2);    /* i
                       controlli per il bollino blu sono piu' veloci rispetto a
                       quello del tagliando */
34
35                  printf("OPERAIO_TIPO_%d-[Thread%d e identificatore %lu] prendo
                        in carico l'auto con id %d per un controllo di BOLLINO BLU
                        di durata %d secondi\n", tipo, id, pthread_self(), *
                       id_auto, *durata);
36              }
37              else
38              {
39                  printf("OPERAIO_TIPO_%d-[Thread%d e identificatore %lu] auto
                       con tipologia TAGLIANDO in coda, la rimuovo dalla coda\n",
                       tipo, id, pthread_self());
```

```
40
41              /* prendo in carica la prima auto che c'e' in coda */
42              *id_auto = coda_tagliando[0];
43              automobili[*id_auto - NUM_THREADS_OPERAI] = id;
44
45              /* rimuovo l'auto dalla coda */
46              for (i = 0; i < contatore_tagliando - 1; i++)
47                  coda_tagliando[i] = coda_tagliando[i+1];
48
49              coda_tagliando[contatore_tagliando -1] = -1;
50              contatore_tagliando--;
51
52              /* calcolo la durata del controllo */
53              *durata = mia_random(MAX_DURATA_OPERAZIONE);    /* i controlli
                    per il bollino blu sono piu' veloci rispetto a quello del
                    tagliando */
54
55              printf("OPERAIO_TIPO_%d-[Thread%d e identificatore %lu] prendo
                     in carico l'auto con id %d per un controllo di TAGLIANDO
                    di durata %d secondi\n", tipo, id, pthread_self(), *id_auto
                    , *durata);
56          }
57      }
58      else
59      {
60          /* attendo che ci sia un auto in coda */
61          while (contatore_tagliando == 0)    /* e' necessario per il
                corretto funzionamento di questa soluzione, utilizzare un while
                 per verificare nuovamente la condizione */
62          {
63              printf("OPERAIO_TIPO_%d-[Thread%d e identificatore %lu] nessun
                     auto e' in coda, mi sospendo\n", tipo, id, pthread_self())
                    ;
64              pthread_cond_wait(&auto_in_coda, &mutex);
65          }
66
```

```
67              printf("OPERAIO_TIPO_%d-[Thread%d e identificatore %lu] auto con
                    tipologia TAGLIANDO in coda, la rimuovo dalla coda\n", tipo, id
                    , pthread_self());
68
69              /* prendo in carica la prima auto che c'e' in coda */
70              *id_auto = coda_tagliando[0];
71              automobili[*id_auto - NUM_THREADS_OPERAI] = id;
72
73              /* rimuovo l'auto dalla coda */
74              for (i = 0; i < contatore_tagliando - 1; i++)
75                  coda_tagliando[i] = coda_tagliando[i+1];
76
77              coda_tagliando[contatore_tagliando -1] = -1;
78              contatore_tagliando--;
79
80              /* calcolo la durata del controllo */
81              *durata = mia_random(MAX_DURATA_OPERAZIONE);    /* i controlli per
                    il bollino blu sono piu' veloci rispetto a quello del tagliando
                     */
82              printf("OPERAIO_TIPO_%d-[Thread%d e identificatore %lu] prendo in
                    carico l'auto con id %d per un controllo di TAGLIANDO di durata
                     %d secondi\n", tipo, id, pthread_self(), *id_auto, *durata);
83          }
84
85          /* notifico l'auto di aver appena iniziato il controllo */
86          pthread_cond_broadcast(&attesa_controllo);
87
88          pthread_mutex_unlock(&mutex);
89      }
```

Codice 3.2: Procedure Entry INIZIA_CONTROLLO del problema dell'Officina

All'interno della procedure entry **FINE_CONTROLLO**, realizzata mediante l'utilizzo del semaforo di mutua esclusione **mutex**, l'operaio modifica lo stato dell'auto, all'interno dell'array **automobili**, in *AUTO_SERVITA*. Dopodiché notifica l'auto del termine del controllo, mediante l'uso di una *signal* in *broadcast* sulla variabile condizione **attesa_termine** e termina la procedure entry.

```
1   void FINE_CONTROLLO(int id, int tipo, int id_auto)
2   {
3       pthread_mutex_lock(&mutex);
4
5       printf("OPERAIO_TIPO_%d-[Thread%d e identificatore %lu] controllo sull
            'auto %d TERMINATO\n", tipo, id, pthread_self(), id_auto);
6
7       /* modifico lo stato dell'auto in controllo effettuato (-2) */
8       automobili[id_auto - NUM_THREADS_OPERAI] = AUTO_SERVITA;
9
10      /* notifico l'auto del completamento del controllo */
11      pthread_cond_broadcast(&attesa_termine);
12
13      pthread_mutex_unlock(&mutex);
14  }
```

Codice 3.3: Procedure Entry FINE_CONTROLLO del problema dell'Officina

3.4 Esempio di Funzionamento

L'esecuzione del programma avviene mediante l'utilizzo di tre argomenti, rispettivamente sono:

1. *NUMERO_OPERAI*
2. *NUMERO_AUTO_BOLLINO_BLU*
3. *NUMERO_AUTO_TAGLIANDO*

Una tipica invocazione del programma è la seguente:

```
1   $ ./officina 6 6 7
```

Codice 3.4: Esempio di invocazione del programma del problema dell'Officina

In questo caso ci saranno sei operai, di cui quattro di tipo 0 e due si tipo 1, sei auto per il controllo per il bollino blu e sette auto per il controllo per il tagliando. Il numero totale di auto, dunque è di 13.

```
1  Numero totale OPERAI: 6
2  Numero AUTO_BOLLINO_BLU: 6
3  Numero AUTO_TAGLIANDO: 7
4  Numero totale di AUTO: 13
5  Numero Operai Tipo 0: 4
6  Numero Operai Tipo 1: 2
7  Sto per creare il thread OPERAIO_TIPO_0 0-esimo
8  SONO IL MAIN e ho creato il Pthread OPERAIO_TIPO_0 0-esimo con id
       =140009119594240
9  Sto per creare il thread OPERAIO_TIPO_0 1-esimo
10 OPERAIO_TIPO_0-[Thread0 e identificatore 140009119594240] STO ARRIVANDO
11 OPERAIO_TIPO_0-[Thread0 e identificatore 140009119594240] nessun auto e'
       in coda, mi sospendo
12 SONO IL MAIN e ho creato il Pthread OPERAIO_TIPO_0 1-esimo con id
       =140009111201536
13 Sto per creare il thread OPERAIO_TIPO_0 2-esimo
14 OPERAIO_TIPO_0-[Thread1 e identificatore 140009111201536] STO ARRIVANDO
15 OPERAIO_TIPO_0-[Thread1 e identificatore 140009111201536] nessun auto e'
       in coda, mi sospendo
16 SONO IL MAIN e ho creato il Pthread OPERAIO_TIPO_0 2-esimo con id
       =140009102808832
17 Sto per creare il thread OPERAIO_TIPO_0 3-esimo
18 OPERAIO_TIPO_0-[Thread2 e identificatore 140009102808832] STO ARRIVANDO
19 OPERAIO_TIPO_0-[Thread2 e identificatore 140009102808832] nessun auto e'
       in coda, mi sospendo
20 SONO IL MAIN e ho creato il Pthread OPERAIO_TIPO_0 3-esimo con id
       =140008887416576
21 Sto per creare il thread OPERAIO_TIPO_1 4-esimo
22 OPERAIO_TIPO_0-[Thread3 e identificatore 140008887416576] STO ARRIVANDO
23 OPERAIO_TIPO_0-[Thread3 e identificatore 140008887416576] nessun auto e'
       in coda, mi sospendo
24 SONO IL MAIN e ho creato il Pthread OPERAIO_TIPO_1 4-esimo con id
       =140008753198848
25 Sto per creare il thread OPERAIO_TIPO_1 5-esimo
26 OPERAIO_TIPO_1-[Thread4 e identificatore 140008753198848] STO ARRIVANDO
27 OPERAIO_TIPO_1-[Thread4 e identificatore 140008753198848] nessun auto e'
```

```
    in coda, mi sospendo
28  SONO IL MAIN e ho creato il Pthread OPERAIO_TIPO_1 5-esimo con id
    =140008879023872
29  OPERAIO_TIPO_1-[Thread5 e identificatore 140008879023872] STO ARRIVANDO
30  OPERAIO_TIPO_1-[Thread5 e identificatore 140008879023872] nessun auto e'
    in coda, mi sospendo
31  Sto per creare il thread AUTO 6-esimo
32  SONO IL MAIN e ho creato il Pthread AUTO 6-esimo con id=140008870631168
33  Sto per creare il thread AUTO 7-esimo
34  AUTO-[Thread6 e identificatore 140008870631168] STO ARRIVANDO
35  AUTO-[Thread6 e identificatore 140008870631168] devo effettuare il
    controllo per il TAGLIANDO
36  SONO IL MAIN e ho creato il Pthread AUTO 7-esimo con id=140008862238464
37  Sto per creare il thread AUTO 8-esimo
38  OPERAIO_TIPO_0-[Thread0 e identificatore 140009119594240] auto con
    tipologia TAGLIANDO in coda, la rimuovo dalla coda
39  OPERAIO_TIPO_0-[Thread0 e identificatore 140009119594240] prendo in carico
     l'auto con id 6 per un controllo di TAGLIANDO di durata 5 secondi
40  AUTO-[Thread7 e identificatore 140008862238464] STO ARRIVANDO
41  SONO IL MAIN e ho creato il Pthread AUTO 8-esimo con id=140008853845760
42  Sto per creare il thread AUTO 9-esimo
43  AUTO-[Thread6 e identificatore 140008870631168] l'operario con id 0 ha
    appena preso in carico il mio controllo
44  AUTO-[Thread8 e identificatore 140008853845760] STO ARRIVANDO
45  OPERAIO_TIPO_0-[Thread2 e identificatore 140009102808832] nessun auto e'
    in coda, mi sospendo
46  OPERAIO_TIPO_0-[Thread3 e identificatore 140008887416576] nessun auto e'
    in coda, mi sospendo
47  AUTO-[Thread9 e identificatore 140008845453056] STO ARRIVANDO
48  OPERAIO_TIPO_1-[Thread4 e identificatore 140008753198848] nessun auto e'
    in coda, mi sospendo
49  SONO IL MAIN e ho creato il Pthread AUTO 9-esimo con id=140008845453056
50  OPERAIO_TIPO_1-[Thread5 e identificatore 140008879023872] nessun auto e'
    in coda, mi sospendo
51  OPERAIO_TIPO_0-[Thread1 e identificatore 140009111201536] nessun auto e'
    in coda, mi sospendo
```

```
52  AUTO-[Thread7 e identificatore 140008862238464] devo effettuare il
        controllo per il TAGLIANDO
53  OPERAIO_TIPO_0-[Thread3 e identificatore 140008887416576] auto con
        tipologia TAGLIANDO in coda, la rimuovo dalla coda
54  OPERAIO_TIPO_0-[Thread3 e identificatore 140008887416576] prendo in carico
         l'auto con id 7 per un controllo di TAGLIANDO di durata 4 secondi
55  Sto per creare il thread AUTO 10-esimo
56  AUTO-[Thread7 e identificatore 140008862238464] l'operario con id 3 ha
        appena preso in carico il mio controllo
57  OPERAIO_TIPO_1-[Thread4 e identificatore 140008753198848] nessun auto e'
        in coda, mi sospendo
58  OPERAIO_TIPO_1-[Thread5 e identificatore 140008879023872] nessun auto e'
        in coda, mi sospendo
59  SONO IL MAIN e ho creato il Pthread AUTO 10-esimo con id=140008837060352
60  Sto per creare il thread AUTO 11-esimo
61  AUTO-[Thread8 e identificatore 140008853845760] devo effettuare il
        controllo per il TAGLIANDO
62  OPERAIO_TIPO_1-[Thread4 e identificatore 140008753198848] auto con
        tipologia TAGLIANDO in coda, la rimuovo dalla coda
63  OPERAIO_TIPO_1-[Thread4 e identificatore 140008753198848] prendo in carico
         l'auto con id 8 per un controllo di TAGLIANDO di durata 4 secondi
64  SONO IL MAIN e ho creato il Pthread AUTO 11-esimo con id=140008744806144
65  Sto per creare il thread AUTO 12-esimo
66  SONO IL MAIN e ho creato il Pthread AUTO 12-esimo con id=140008736413440
67  Sto per creare il thread AUTO 13-esimo
68  SONO IL MAIN e ho creato il Pthread AUTO 13-esimo con id=140008728020736
69  Sto per creare il thread AUTO 14-esimo
70  SONO IL MAIN e ho creato il Pthread AUTO 14-esimo con id=140008719628032
71  Sto per creare il thread AUTO 15-esimo
72  SONO IL MAIN e ho creato il Pthread AUTO 15-esimo con id=140008711235328
73  Sto per creare il thread AUTO 16-esimo
74  SONO IL MAIN e ho creato il Pthread AUTO 16-esimo con id=140008702842624
75  Sto per creare il thread AUTO 17-esimo
76  SONO IL MAIN e ho creato il Pthread AUTO 17-esimo con id=140008216327936
77  Sto per creare il thread AUTO 18-esimo
78  SONO IL MAIN e ho creato il Pthread AUTO 18-esimo con id=140008207935232
```

```
79  AUTO-[Thread10 e identificatore 140008837060352] STO ARRIVANDO
80  OPERAIO_TIPO_0-[Thread1 e identificatore 140009111201536] nessun auto e'
        in coda, mi sospendo
81  AUTO-[Thread10 e identificatore 140008837060352] devo effettuare il
        controllo per il BOLLINO BLU
82  OPERAIO_TIPO_1-[Thread5 e identificatore 140008879023872] nessun auto e'
        in coda, mi sospendo
83  AUTO-[Thread8 e identificatore 140008853845760] l'operario con id 4 ha
        appena preso in carico il mio controllo
84  OPERAIO_TIPO_0-[Thread2 e identificatore 140009102808832] auto con
        tipologia BOLLINO BLU in coda, la rimuovo dalla coda
85  OPERAIO_TIPO_0-[Thread2 e identificatore 140009102808832] prendo in carico
         l'auto con id 10 per un controllo di BOLLINO BLU di durata 1 secondi
86  AUTO-[Thread18 e identificatore 140008207935232] STO ARRIVANDO
87  AUTO-[Thread18 e identificatore 140008207935232] devo effettuare il
        controllo per il TAGLIANDO
88  OPERAIO_TIPO_0-[Thread1 e identificatore 140009111201536] auto con
        tipologia TAGLIANDO in coda, la rimuovo dalla coda
89  OPERAIO_TIPO_0-[Thread1 e identificatore 140009111201536] prendo in carico
         l'auto con id 18 per un controllo di TAGLIANDO di durata 3 secondi
90  AUTO-[Thread18 e identificatore 140008207935232] l'operario con id 1 ha
        appena preso in carico il mio controllo
91  AUTO-[Thread9 e identificatore 140008845453056] devo effettuare il
        controllo per il TAGLIANDO
92  AUTO-[Thread10 e identificatore 140008837060352] l'operario con id 2 ha
        appena preso in carico il mio controllo
93  OPERAIO_TIPO_1-[Thread5 e identificatore 140008879023872] auto con
        tipologia TAGLIANDO in coda, la rimuovo dalla coda
94  OPERAIO_TIPO_1-[Thread5 e identificatore 140008879023872] prendo in carico
         l'auto con id 9 per un controllo di TAGLIANDO di durata 6 secondi
95  AUTO-[Thread9 e identificatore 140008845453056] l'operario con id 5 ha
        appena preso in carico il mio controllo
96  AUTO-[Thread11 e identificatore 140008744806144] STO ARRIVANDO
97  AUTO-[Thread11 e identificatore 140008744806144] devo effettuare il
        controllo per il BOLLINO BLU
98  AUTO-[Thread16 e identificatore 140008702842624] STO ARRIVANDO
```

```
99  AUTO-[Thread16 e identificatore 140008702842624] devo effettuare il
        controllo per il TAGLIANDO
100 AUTO-[Thread12 e identificatore 140008736413440] STO ARRIVANDO
101 AUTO-[Thread12 e identificatore 140008736413440] devo effettuare il
        controllo per il TAGLIANDO
102 AUTO-[Thread14 e identificatore 140008719628032] STO ARRIVANDO
103 AUTO-[Thread14 e identificatore 140008719628032] devo effettuare il
        controllo per il BOLLINO BLU
104 AUTO-[Thread17 e identificatore 140008216327936] STO ARRIVANDO
105 AUTO-[Thread17 e identificatore 140008216327936] devo effettuare il
        controllo per il BOLLINO BLU
106 AUTO-[Thread15 e identificatore 140008711235328] STO ARRIVANDO
107 AUTO-[Thread15 e identificatore 140008711235328] devo effettuare il
        controllo per il BOLLINO BLU
108 AUTO-[Thread13 e identificatore 140008728020736] STO ARRIVANDO
109 AUTO-[Thread13 e identificatore 140008728020736] devo effettuare il
        controllo per il BOLLINO BLU
110 OPERAIO_TIPO_0-[Thread2 e identificatore 140009102808832] controllo sull'
        auto 10 TERMINATO
111 OPERAIO_TIPO_0-[Thread2 e identificatore 140009102808832] auto con
        tipologia BOLLINO BLU in coda, la rimuovo dalla coda
112 OPERAIO_TIPO_0-[Thread2 e identificatore 140009102808832] prendo in carico
         l'auto con id 11 per un controllo di BOLLINO BLU di durata 2 secondi
113 AUTO-[Thread10 e identificatore 140008837060352] controllo terminato, VADO
         A CASA
114 AUTO-[Thread11 e identificatore 140008744806144] l'operario con id 2 ha
        appena preso in carico il mio controllo
115 OPERAIO_TIPO_0-[Thread2 e identificatore 140009102808832] controllo sull'
        auto 11 TERMINATO
116 OPERAIO_TIPO_0-[Thread2 e identificatore 140009102808832] auto con
        tipologia BOLLINO BLU in coda, la rimuovo dalla coda
117 OPERAIO_TIPO_0-[Thread2 e identificatore 140009102808832] prendo in carico
         l'auto con id 14 per un controllo di BOLLINO BLU di durata 1 secondi
118 AUTO-[Thread14 e identificatore 140008719628032] l'operario con id 2 ha
        appena preso in carico il mio controllo
119 AUTO-[Thread11 e identificatore 140008744806144] controllo terminato, VADO
```

```
    A CASA
120 OPERAIO_TIPO_0-[Thread1 e identificatore 140009111201536] controllo sull'
    auto 18 TERMINATO
121 OPERAIO_TIPO_0-[Thread1 e identificatore 140009111201536] auto con
    tipologia BOLLINO BLU in coda, la rimuovo dalla coda
122 OPERAIO_TIPO_0-[Thread1 e identificatore 140009111201536] prendo in carico
    l'auto con id 17 per un controllo di BOLLINO BLU di durata 1 secondi
123 AUTO-[Thread18 e identificatore 140008207935232] controllo terminato, VADO
    A CASA
124 AUTO-[Thread17 e identificatore 140008216327936] l'operario con id 1 ha
    appena preso in carico il mio controllo
125 OPERAIO_TIPO_0-[Thread3 e identificatore 140008887416576] controllo sull'
    auto 7 TERMINATO
126 OPERAIO_TIPO_0-[Thread3 e identificatore 140008887416576] auto con
    tipologia BOLLINO BLU in coda, la rimuovo dalla coda
127 OPERAIO_TIPO_0-[Thread3 e identificatore 140008887416576] prendo in carico
    l'auto con id 15 per un controllo di BOLLINO BLU di durata 2 secondi
128 AUTO-[Thread15 e identificatore 140008711235328] l'operario con id 3 ha
    appena preso in carico il mio controllo
129 OPERAIO_TIPO_1-[Thread4 e identificatore 140008753198848] controllo sull'
    auto 8 TERMINATO
130 OPERAIO_TIPO_1-[Thread4 e identificatore 140008753198848] auto con
    tipologia TAGLIANDO in coda, la rimuovo dalla coda
131 OPERAIO_TIPO_1-[Thread4 e identificatore 140008753198848] prendo in carico
    l'auto con id 16 per un controllo di TAGLIANDO di durata 3 secondi
132 AUTO-[Thread7 e identificatore 140008862238464] controllo terminato, VADO
    A CASA
133 AUTO-[Thread8 e identificatore 140008853845760] controllo terminato, VADO
    A CASA
134 AUTO-[Thread16 e identificatore 140008702842624] l'operario con id 4 ha
    appena preso in carico il mio controllo
135 OPERAIO_TIPO_0-[Thread2 e identificatore 140009102808832] controllo sull'
    auto 14 TERMINATO
136 OPERAIO_TIPO_0-[Thread2 e identificatore 140009102808832] auto con
    tipologia BOLLINO BLU in coda, la rimuovo dalla coda
137 OPERAIO_TIPO_0-[Thread2 e identificatore 140009102808832] prendo in carico
```

```
        l'auto con id 13 per un controllo di BOLLINO BLU di durata 3 secondi
138     AUTO-[Thread13 e identificatore 140008728020736] l'operario con id 2 ha
            appena preso in carico il mio controllo
139     AUTO-[Thread14 e identificatore 140008719628032] controllo terminato, VADO
            A CASA
140     OPERAIO_TIPO_0-[Thread1 e identificatore 140009111201536] controllo sull'
            auto 17 TERMINATO
141     OPERAIO_TIPO_0-[Thread1 e identificatore 140009111201536] auto con
            tipologia TAGLIANDO in coda, la rimuovo dalla coda
142     OPERAIO_TIPO_0-[Thread1 e identificatore 140009111201536] prendo in carico
            l'auto con id 12 per un controllo di TAGLIANDO di durata 1 secondi
143     AUTO-[Thread12 e identificatore 140008736413440] l'operario con id 1 ha
            appena preso in carico il mio controllo
144     AUTO-[Thread17 e identificatore 140008216327936] controllo terminato, VADO
            A CASA
145     OPERAIO_TIPO_0-[Thread0 e identificatore 140009119594240] controllo sull'
            auto 6 TERMINATO
146     OPERAIO_TIPO_0-[Thread0 e identificatore 140009119594240] nessun auto e'
            in coda, mi sospendo
147     AUTO-[Thread6 e identificatore 140008870631168] controllo terminato, VADO
            A CASA
148     Pthread 6-esimo restituisce 6
149     Pthread 7-esimo restituisce 7
150     Pthread 8-esimo restituisce 8
151     OPERAIO_TIPO_0-[Thread1 e identificatore 140009111201536] controllo sull'
            auto 12 TERMINATO
152     OPERAIO_TIPO_0-[Thread1 e identificatore 140009111201536] nessun auto e'
            in coda, mi sospendo
153     AUTO-[Thread12 e identificatore 140008736413440] controllo terminato, VADO
            A CASA
154     OPERAIO_TIPO_0-[Thread3 e identificatore 140008887416576] controllo sull'
            auto 15 TERMINATO
155     OPERAIO_TIPO_0-[Thread3 e identificatore 140008887416576] nessun auto e'
            in coda, mi sospendo
156     AUTO-[Thread15 e identificatore 140008711235328] controllo terminato, VADO
            A CASA
```

```
157 OPERAIO_TIPO_1-[Thread5 e identificatore 140008879023872] controllo sull'
        auto 9 TERMINATO
158 OPERAIO_TIPO_1-[Thread5 e identificatore 140008879023872] nessun auto e'
        in coda, mi sospendo
159 AUTO-[Thread9 e identificatore 140008845453056] controllo terminato, VADO
        A CASA
160 Pthread 9-esimo restituisce 9
161 Pthread 10-esimo restituisce 10
162 Pthread 11-esimo restituisce 11
163 Pthread 12-esimo restituisce 12
164 OPERAIO_TIPO_1-[Thread4 e identificatore 140008753198848] controllo sull'
        auto 16 TERMINATO
165 OPERAIO_TIPO_1-[Thread4 e identificatore 140008753198848] nessun auto e'
        in coda, mi sospendo
166 AUTO-[Thread16 e identificatore 140008702842624] controllo terminato, VADO
         A CASA
167 OPERAIO_TIPO_0-[Thread2 e identificatore 140009102808832] controllo sull'
        auto 13 TERMINATO
168 OPERAIO_TIPO_0-[Thread2 e identificatore 140009102808832] nessun auto e'
        in coda, mi sospendo
169 AUTO-[Thread13 e identificatore 140008728020736] controllo terminato, VADO
         A CASA
170 Pthread 13-esimo restituisce 13
171 Pthread 14-esimo restituisce 14
172 Pthread 15-esimo restituisce 15
173 Pthread 16-esimo restituisce 16
174 Pthread 17-esimo restituisce 17
175 Pthread 18-esimo restituisce 18
176 Contatore coda auto bollino blu: 0
177 Contatore coda auto tagliando: 0
178 Stato automobili: [ -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 ]
```

Codice 3.5: Output del programma del problema dell'Officina

3.5 Possibili problemi di Starvation o Deadlock

La soluzione presentata è stata realizzata evitando qualunque problema di Deadlock, dunque non sono presenti problemi di questo genere. Per quanto riguarda i problemi di Starvation, si presenta tale problema in invocazioni di questo tipo:

```
1    $ ./officina 6 6 0
```

In cui non sono presenti auto per la tipologia tagliando, ma sono comunque presenti, per via della specifica, due operai di tipo 0, i quali sono autorizzati soltanto ad eseguire controlli per i tagliandi. Per tanto si verificherà una situazione di Starvation dei thread *OPERAI_TIPO_0*, i quali rimarranno continuamente in attesa di un auto che vorrà effettuare un controllo per il tagliando, ma che non si presenterà mai. Nella soluzione, dato che questa invocazione rimane comunque valida per la soluzione del problema, verrà mostrato solamente un messaggio che indica la presenza di questo avvenimento ed il programma proseguirà la sua esecuzione.

Non sono presenti altri casi di Starvation in quanto, nonostante i thread *OPERAI_TIPO_0* applicano una preferenza alle auto per il controllo per il bollino blu, le automobili in coda per il tagliando verranno comunque servite dai thread *OPERAI_TIPO_1*, non creano situazioni di Starvation per i thread *AUTO*.

4. Vetrina Online

4.1 Descrizione del problema

Una **vetrina online** permette agli **utenti** di ordinare bevande. Le bevande ordinate vengono successivamente spedite agli utenti tramite C **corrieri**. Ogni utente può ordinare da un minimo di 2 ad un massimo di 18 scatoloni di bevande; gli scatoloni ordinati devono comunque essere multipli di 2 (2, 4, 6, 8, ...). La vetrina online gestisce gli ordini e li passa al primo corriere libero, dando la precedenza agli ordini di 18 scatoloni. Una volta avuto l'ordine, il corriere consegna (in un tempo random deciso dal corriere), si fa pagare dall'utente che riceve la merce e rientra.

Si implementi una soluzione usando il costrutto *monitor* per modellare la **vetrina online**, i *processi* per modellare gli **utenti** e i **corrireri**. Nella soluzione si massimizzi l'utilizzo delle risorse. Si discuta se la soluzione proposta può presentare starvation e in caso positivo per quali processi, e si propongano modifiche e/o aggiunte per evitare la starvation.

4.2 Descrizione delle strutture dati

4.2.1 Array di utenti

Per risolvere il problema, utilizzeremo un array di **Utenti** di tipo *int*, di dimensione pari al numero di utenti scelto per l'esecuzione del programma. Sul quale andremo a memorizzare l'id del thread *corriere* che ha preso in carico l'ordine e sta effettuando la spedizione.

```
1    int *utenti;
```

L'array di **Utenti** verrà inizializzato con il valore -1, indicando che tale utente non è stato ancora servito, mentre al termine della spedizione avrà valore -2, indicando che tale utente è stato servito. A tal proposito sono state realizzare le seguenti due macro per rappresentare lo stato dell'ordine di ogni singolo utente:

```
1    #define UTENTE_NON_SERVITO -1
2    #define UTENTE_SERVITO -2
```

4.2.2 Realizzazione delle code

Ogni utente non appena effettua il suo ordine, viene inserito in una delle due code messe a disposizione, in base alle priorità che gli verrà assegnata. La priorità, come da specifica, viene assegnata in base al numero di scatoloni che vengono ordinati. Soltanto se il numero di scatoloni è pari a 18, l'ordine viene considerato ad alta priorità, altrimenti l'ordine avrà una bassa priorità.

L'implementazione delle due code è stata realizzata mediante due array, **coda_prioritaria** e **coda_normale**, di tipo *int* e di dimensioni pari al numero di utenti scelto per l'esecuzione del programma.

Inizializzati a -1, per indicare l'assenza di utenti all'interno della coda, e successivamente contenenti l'*id* del thread *Utente* in attesa che il suo ordine venga preso in carico da uno dei corrieri disponibili. Verranno, inoltre, utilizzati due rispettivi contatori, **contatore_coda_prioritaria** e **contatore_coda_normale**, inizializzati a 0, per indicare il numero di thread *Utenti* attualmente in coda.

```
1    int *coda_prioritaria;
2    int contatore_coda_prioritaria;
3
4    int *coda_normale;
5    int contatore_coda_normale;
```

4.2.3 Implementazione del costrutto monitor

La protezione di tali strutture dati, verrà effettuata dal costrutto *monitor* realizzato mediante l'utilizzo del seguente semaforo binario di mutua esclusione.

```
1    pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
```

4.2.4 Variabili condizione

Per la comunicazione tra i thread di tipo *Utente* e i thread di tipo *Corrieri*, utilizzeremo le seguenti variabili condizione:

```
1    pthread_cond_t utenti_in_coda = PTHREAD_COND_INITIALIZER;
2    pthread_cond_t attesa_selezione = PTHREAD_COND_INITIALIZER;
```

```
3     pthread_cond_t attesa_arrivo = PTHREAD_COND_INITIALIZER;
```

Dove **utenti_in_coda** rappresenta la variabile condizione in cui i thread *Corrieri* si bloccano in attesa che un utente si inserisca in una delle due code, pronto per poter essere servito da un corriere. Mentre la variabile condizione **attesa_selezione** serve ai thread *Utenti* per bloccarsi in attesa che un thread *Corriere* selezioni il suo ordine e parta per la spedizione. Infine, **attesa_arrivo** è la variabile condizione che serve ai thread *Utenti*, una volta che un thread *Corriere* ha selezionato l'ordine ed è partito per la spedione, per bloccarsi in attesa che il corriere arrivi a destinazione.

4.3 Implementazione della soluzione

4.3.1 Schema thread Utente

Lo schema del thread *Utente* è composto dalla generazione random del numero di scatoloni da ordinare e successivamente dalla chiamata della procedure entry **Ordina**.

```
1     /* genero il numero di scatoloni da ordinare */
2     scatoloni = mia_random(MAX_SCATOLONI);
3
4     if ((scatoloni%2) != 0)    /* mi assicuro che il numero generato sia un
          multiplo di 2 */
5         scatoloni++;
6
7     /* chiamo la funzione ORDINA */
8     ORDINA(*pi, scatoloni);
```

All'interno della procedure entry **Ordina**, realizzata mediante l'utilizzo del semaforo di mutua esclusione **mutex**, viene selezionata la priorità dell'ordine, in base al numero di scatoloni generato. A tal proposito sono state definite le seguenti due macro per la rappresentazione della priorità:

```
1     #define ALTA_PRIORITA 1
2     #define BASSA_PRIORITA 0
```

Una volta selezionata la priorità, l'utente inserisce il proprio *id* nella rispettiva coda di priorità ed effettua una *signal* in *broadcast* sulla variabile condizione **utenti_in_coda**, risvegliando tutti i thread *Corriere* in attesa.

Successivamente, viene verificata la condizione di selezione dell'ordine da parte di un thread *Corriere* e in caso contrario si sospende effettuando una *wait* sulla variabile condizione **attesa_selezione**, in attesa che un corriere selezioni l'ordine da consegnare.

Risvegliato da parte di un thread *Corriere*, l'utente stampa su standard output che un corriere, specificando anche l'*id*, ha preso in carico il suo ordine. Dopodiché si sospende in attesa che il corriere arrivi a destinazione, mediante una *wait* sulla variabile condizione **attesa_arrivo**.

Risvegliato nuovamente da parte del thread *Corriere*, stampa su standard output la conferma della consegna dell'ordine, paga la spedizione e termina la sua esecuzione.

```
1   void ORDINA(int id, int scatoloni)
2   {
3       pthread_mutex_lock(&mutex);
4
5       int priorita;
6
7       priorita = BASSA_PRIORITA;
8
9       if (scatoloni == 18)
10          priorita = ALTA_PRIORITA;
11
12      if (priorita == ALTA_PRIORITA)
13      {
14          printf("UTENTE-[Thread%d e identificatore %lu] voglio ordinare %d
                scatoloni, mi inserisco nella coda PRIORITARIA \n", id,
                pthread_self(), scatoloni);
15
16          coda_prioritaria[contatore_coda_prioritaria] = id;
17          contatore_coda_prioritaria++;
18      }
19      else
20      {
21          printf("UTENTE-[Thread%d e identificatore %lu] voglio ordinare %d
                scatoloni, mi inserisco nella coda NORMALE \n", id,
                pthread_self(), scatoloni);
22
```

```
23              coda_normale[contatore_coda_normale] = id;
24              contatore_coda_normale++;
25          }
26
27          /* sveglio i corrieri in attesa, per poter essere servito */
28          pthread_cond_broadcast(&utenti_in_coda);
29
30          while(utenti[id - NUM_THREADS_CORRIERI] == UTENTE_NON_SERVITO)
31          {
32              /* mi sospendo in attesa che un corriere selezioni il mio ordine
                    */
33              pthread_cond_wait(&attesa_selezione, &mutex);
34          }
35
36          printf("UTENTE-[Thread%d e identificatore %lu] il corriere con id %d
                ha appena preso in carico il mio ordine\n", id, pthread_self(),
                utenti[id - NUM_THREADS_CORRIERI]);
37
38          while (utenti[id - NUM_THREADS_CORRIERI] != UTENTE_SERVITO)
39          {
40              /* il corriere e' appena partito, mi sospendo in attesa del suo
                    arrivo */
41              pthread_cond_wait(&attesa_arrivo, &mutex);
42          }
43
44          printf("UTENTE-[Thread%d e identificatore %lu] il corriere e' appena
                arrivato, pago e termino l'ordine\n", id, pthread_self());
45
46          pthread_mutex_unlock(&mutex);
47      }
```

Codice 4.1: Procedure Entry ORDINA del problema della Vetrina Online

4.3.2 Schema thread Corriere

Lo schema del thread *Corriere* è composto da un ciclo infinito in cui il corriere ripete le seguenti operazioni:

- **PARTI**, procedure entry in cui il corriere si mette a disposizione per la presa in carico di una spedizione
- simulazione del viaggio di andata mediante una sleep
- **CONSEGNA**, procedure entry in cui il corriere consegna l'ordine al cliente
- simulazione del viaggio di ritorno mediante una sleep

```
1   while (1)
2   {
3       /* mi rendo disponibile per la presa in cario di una spedizione */
4       PARTI(*pi, &id_utente, &durata);
5
6       /* simulo la durata del viaggio mediante una sleep */
7       sleep(durata);
8
9       /* effettuo la consegna dell'ordine al cliente */
10      CONSEGNA(*pi, &id_utente, &durata);
11
12      /* simulo la durata del viaggio di ritorno in negozio mediante una
            sleep */
13      sleep(durata);
14  }
```

All'interno della procedure entry **PARTI**, realizzata mediante l'utilizzo del semaforo di mutua esclusione **mutex**, il corriere si sospende, mediante una *wait* sulla variabile condizione **utenti_in_coda**, in attesa che un thread *Utente* si inserisca in una delle due code.

Risvegliato da parte di un thread *Utente*, controlla prima se è presente un ordine di un utente nella coda prioritaria ed in caso contrario controlla la coda normale. Successivamente inserisce il proprio *id* all'interno dell'array **utenti** con indice l'*id* del thread *Utente*. Calcola la durata della

spedizione in modo random e notifica l'utente di aver preso in carico il suo ordine, mediante una *signal* in *broadcast* sulla variabile condizione **attesa_selezione**. Terminando così la procedure entry **PARTI**.

```
1   void PARTI(int id, int *id_utente, int *durata)
2   {
3       pthread_mutex_lock(&mutex);
4
5       int i;  /* variabile contatore utilizzata per scorrere le code */
6
7       /* attendo che un utente effettui un ordine */
8       while (contatore_coda_prioritaria == 0 && contatore_coda_normale == 0)
9       {
10          printf("CORRIERE-[Thread%d e identificatore %lu] non ci sono
                utenti in coda, mi sospendo\n", id, pthread_self());
11          pthread_cond_wait(&utenti_in_coda, &mutex); /* mi sospendo in
                attesa che un utente si metta in una delle code */
12      }
13
14      if (contatore_coda_prioritaria > 0) /* controllo se ci sono utenti
            nella coda prioritaria */
15      {
16          printf("CORRIERE-[Thread%d e identificatore %lu] un utente e' in
                coda PRIORITARIA, lo rimuovo dalla coda\n", id, pthread_self())
                ;
17
18          /* prendo in carico l'ordine del primo utente che c'e' in coda */
19          *id_utente = coda_prioritaria[0];
20          utenti[*id_utente - NUM_THREADS_CORRIERI] = id;
21
22          /* rimuovo dalla coda l'utente */
23          for (i = 0; i < contatore_coda_prioritaria - 1; i++)
24              coda_prioritaria[i] = coda_prioritaria[i+1];
25
26          coda_prioritaria[contatore_coda_prioritaria -1] = -1;
27          contatore_coda_prioritaria--;
```

```
28          }
29          else    /* non ci sono utenti nella coda prioritaria, allora gli
                utenti sono solo nella coda normale */
30          {
31              printf("CORRIERE-[Thread%d e identificatore %lu] un utente e' in
                    coda NORMALE, lo rimuovo dalla coda\n", id, pthread_self());
32
33              /* prendo in carico l'ordine del primo utente che c'e' in coda */
34              *id_utente = coda_normale[0];
35              utenti[*id_utente - NUM_THREADS_CORRIERI] = id;
36
37              /* rimuovo dalla coda l'utente */
38              for(i = 0; i < contatore_coda_normale - 1; i++)
39                  coda_normale[i] = coda_normale[i+1];
40
41              coda_normale[contatore_coda_normale -1] = -1;
42              contatore_coda_normale--;
43          }
44
45          /* calcolo la durata della spedizione */
46          *durata = mia_random(MAX_DURATA_VIAGGIO);
47
48          printf("CORRIERE-[Thread%d e identificatore %lu] prendo in carico l'
                ordine dell'utente con id %d, la spedizione avra' durata di %d
                secondi\n", id, pthread_self(), *id_utente, *durata);
49
50          /* notifico l'utente di aver preso in carico il suo ordine e che sono
                appena partito */
51          pthread_cond_broadcast(&attesa_selezione);
52
53          pthread_mutex_unlock(&mutex);
54      }
```

Codice 4.2: Procedure Entry PARTI del problema della Vetrina Online

All'interno della procedure entry **CONSEGNA**, realizzata mediante l'utilizzo del semaforo di mutua esclusione **mutex**, il corriere modifica lo stato dell'utente, all'interno dell'array **utenti**, in

UTENTE_SERVITO. Dopodiché notifica l'utente del termine della consegna, mediante l'uso di una *signal* in *broadcast* sulla variabile condizione **attesa_arrivo** e termina la procedure entry.

```
1   void CONSEGNA(int id, int id_utente, int durata)
2   {
3       pthread_mutex_lock(&mutex);
4
5       printf("CORRIERE-[Thread%d e identificatore %lu] ho consegnato l'
            ordine dell'utente %d, sto rientrando (durata rientro: %d secondi)\
            n", id, pthread_self(), id_utente, durata);
6
7       /* modifico lo stato dell'utente in: UTENTE_SERVITO (-2) */
8       utenti[id_utente - NUM_THREADS_CORRIERI] = UTENTE_SERVITO;
9
10      /* notifico l'utente della consegna dell'ordine */
11      pthread_cond_broadcast(&attesa_arrivo);
12
13      pthread_mutex_unlock(&mutex);
14  }
```

Codice 4.3: Procedure Entry CONSEGNA del problema della Vetrina Online

4.4 Esempio di Funzionamento

L'esecuzione del programma avviene mediante l'utilizzo di due argomenti, rispettivamente sono:

1. *NUMERO_CORRIERI*
2. *NUMERO_UTENTI*

Una tipica invocazione del programma è la seguente:

```
1   $ ./vetrinaonline 3 15
```

Codice 4.4: Esempio di invocazione del programma del problema della Vetrina Online

In questo caso ci saranno tre corrieri e quindici utenti che effettuano l'ordine in negozio. Ho scelto quindici utenti per avere maggiore probabilità che ad almeno un utente vengano scelti 18

scatoloni. Possiamo, infatti, notare che alla riga 30 l'*Utente* thread 4 vuole ordinare 18 scatoloni e dalla riga 33 si può notare che il *Corriere* thread 1 si è subito accorto della presenza di un utente nella coda PRIORITARIA e dalla riga successiva, riga 34, prende in carico il suo ordine.

```
1  Numero totale di CORRIERI: 3
2  Numero totale di UTENTI: 15
3  Sto per creare il thread CORRIERE 0-esimo
4  SONO IL MAIN e ho creato il Pthread CORRIERE 0-esimo con id
       =139800251307776
5  Sto per creare il thread CORRIERE 1-esimo
6  CORRIERE-[Thread0 e identificatore 139800251307776] STO ARRIVANDO
7  CORRIERE-[Thread0 e identificatore 139800251307776] non ci sono utenti in
       coda, mi sospendo
8  SONO IL MAIN e ho creato il Pthread CORRIERE 1-esimo con id
       =139800108697344
9  Sto per creare il thread CORRIERE 2-esimo
10 CORRIERE-[Thread1 e identificatore 139800108697344] STO ARRIVANDO
11 SONO IL MAIN e ho creato il Pthread CORRIERE 2-esimo con id
       =139800242915072
12 CORRIERE-[Thread2 e identificatore 139800242915072] STO ARRIVANDO
13 CORRIERE-[Thread1 e identificatore 139800108697344] non ci sono utenti in
       coda, mi sospendo
14 CORRIERE-[Thread2 e identificatore 139800242915072] non ci sono utenti in
       coda, mi sospendo
15 Sto per creare il thread UTENTE 3-esimo
16 SONO IL MAIN e ho creato il Pthread UTENTE 3-esimo con id=139800234522368
17 Sto per creare il thread UTENTE 4-esimo
18 UTENTE-[Thread3 e identificatore 139800234522368] STO ARRIVANDO
19 UTENTE-[Thread3 e identificatore 139800234522368] voglio ordinare 2
       scatoloni, mi inserisco nella coda NORMALE
20 SONO IL MAIN e ho creato il Pthread UTENTE 4-esimo con id=139800226129664
21 Sto per creare il thread UTENTE 5-esimo
22 UTENTE-[Thread4 e identificatore 139800226129664] STO ARRIVANDO
23 CORRIERE-[Thread0 e identificatore 139800251307776] un utente e' in coda
       NORMALE, lo rimuovo dalla coda
24 CORRIERE-[Thread0 e identificatore 139800251307776] prendo in carico l'
```

```
    ordine dell'utente con id 3, la spedizione avra' durata di 3 secondi
25  SONO IL MAIN e ho creato il Pthread UTENTE 5-esimo con id=139800217736960
26  Sto per creare il thread UTENTE 6-esimo
27  UTENTE-[Thread3 e identificatore 139800234522368] il corriere con id 0 ha
        appena preso in carico il mio ordine
28  UTENTE-[Thread5 e identificatore 139800217736960] STO ARRIVANDO
29  UTENTE-[Thread6 e identificatore 139800209344256] STO ARRIVANDO
30  UTENTE-[Thread4 e identificatore 139800226129664] voglio ordinare 18
        scatoloni, mi inserisco nella coda PRIORITARIA
31  SONO IL MAIN e ho creato il Pthread UTENTE 6-esimo con id=139800209344256
32  Sto per creare il thread UTENTE 7-esimo
33  CORRIERE-[Thread1 e identificatore 139800108697344] un utente e' in coda
        PRIORITARIA, lo rimuovo dalla coda
34  CORRIERE-[Thread1 e identificatore 139800108697344] prendo in carico l'
        ordine dell'utente con id 4, la spedizione avra' durata di 4 secondi
35  UTENTE-[Thread4 e identificatore 139800226129664] il corriere con id 1 ha
        appena preso in carico il mio ordine
36  SONO IL MAIN e ho creato il Pthread UTENTE 7-esimo con id=139800200951552
37  Sto per creare il thread UTENTE 8-esimo
38  UTENTE-[Thread5 e identificatore 139800217736960] voglio ordinare 8
        scatoloni, mi inserisco nella coda NORMALE
39  UTENTE-[Thread7 e identificatore 139800200951552] STO ARRIVANDO
40  UTENTE-[Thread6 e identificatore 139800209344256] voglio ordinare 6
        scatoloni, mi inserisco nella coda NORMALE
41  SONO IL MAIN e ho creato il Pthread UTENTE 8-esimo con id=139800192558848
42  Sto per creare il thread UTENTE 9-esimo
43  CORRIERE-[Thread2 e identificatore 139800242915072] un utente e' in coda
        NORMALE, lo rimuovo dalla coda
44  CORRIERE-[Thread2 e identificatore 139800242915072] prendo in carico l'
        ordine dell'utente con id 5, la spedizione avra' durata di 3 secondi
45  UTENTE-[Thread9 e identificatore 139800100304640] STO ARRIVANDO
46  UTENTE-[Thread7 e identificatore 139800200951552] voglio ordinare 12
        scatoloni, mi inserisco nella coda NORMALE
47  SONO IL MAIN e ho creato il Pthread UTENTE 9-esimo con id=139800100304640
48  Sto per creare il thread UTENTE 10-esimo
49  UTENTE-[Thread5 e identificatore 139800217736960] il corriere con id 2 ha
```

```
    appena preso in carico il mio ordine
50  UTENTE-[Thread8 e identificatore 139800192558848] STO ARRIVANDO
51  UTENTE-[Thread8 e identificatore 139800192558848] voglio ordinare 2
        scatoloni, mi inserisco nella coda NORMALE
52  SONO IL MAIN e ho creato il Pthread UTENTE 10-esimo con id=139800091911936
53  Sto per creare il thread UTENTE 11-esimo
54  SONO IL MAIN e ho creato il Pthread UTENTE 11-esimo con id=139800083519232
55  Sto per creare il thread UTENTE 12-esimo
56  UTENTE-[Thread9 e identificatore 139800100304640] voglio ordinare 16
        scatoloni, mi inserisco nella coda NORMALE
57  UTENTE-[Thread11 e identificatore 139800083519232] STO ARRIVANDO
58  UTENTE-[Thread11 e identificatore 139800083519232] voglio ordinare 16
        scatoloni, mi inserisco nella coda NORMALE
59  SONO IL MAIN e ho creato il Pthread UTENTE 12-esimo con id=139800075126528
60  Sto per creare il thread UTENTE 13-esimo
61  UTENTE-[Thread12 e identificatore 139800075126528] STO ARRIVANDO
62  UTENTE-[Thread12 e identificatore 139800075126528] voglio ordinare 14
        scatoloni, mi inserisco nella coda NORMALE
63  SONO IL MAIN e ho creato il Pthread UTENTE 13-esimo con id=139800066733824
64  Sto per creare il thread UTENTE 14-esimo
65  UTENTE-[Thread13 e identificatore 139800066733824] STO ARRIVANDO
66  UTENTE-[Thread13 e identificatore 139800066733824] voglio ordinare 16
        scatoloni, mi inserisco nella coda NORMALE
67  SONO IL MAIN e ho creato il Pthread UTENTE 14-esimo con id=139800058341120
68  Sto per creare il thread UTENTE 15-esimo
69  UTENTE-[Thread14 e identificatore 139800058341120] STO ARRIVANDO
70  UTENTE-[Thread14 e identificatore 139800058341120] voglio ordinare 8
        scatoloni, mi inserisco nella coda NORMALE
71  SONO IL MAIN e ho creato il Pthread UTENTE 15-esimo con id=139799172216576
72  Sto per creare il thread UTENTE 16-esimo
73  UTENTE-[Thread15 e identificatore 139799172216576] STO ARRIVANDO
74  UTENTE-[Thread15 e identificatore 139799172216576] voglio ordinare 6
        scatoloni, mi inserisco nella coda NORMALE
75  SONO IL MAIN e ho creato il Pthread UTENTE 16-esimo con id=139799037998848
76  Sto per creare il thread UTENTE 17-esimo
77  UTENTE-[Thread16 e identificatore 139799037998848] STO ARRIVANDO
```

```
78  UTENTE-[Thread16 e identificatore 139799037998848] voglio ordinare 6
        scatoloni, mi inserisco nella coda NORMALE
79  SONO IL MAIN e ho creato il Pthread UTENTE 17-esimo con id=139799163823872
80  UTENTE-[Thread17 e identificatore 139799163823872] STO ARRIVANDO
81  UTENTE-[Thread17 e identificatore 139799163823872] voglio ordinare 8
        scatoloni, mi inserisco nella coda NORMALE
82  UTENTE-[Thread10 e identificatore 139800091911936] STO ARRIVANDO
83  UTENTE-[Thread10 e identificatore 139800091911936] voglio ordinare 2
        scatoloni, mi inserisco nella coda NORMALE
84  CORRIERE-[Thread0 e identificatore 139800251307776] ho consegnato l'ordine
        dell'utente 3, sto rientrando (durata rientro: 3 secondi)
85  UTENTE-[Thread3 e identificatore 139800234522368] il corriere e' appena
        arrivato, pago e termino l'ordine
86  Pthread 3-esimo restituisce 3
87  CORRIERE-[Thread2 e identificatore 139800242915072] ho consegnato l'ordine
        dell'utente 5, sto rientrando (durata rientro: 3 secondi)
88  UTENTE-[Thread5 e identificatore 139800217736960] il corriere e' appena
        arrivato, pago e termino l'ordine
89  CORRIERE-[Thread1 e identificatore 139800108697344] ho consegnato l'ordine
        dell'utente 4, sto rientrando (durata rientro: 4 secondi)
90  UTENTE-[Thread4 e identificatore 139800226129664] il corriere e' appena
        arrivato, pago e termino l'ordine
91  Pthread 4-esimo restituisce 4
92  Pthread 5-esimo restituisce 5
93  CORRIERE-[Thread0 e identificatore 139800251307776] un utente e' in coda
        NORMALE, lo rimuovo dalla coda
94  CORRIERE-[Thread0 e identificatore 139800251307776] prendo in carico l'
        ordine dell'utente con id 6, la spedizione avra' durata di 3 secondi
95  UTENTE-[Thread6 e identificatore 139800209344256] il corriere con id 0 ha
        appena preso in carico il mio ordine
96  CORRIERE-[Thread2 e identificatore 139800242915072] un utente e' in coda
        NORMALE, lo rimuovo dalla coda
97  CORRIERE-[Thread2 e identificatore 139800242915072] prendo in carico l'
        ordine dell'utente con id 7, la spedizione avra' durata di 3 secondi
98  UTENTE-[Thread7 e identificatore 139800200951552] il corriere con id 2 ha
        appena preso in carico il mio ordine
```

```
99  CORRIERE-[Thread1 e identificatore 139800108697344] un utente e' in coda
        NORMALE, lo rimuovo dalla coda
100 CORRIERE-[Thread1 e identificatore 139800108697344] prendo in carico l'
        ordine dell'utente con id 8, la spedizione avra' durata di 4 secondi
101 UTENTE-[Thread8 e identificatore 139800192558848] il corriere con id 1 ha
        appena preso in carico il mio ordine
102 CORRIERE-[Thread0 e identificatore 139800251307776] ho consegnato l'ordine
         dell'utente 6, sto rientrando (durata rientro: 3 secondi)
103 UTENTE-[Thread6 e identificatore 139800209344256] il corriere e' appena
        arrivato, pago e termino l'ordine
104 CORRIERE-[Thread2 e identificatore 139800242915072] ho consegnato l'ordine
         dell'utente 7, sto rientrando (durata rientro: 3 secondi)
105 UTENTE-[Thread7 e identificatore 139800200951552] il corriere e' appena
        arrivato, pago e termino l'ordine
106 Pthread 6-esimo restituisce 6
107 Pthread 7-esimo restituisce 7
108 CORRIERE-[Thread0 e identificatore 139800251307776] un utente e' in coda
        NORMALE, lo rimuovo dalla coda
109 CORRIERE-[Thread0 e identificatore 139800251307776] prendo in carico l'
        ordine dell'utente con id 9, la spedizione avra' durata di 1 secondi
110 UTENTE-[Thread9 e identificatore 139800100304640] il corriere con id 0 ha
        appena preso in carico il mio ordine
111 CORRIERE-[Thread2 e identificatore 139800242915072] un utente e' in coda
        NORMALE, lo rimuovo dalla coda
112 CORRIERE-[Thread2 e identificatore 139800242915072] prendo in carico l'
        ordine dell'utente con id 11, la spedizione avra' durata di 6 secondi
113 UTENTE-[Thread11 e identificatore 139800083519232] il corriere con id 2 ha
         appena preso in carico il mio ordine
114 CORRIERE-[Thread1 e identificatore 139800108697344] ho consegnato l'ordine
         dell'utente 8, sto rientrando (durata rientro: 4 secondi)
115 UTENTE-[Thread8 e identificatore 139800192558848] il corriere e' appena
        arrivato, pago e termino l'ordine
116 Pthread 8-esimo restituisce 8
117 CORRIERE-[Thread0 e identificatore 139800251307776] ho consegnato l'ordine
         dell'utente 9, sto rientrando (durata rientro: 1 secondi)
118 UTENTE-[Thread9 e identificatore 139800100304640] il corriere e' appena
```

```
    arrivato, pago e termino l'ordine
119 Pthread 9-esimo restituisce 9
120 CORRIERE-[Thread0 e identificatore 139800251307776] un utente e' in coda
    NORMALE, lo rimuovo dalla coda
121 CORRIERE-[Thread0 e identificatore 139800251307776] prendo in carico l'
    ordine dell'utente con id 12, la spedizione avra' durata di 3 secondi
122 UTENTE-[Thread12 e identificatore 139800075126528] il corriere con id 0 ha
     appena preso in carico il mio ordine
123 CORRIERE-[Thread1 e identificatore 139800108697344] un utente e' in coda
    NORMALE, lo rimuovo dalla coda
124 CORRIERE-[Thread1 e identificatore 139800108697344] prendo in carico l'
    ordine dell'utente con id 13, la spedizione avra' durata di 3 secondi
125 UTENTE-[Thread13 e identificatore 139800066733824] il corriere con id 1 ha
     appena preso in carico il mio ordine
126 CORRIERE-[Thread0 e identificatore 139800251307776] ho consegnato l'ordine
     dell'utente 12, sto rientrando (durata rientro: 3 secondi)
127 UTENTE-[Thread12 e identificatore 139800075126528] il corriere e' appena
    arrivato, pago e termino l'ordine
128 CORRIERE-[Thread2 e identificatore 139800242915072] ho consegnato l'ordine
     dell'utente 11, sto rientrando (durata rientro: 6 secondi)
129 UTENTE-[Thread11 e identificatore 139800083519232] il corriere e' appena
    arrivato, pago e termino l'ordine
130 CORRIERE-[Thread1 e identificatore 139800108697344] ho consegnato l'ordine
     dell'utente 13, sto rientrando (durata rientro: 3 secondi)
131 UTENTE-[Thread13 e identificatore 139800066733824] il corriere e' appena
    arrivato, pago e termino l'ordine
132 CORRIERE-[Thread0 e identificatore 139800251307776] un utente e' in coda
    NORMALE, lo rimuovo dalla coda
133 CORRIERE-[Thread0 e identificatore 139800251307776] prendo in carico l'
    ordine dell'utente con id 14, la spedizione avra' durata di 6 secondi
134 UTENTE-[Thread14 e identificatore 139800058341120] il corriere con id 0 ha
     appena preso in carico il mio ordine
135 CORRIERE-[Thread1 e identificatore 139800108697344] un utente e' in coda
    NORMALE, lo rimuovo dalla coda
136 CORRIERE-[Thread1 e identificatore 139800108697344] prendo in carico l'
    ordine dell'utente con id 15, la spedizione avra' durata di 6 secondi
```

```
137     UTENTE-[Thread15 e identificatore 139799172216576] il corriere con id 1 ha
            appena preso in carico il mio ordine
138     CORRIERE-[Thread2 e identificatore 139800242915072] un utente e' in coda
            NORMALE, lo rimuovo dalla coda
139     CORRIERE-[Thread2 e identificatore 139800242915072] prendo in carico l'
            ordine dell'utente con id 16, la spedizione avra' durata di 7 secondi
140     UTENTE-[Thread16 e identificatore 139799037998848] il corriere con id 2 ha
            appena preso in carico il mio ordine
141     CORRIERE-[Thread0 e identificatore 139800251307776] ho consegnato l'ordine
            dell'utente 14, sto rientrando (durata rientro: 6 secondi)
142     UTENTE-[Thread14 e identificatore 139800058341120] il corriere e' appena
            arrivato, pago e termino l'ordine
143     CORRIERE-[Thread1 e identificatore 139800108697344] ho consegnato l'ordine
            dell'utente 15, sto rientrando (durata rientro: 6 secondi)
144     UTENTE-[Thread15 e identificatore 139799172216576] il corriere e' appena
            arrivato, pago e termino l'ordine
145     CORRIERE-[Thread2 e identificatore 139800242915072] ho consegnato l'ordine
            dell'utente 16, sto rientrando (durata rientro: 7 secondi)
146     UTENTE-[Thread16 e identificatore 139799037998848] il corriere e' appena
            arrivato, pago e termino l'ordine
147     CORRIERE-[Thread0 e identificatore 139800251307776] un utente e' in coda
            NORMALE, lo rimuovo dalla coda
148     CORRIERE-[Thread0 e identificatore 139800251307776] prendo in carico l'
            ordine dell'utente con id 17, la spedizione avra' durata di 2 secondi
149     UTENTE-[Thread17 e identificatore 139799163823872] il corriere con id 0 ha
            appena preso in carico il mio ordine
150     CORRIERE-[Thread1 e identificatore 139800108697344] un utente e' in coda
            NORMALE, lo rimuovo dalla coda
151     CORRIERE-[Thread1 e identificatore 139800108697344] prendo in carico l'
            ordine dell'utente con id 10, la spedizione avra' durata di 6 secondi
152     UTENTE-[Thread10 e identificatore 139800091911936] il corriere con id 1 ha
            appena preso in carico il mio ordine
153     CORRIERE-[Thread0 e identificatore 139800251307776] ho consegnato l'ordine
            dell'utente 17, sto rientrando (durata rientro: 2 secondi)
154     UTENTE-[Thread17 e identificatore 139799163823872] il corriere e' appena
            arrivato, pago e termino l'ordine
```

```
155    CORRIERE-[Thread0 e identificatore 139800251307776] non ci sono utenti in
           coda, mi sospendo
156    CORRIERE-[Thread2 e identificatore 139800242915072] non ci sono utenti in
           coda, mi sospendo
157    CORRIERE-[Thread1 e identificatore 139800108697344] ho consegnato l'ordine
            dell'utente 10, sto rientrando (durata rientro: 6 secondi)
158    UTENTE-[Thread10 e identificatore 139800091911936] il corriere e' appena
           arrivato, pago e termino l'ordine
159    Pthread 10-esimo restituisce 10
160    Pthread 11-esimo restituisce 11
161    Pthread 12-esimo restituisce 12
162    Pthread 13-esimo restituisce 13
163    Pthread 14-esimo restituisce 14
164    Pthread 15-esimo restituisce 15
165    Pthread 16-esimo restituisce 16
166    Pthread 17-esimo restituisce 17
167    Contatore coda prioritaria: 0
168    Contatore coda normale: 0
169    Stato utenti: [ -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 ]
```

Codice 4.5: Output del programma del problema della Vetrina Online

4.5 Possibili problemi di Starvation o Deadlock

La soluzione presentata è stata realizzata evitando qualunque problema di Deadlock, dunque non sono presenti problemi di questo genere. Per quanto riguarda i problemi di Starvation, è possibile che si presenti tale problema nel caso in cui la maggior parte degli utenti ordina un numero di scatoloni pari a 18. In questo caso la coda prioritaria sarà sempre popolata e provocherebbe una fase di Starvation per i thread *Utenti* in coda non prioritaria. È possibile evitare tale problema con l'aggiunta di un vincolo sul numero di utenti prioritari che vengono serviti consecutivamente, mediante l'utilizzo di un contatore di utenti prioritari serviti in successione ed un controllo nella scelta della coda da servire, da parte dei thread *Corrieri*.

Conclusioni

In conclusione, abbiamo visto come risolvere quattro problemi inerenti alla programmazione concorrente, mediante l'utilizzo della libreria *Pthread*. Abbiamo, inoltre, visto come implementare il costrutto *monitor* mediante l'utilizzo di un semaforo binario di mutua esclusione e come far comunicare i thread tra di loro mediante l'uso di variabili di condizione. Si specifica, inoltre, che le soluzioni proposte in questo elaborato sono solo uno dei molteplici modi per risolvere i quattro problemi proposti e tali problemi possono essere risolti anche mediante l'utilizzo di semafori.

Sitografia

[1] *Testo relativo al problema dell'**Elicottero***. URL: https://www.didattica.agentgroup.unimo.it/didattica/psoLM/TestiEsami/10-11/10Dicembre2010.pdf.

[2] *Testo relativo al problema dell'**Officina***. URL: https://www.didattica.agentgroup.unimo.it/didattica/psoLM/TestiEsami/09-10/16Luglio2010.pdf.

[3] *Testo relativo al problema della **Banca***. URL: https://www.didattica.agentgroup.unimo.it/didattica/psoLM/TestiEsami/13-14/15Gennaio2014.pdf.

[4] *Testo relativo al problema della **Vetrina Online***. URL: https://www.didattica.agentgroup.unimo.it/didattica/psoLM/TestiEsami/14-15/15Dicembre2014.pdf.