

[Login >](#)

Secgroup Ca' Foscari DSI

- [Home](#)
- [Projects](#)
- [Teaching](#)
- [Competitions](#)
- [Contacts](#)
- [About](#)
- [Blog](#)



[Secgroup Ca' Foscari DSI](#) > [Teaching](#) > [Sistemi Operativi – modulo 2](#) > [Verifiche anni precedenti](#)
> [\[semafori\] Check-in in aeroporto](#)

- [Creazione di processi](#)
- [Esecuzione e terminazione](#)
- [Segnali](#)
- [Comunicazione tra processi](#)
- [Pipe](#)
- [Esercitazione sulla pipe](#)
- [Produttore e consumatore](#)
- [I Thread POSIX](#)
- [Sezione critica](#)
- [Semafori](#)
- [Programmazione con i semafori](#)
- [Semafori POSIX](#)
- [Monitor](#)
- [Thread in Java](#)
- [Programmazione con i Monitor](#)
- [Stallo](#)
- [Risultati verifiche](#)
- [Verifiche anni precedenti](#)
 - [\[2012-13\] Semafori: robots](#)
 - [\[2012-13\] Monitor: scheduler](#)
 - [\[2011-12\] Pipe](#)
 - [\[2011-12\] Semafori](#)
 - [\[2011-12\] Monitor](#)
 - [\[pipe\] Crackme](#)
 - [\[semafori\] Check-in in aeroporto](#)
 - [\[monitor\] Gioco di squadra](#)

[semafori] Check-in in aeroporto

Scopo del progetto è utilizzare i semafori POSIX per sincronizzare, in un aeroporto, N viaggiatori in coda per il check-in da effettuarsi in una delle CIN postazioni presenti.

I viaggiatori devono attendere nel caso tutte le postazioni siano occupate. Altrimenti si recano alla prima postazione libera, effettuano il check-in e vanno all'imbarco.

Gli schemi del main e del viaggiatore sono:

```
1  void * viaggiatore(void * i) {
2
3      < entra in aeroporto >
4
5      j = attendi_postazione();
6
7      < usa la postazione di check in >
8
9      libera_postazione(j);
10
11     < va all'imbarco >
12 }
13
14 main() {
15     pthread_t th[N];
16
17     // inizializza le strutture dati e i semafori
18     inizializza();
19
20     // crea i thread viaggiatori
21     crea_thread(th);
22
23     // attende la terminazione dei thread viaggiatori
24     attendi_thread(th);
25
26     // distrugge i semafori
27     chiudi();
28 }
```

Viene richiesto di implementare le quattro funzioni

1. void inizializza() che inizializza le strutture dati e i semafori;
2. void chiudi() che 'distrugge' i semafori;
3. int attendi_postazione() attende finché non c'è almeno uno sportello libero. Cerca il numero del primo sportello libero e lo restituisce. (Suggerimento: utilizzare un array di booleani per mantenere lo stato degli sportelli. Attenzione a proteggere le eventuali sezioni critiche!)
4. libera_postazione(int i) libera la postazione i-esima

Ecco il programma di test:

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <pthread.h>
4  #include <sys/types.h>
```

```
5  #include <semaphore.h>
6  #include <stdint.h>
7
8  #define N 100    // Numero di viaggiatori
9  #define CIN 10   // Numero postazioni di check-int
10
11  /***** variabili per il testing *****/
12      int check_postazione[CIN];
13      sem_t check_mutuo;
14  /*****/
15
16  void * viaggiatore(void * i);
17
18  void die(char * s, int i) {
19      printf("--> %s numero %i\n", s, i);
20      exit(1);
21  }
22
23
24  /***** parte da consegnare *****/
25
26  // dichiarazione di variabili globali
27
28  // funzioni del main
29  inizializza() {
30
31  }
32
33
34  chiudi() {
35
36  }
37
38  // funzioni del viaggiatore
39  int attendi_postazione() {
40
41  }
42
43  int libera_postazione(int j) {
44
45  }
46  /*****fine parte da consegnare *****/
47
48
49  void * viaggiatore(void * i) {
50      int j;
51      intptr_t id = (intptr_t)i;
52
53      printf("[Viaggiatore %d] Entro in aeroporto\n", (int) i
54
55      j = attendi_postazione();
56
57      /***** testing *****/
```

```
58         sem_wait(&check_mutuo);
59         if (check_postazione[j]==1)
60             die("postazione gia' occupata",j);
61         check_postazione[j]=1;
62         sem_post(&check_mutuo);
63         /***** */
64
65         printf("[Viaggiatore %d] Sto usando lo sportello %d \n",
66
67         /***** testing *****/
68             sleep(1);
69             check_postazione[j]=-1;
70         /***** */
71
72         libera_postazione(j);
73
74
75         printf("[Viaggiatore %d] Vado al gate! \n", (int) id);
76
77     }
78
79     crea_thread(pthread_t *th) {
80         intptr_t i;
81         int ret;
82
83         for (i=0;i<N;i++)
84             if (ret=pthread_create(&th[i],NULL,viaggiatore,(void)
85                 die("errore create",ret);
86     }
87
88     attendi_thread(pthread_t *th) {
89         intptr_t i;
90         int ret;
91
92         for (i=0;i<N;i++)
93             if (ret=pthread_join(th[i], NULL))
94                 die("errore join",ret);
95     }
96
97
98     main() {
99         pthread_t th[N];
100         int i;
101
102         sem_init(&check_mutuo,0,1);
103
104         for (i=0; i<CIN; i++)
105             check_postazione[i] = 0;
106
107         // inizializza le strutture dati e i semafori
108         inizializza();
109
110         // crea i thread viaggiatori
```

```

111     crea_thread(th);
112
113     printf("Creati %i viaggiatori \n", N);
114
115     // attende la terminazione dei thread viaggiatori
116     attendi_thread(th);
117
118     // distrugge i semafori
119     chiudi();
120
121     /***** testing *****/
122     for (i=0; i<CIN; i++) {
123         if (check_postazione[i] == 0)
124             die("postazione mai utilizzata", i);
125     }
126     /*****/
127 }

```

Comments: 4

[Leave a reply »](#)



Bogdan Alexandru

[March 6th, 2014 at 22:13](#)

Ragazzi posto anch'io la mia soluzione. Ciao

```

1  /***** parte da consegnare *****/
2
3  // dichiarazione di variabili globali
4  sem_t codaPostazioni; //semaforo per gestire la fila a
5  sem_t mutex; //semaforo mutua esclusione per
6  int statoPostazioni[CIN];
7  // funzioni del main
8  inizializza() {
9      sem_init(&codaPostazioni, 0, CIN);
10     sem_init(&mutex, 0, 1);
11     int i;
12     for(i=0; i<CIN; i++)
13         statoPostazioni[i]=0;
14 }
15
16
17 chiudi() {
18     sem_destroy(&codaPostazioni);
19     sem_destroy(&mutex);
20
21 }
22
23 // funzioni del viaggiatore

```

```

24  int attendi_postazione() {
25      int i;
26      sem_wait(&codaPostazioni); //accoda il viaggiatore nell
27      sem_wait(&mutex);          //proteggero la parte critica (let
28      for(i=0;i<CIN && statoPostazioni[i]==1;i++);
29      statoPostazioni[i]=1;
30      sem_post(&mutex);
31      return i;
32  }
33
34  int libera_postazione(int j) {
35      statoPostazioni[j]=0;
36      sem_post(&codaPostazioni); //sblocco il prossimo viaggi
37      return j;
38  }
39  }
40  /*****fine parte da consegnare *****/

```



Leonardo Minati

[March 7th, 2014 at 17:22](#)

```

1  /***** parte da consegnare *****/
2
3  // dichiarazione di variabili globali
4  sem_t liberi;
5  sem_t sportelli[CIN]; /*semafori per l'accesso agli sportell
6  int stato_sportelli[CIN]; /*0 -> libero ; 1 -> occupato*/
7
8  // funzioni del main
9  void inizializza()
10 {
11     /*inizializzo gli sportelli a liberi ed i semafori*/
12     int i;
13     for(i = 0; i < CIN; i++)
14     {
15         stato_sportelli[i] = 0;
16         sem_init(&sportelli[i], 0, 1);
17     }
18     sem_init(&liberi, 0, CIN);
19 }
20
21
22 void chiudi()
23 {
24     sem_destroy(&liberi);
25     int i;
26     for(i = 0; i < CIN; i++)
27         sem_destroy(&sportelli[i]);
28 }
29
30 // funzioni del viaggiatore
31 int attendi_postazione()

```

```
32  {
33      sem_wait(&liberi);
34      /*ok, ora si è liberato uno sportello: cerco quello libe
35      int i;
36      for(i = 0; i < CIN; i++)
37          if(stato_sportelli[i] == 0)
38              {
39                  sem_wait(&sportelli[i]);
40                  return i;
41              }
42  }
43
44  int libera_postazione(int j)
45  {
46      sem_post(&sportelli[j]);
47      sem_post(&liberi);
48  }
49  /*****fine parte da consegnare *****/
```

Ecco la mia soluzione...durante l'esecuzione lo sportello utilizzato è sempre il primo...è corretto?



Leonardo Minati

[March 7th, 2014 at 17:36](#)

Mi correggo, ho trovato l'errore



[riccardo](#)

[March 10th, 2014 at 12:01](#)

@Leonardo: ti sei dimenticato di proteggere con un mutex la sezione critica. Potrebbero esserci interferenze quando viene scelto lo sportello

Leave a Reply

Name *

Mail *

(will not be published)

Website

Comment

Submit Comment

© 2014 Secgroup Ca' Foscari DSI