

# Il problema dei filosofi a cena

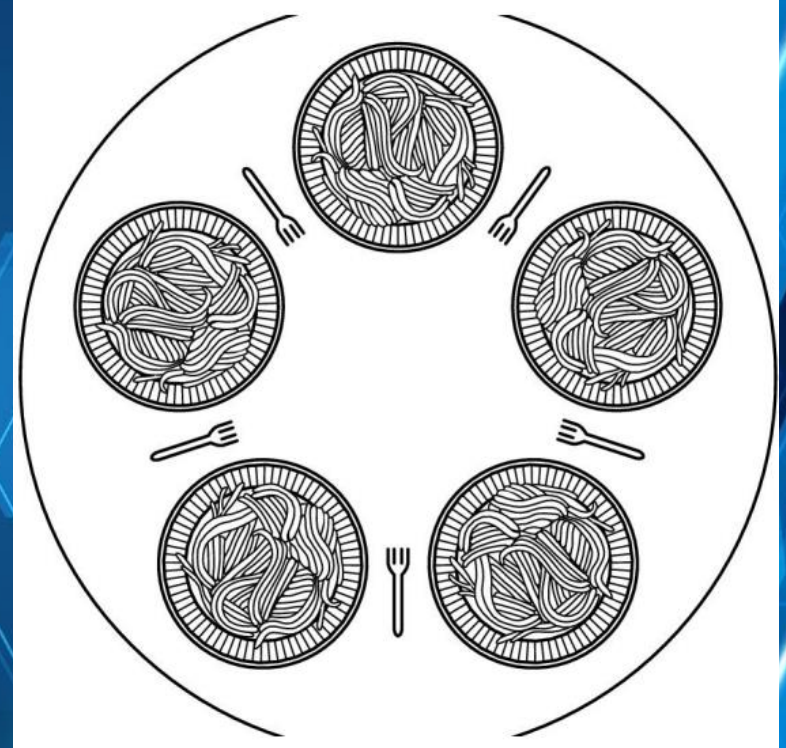


Università degli studi della Basilicata  
C.d.L. Informatica – Sistemi operativi  
Claps Federico (matr. 61635)  
Di Stefano Giuseppe (matr. 61436)

# Il problema dei filosofi a cena

Il problema dei filosofi a cena è un classico esempio di informatica che fu descritto nel 1965 da Edsger Dijkstra per illustrare un problema di sincronizzazione.

Ci sono dunque cinque filosofi, cinque piatti di spaghetti e cinque forchette. La vita di un filosofo consiste di periodi alterni di mangiare e pensare, e che ciascun filosofo abbia bisogno di due forchette per mangiare, ma che le forchette vengano prese una per volta. Dopo essere riuscito a prendere due forchette il filosofo mangia per un po', poi lascia le forchette e ricomincia a pensare



# Il problema dei filosofi a cena

E' possibile imbattersi in due situazioni:

**Deadlock:** situazione in cui due o più processi o azioni si bloccano a vicenda, aspettando che uno esegua una certa azione che serve all'altro e viceversa.

Il deadlock può verificarsi se ciascuno dei filosofi tiene in mano una forchetta senza mai riuscire a prendere l'altra. Il filosofo F1 aspetta di prendere la forchetta che ha in mano il filosofo F2, che aspetta la forchetta che ha in mano il filosofo F3, e così via in un circolo vizioso

**Starvation:** si intende l'impossibilità perpetua, da parte di un processo pronto all'esecuzione, di ottenere le risorse sia hardware sia software di cui necessita per essere eseguito.

La situazione di starvation può verificarsi indipendentemente dal deadlock se uno dei filosofi non riesce mai a prendere entrambe le forchette.



# Il problema dei filosofi a cena: Codice

Link codice GitHub <https://github.com/giuseppeDiStefano001/Filosofi-a-cenaOSO>

```
~/Scrivania/SO/problemaFilosofi.cpp - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
problemaFilosofi.cpp
1 #include <iostream>
2 #include <pthread.h>
3 #include <unistd.h>
4 #include <stdlib.h>
5 #include <semaphore.h>
6
7 using namespace std;
8
9 int const NUMERO_FILOSOFI = 5;
10
11 int statoFilosofo[NUMERO_FILOSOFI];
12 sem_t S; // dichiarazione semaforo binario (ogni filosofo ha un semaforo)
13 pthread_t filosofi[NUMERO_FILOSOFI]; // dichiarazione array di thread di filosofi
14 const int PENSA = 0;
15 const int MANGIA = 1;
16 const int AFFAMATO = 2;
17
18 void schermataIniziale();
19 void inizializza();
20 // void creaThread(pthread_t* filosofi);
21 void creaThread();
22 int generaDurataInMs(int min, int max);
23 int filosofoDestro(int i);
24 int filosofoSinistro(int i);
25 void prendiForchette(int i);
26 void rilasciaForchette(int i);
27 void *filosofo(void *i);
28 void pensa(int i);
29 void mangia(int i);
30 void wait(int &s);
31 void signal(int &s);
32 void joinThread();
33 void stampaMessaggio(string stringa1, int varInt, string stringa2);
34
35 int main()
36 {
37
38     // int numForchette = NUMERO_FILOSOFI;
39     inizializza();
40     schermataIniziale();
41     // while(true){
42     //     creaThread(filosofi);
43     //     creaThread();
44 }
```

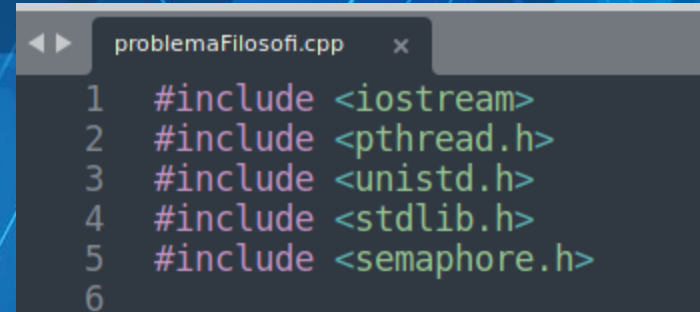
Line 59, Column 1

# Il problema dei filosofi a cena: Codice

La soluzione di Dijkstra utilizza un mutex, un semaforo per filosofo e una variabile di stato per filosofo.

Per la scrittura del codice sono state utilizzate le librerie:

- ❑ **pthread.h** : per i thread e i mutex
- ❑ **semaphore.h**: per i semafori



```
1  #include <iostream>
2  #include <pthread.h>
3  #include <unistd.h>
4  #include <stdlib.h>
5  #include <semaphore.h>
6
```

```
14  int statoFilosofo[NUMERO_FILOSOFI];
```

Inizializzazione array stato filosofo

```
15  sem_t S;
```

Inizializzazione semaforo

```
16  pthread_t filosofi[NUMERO_FILOSOFI];
```

Inizializzazione array thread filosofi

```
17  pthread_mutex_t mutexStampa;
```

Inizializzazione mutex

# Codice : Funzioni utilizzate

Principali metodi utilizzati:

## ❑ creaThread():

La funzione pthread\_create permette di creare un thread. Vengono passati come riferimenti:

- puntatore alla variabile che contiene l'id del thread
- attributi del thread
- procedura che il thread deve eseguire
- parametri della procedura da eseguire

## ❑ filosofo():

Nella funzione sono presenti quattro funzioni:

- pensa(): stato filosofo è "PENSA"
- prendiForkette(): stato filosofo è "AFFAMATO"
- mangia(): stato filosofo è "MANGIA"
- rilasciaForkette(): dopo aver mangiato il filosofo rilascia le due forchette

Lo stato dei filosofi è rappresentato da tre costanti intere: PENSA, MANGIA, AFFAMATO.

Per verificare lo stato di ogni filosofo, si utilizza un array di interi.

```
60
61 void creaThread()
62 {
63     for (unsigned long i = 0; i < NUMERO_FILOSOFI; i++)
64     {
65         usleep(1000000);
66         pthread_create(&filosofi[i], NULL, filosofo, (void *)i);
67     }
68 }
69
```

```
162
163 void *filosofo(void *i)
164 {
165     while (true) {
166         unsigned long threadID = (unsigned long)i;
167         pensa(threadID);
168         prendiForkette(threadID);
169         mangia(threadID);
170         rilasciaForkette(threadID);
171     }
172 }
173
```

```
11
12 int statoFilosofo[NUMERO_FILOSOFI];
13
14 const int PENSA = 0;
15 const int MANGIA = 1;
16 const int AFFAMATO = 2;
17
```

# Codice : Funzioni utilizzate

## ❑ **joinThread():**

La funzione `pthread_join` permette di mettere in attesa un thread. E di aspettare che termini il thread in esecuzione. Vengono passati come riferimenti:

- l'id del thread
- NULL

```
144
145 ▼ void joinThread() {
146     for (unsigned long i = 0; i < NUMERO_FILOSOFI; i++){
147         pthread_join(filosofi[i], NULL);
148     }
149 }
```

## ❑ **filosofoDestro() / filosofoSinistro():**

Queste due funzioni restituiscono rispettivamente la posizione del filosofo destro e del filosofo sinistro.

Es. Il filosofo 1 avrà alla sua destra il filosofo in posizione 2 e alla sua sinistra il filosofo in posizione 5.

```
74
75 int filosofoDestro(int i){
76     return (i + 1) % NUMERO_FILOSOFI;
77 }
78
79 int filosofoSinistro(int i){
80     return (i + NUMERO_FILOSOFI - 1) % NUMERO_FILOSOFI;
81 }
82
```



# Codice : Funzioni utilizzati

La funzione `filosofoStaMangiando()` e il suo utilizzo in `prendiForkette()` e `rilasciaForkette()` rendono la soluzione Dijkstra priva di deadlock.

```
117
118 void prendiForkette(int i)
119 {
120     statoFilosofo[i] = AFFAMATO;
121     pthread_mutex_lock(&mutexStampa);
122     usleep(1000000);
123     cout << "Il filosofo in pos. " << i << " e' AFFAMATO " << endl;
124     pthread_mutex_unlock(&mutexStampa);
125     filosofoStaMangiando(i);
126     sem_wait(&S);
127     sem_wait(&S);
128 }
129
130 void rilasciaForkette(int i)
131 {
132     statoFilosofo[i] = PENSA;
133     int sinistro = filosofoSinistro(i);
134     int destro = filosofoDestro(i);
135     filosofoStaMangiando(sinistro);
136     filosofoStaMangiando(destro);
137     sem_post(&S);
138     sem_post(&S);
139 }
140
```

La funzione `filosofoStaMangiando` verifica se il filosofo destro o il filosofo sinistro sta mangiando. Se nessuno dei due mangia e il filosofo è "AFFAMATO", allora prende la forchetta e "MANGIA".

```
102
103 void filosofoStaMangiando(int i)
104 {
105     int destro = filosofoDestro(i);
106     int sinistro = filosofoSinistro(i);
107     if ((statoFilosofo[i] == AFFAMATO) && (statoFilosofo[destro] != MANGIA) && (statoFilosofo[sinistro] != MANGIA))
108     {
109         statoFilosofo[i] = MANGIA;
110         sem_post(&S);
111     }
112 }
113
```



# Situazione di stallo: Deadlock

```
135 void *filosofo(void *i){
136     while (true) {
137         unsigned long threadID = (unsigned long)i;
138         pensa(threadID);
139         prendiForkette(threadID);
140         mangia(threadID);
141         //rilasciaForkette(threadID);
142     }
143 }
```

Commentando l'istruzione **rilasciaForkette(threadID)** situata all'interno del metodo filosofo abbiamo causato volontariamente una situazione di stallo. Il filosofo dopo aver pensato e dopo aver preso le forchette continua a mangiare, non rilasciando mai le forchette, andando quindi a occupare in modo indefinito le risorse (forchette) che servirebbero agli altri filosofi per mangiare.

```
*****
*      PROBLEMA DEI FILOSOFI A CENA      *
*      Corso di Sistemi Operativi        *
*      FEDERICO CLAPS - GIUSEPPE DI STEFANO *
*****
Il filosofo in posizione: 0  PENSA per 783 (ms)
Il filosofo in posizione: 2  PENSA per 686 (ms)
Il filosofo in posizione: 3  PENSA per 977 (ms)
Il filosofo in posizione: 1  PENSA per 715 (ms)
Il filosofo in posizione: 4  PENSA per 793 (ms)
Il filosofo in posizione: 0  E' AFFAMATO
Il filosofo in posizione: 0  MANGIA per 935 (ms)
Il filosofo in posizione: 3  E' AFFAMATO
Il filosofo in posizione: 3  MANGIA per 692 (ms)
Il filosofo in posizione: 4  E' AFFAMATO
Il filosofo in posizione: 4  MANGIA per 821 (ms)
Il filosofo in posizione: 0  PENSA per 786 (ms)
Il filosofo in posizione: 1  E' AFFAMATO
Il filosofo in posizione: 3  PENSA per 849 (ms)
Il filosofo in posizione: 2  E' AFFAMATO
Il filosofo in posizione: 4  PENSA per 962 (ms)
Il filosofo in posizione: 0  E' AFFAMATO
Il filosofo in posizione: 3  E' AFFAMATO
Il filosofo in posizione: 4  E' AFFAMATO
□
```

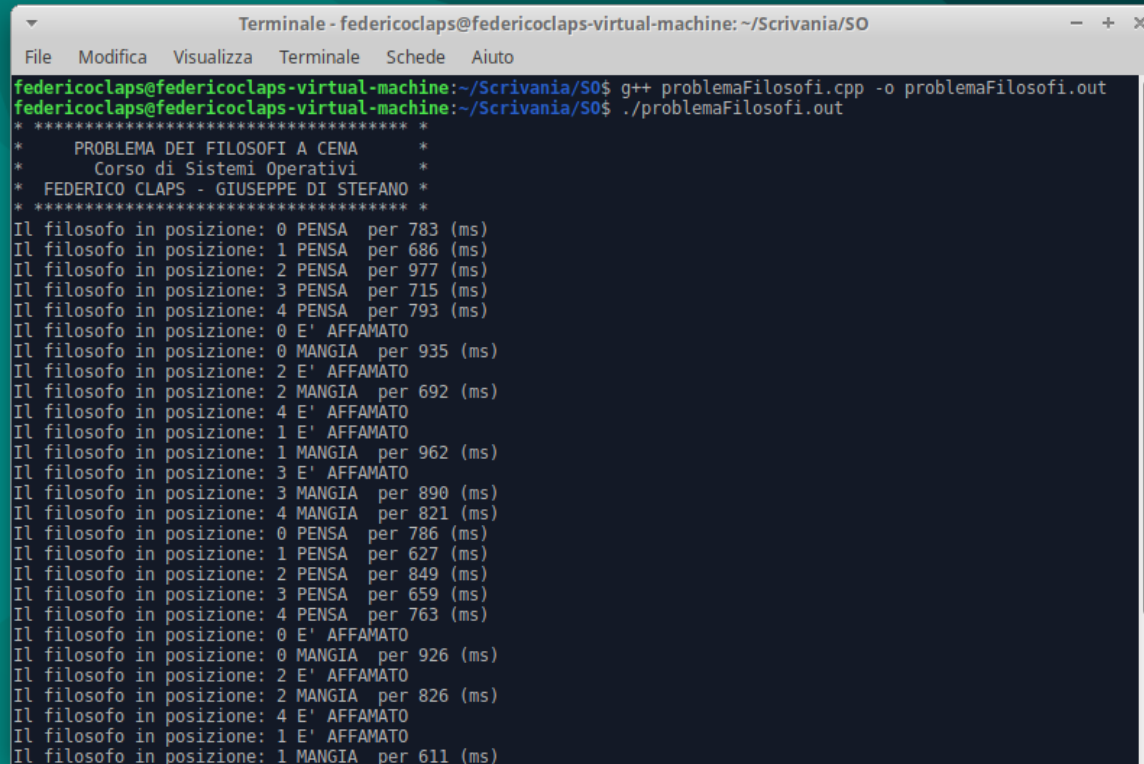
# Situazione senza Deadlock o Starvation

Codice compilato ed eseguito:

```
Terminale - federicoclaps@federicoclaps-virtual-machine: ~/Scrivania/SO
File Modifica Visualizza Terminale Schede Aiuto
federicoclaps@federicoclaps-virtual-machine:~/Scrivania/SO$ g++ problemaFilosofi.cpp -o problemaFilosofi.out
federicoclaps@federicoclaps-virtual-machine:~/Scrivania/SO$ ./problemaFilosofi.out
*****
* PROBLEMA DEI FILOSOFI A CENA *
* Corso di Sistemi Operativi *
* FEDERICO CLAPS - GIUSEPPE DI STEFANO *
*****
Il filosofo in posizione: 0 PENSA per 783 (ms)
Il filosofo in posizione: 1 PENSA per 686 (ms)
Il filosofo in posizione: 2 PENSA per 977 (ms)
Il filosofo in posizione: 3 PENSA per 715 (ms)
Il filosofo in posizione: 4 PENSA per 793 (ms)
Il filosofo in posizione: 0 E' AFFAMATO
Il filosofo in posizione: 0 MANGIA per 935 (ms)
Il filosofo in posizione: 2 E' AFFAMATO
Il filosofo in posizione: 2 MANGIA per 692 (ms)
Il filosofo in posizione: 4 E' AFFAMATO
Il filosofo in posizione: 1 E' AFFAMATO
Il filosofo in posizione: 1 MANGIA per 962 (ms)
Il filosofo in posizione: 3 E' AFFAMATO
Il filosofo in posizione: 3 MANGIA per 890 (ms)
Il filosofo in posizione: 4 MANGIA per 821 (ms)
Il filosofo in posizione: 0 PENSA per 786 (ms)
Il filosofo in posizione: 1 PENSA per 627 (ms)
Il filosofo in posizione: 2 PENSA per 849 (ms)
Il filosofo in posizione: 3 PENSA per 659 (ms)
Il filosofo in posizione: 4 PENSA per 763 (ms)
Il filosofo in posizione: 0 E' AFFAMATO
Il filosofo in posizione: 0 MANGIA per 926 (ms)
Il filosofo in posizione: 2 E' AFFAMATO
Il filosofo in posizione: 2 MANGIA per 826 (ms)
Il filosofo in posizione: 4 E' AFFAMATO
Il filosofo in posizione: 1 E' AFFAMATO
Il filosofo in posizione: 1 MANGIA per 611 (ms)
```

# Codice compilato

Eseguendo il codice, è possibile notare come nessuno dei thread vada in deadlock o vada incontro a starvation



```
Terminale - federicoclaps@federicoclaps-virtual-machine: ~/Scrivania/SO
File Modifica Visualizza Terminale Schede Aiuto
federicoclaps@federicoclaps-virtual-machine:~/Scrivania/SO$ g++ problemaFilosofi.cpp -o problemaFilosofi.out
federicoclaps@federicoclaps-virtual-machine:~/Scrivania/SO$ ./problemaFilosofi.out
*****
* PROBLEMA DEI FILOSOFI A CENA *
* Corso di Sistemi Operativi *
* FEDERICO CLAPS - GIUSEPPE DI STEFANO *
*****
Il filosofo in posizione: 0 PENSA per 783 (ms)
Il filosofo in posizione: 1 PENSA per 686 (ms)
Il filosofo in posizione: 2 PENSA per 977 (ms)
Il filosofo in posizione: 3 PENSA per 715 (ms)
Il filosofo in posizione: 4 PENSA per 793 (ms)
Il filosofo in posizione: 0 E' AFFAMATO
Il filosofo in posizione: 0 MANGIA per 935 (ms)
Il filosofo in posizione: 2 E' AFFAMATO
Il filosofo in posizione: 2 MANGIA per 692 (ms)
Il filosofo in posizione: 4 E' AFFAMATO
Il filosofo in posizione: 1 E' AFFAMATO
Il filosofo in posizione: 1 MANGIA per 962 (ms)
Il filosofo in posizione: 3 E' AFFAMATO
Il filosofo in posizione: 3 MANGIA per 890 (ms)
Il filosofo in posizione: 4 MANGIA per 821 (ms)
Il filosofo in posizione: 0 PENSA per 786 (ms)
Il filosofo in posizione: 1 PENSA per 627 (ms)
Il filosofo in posizione: 2 PENSA per 849 (ms)
Il filosofo in posizione: 3 PENSA per 659 (ms)
Il filosofo in posizione: 4 PENSA per 763 (ms)
Il filosofo in posizione: 0 E' AFFAMATO
Il filosofo in posizione: 0 MANGIA per 926 (ms)
Il filosofo in posizione: 2 E' AFFAMATO
Il filosofo in posizione: 2 MANGIA per 826 (ms)
Il filosofo in posizione: 4 E' AFFAMATO
Il filosofo in posizione: 1 E' AFFAMATO
Il filosofo in posizione: 1 MANGIA per 611 (ms)
```

Per interrompere l'esecuzione che andrebbe all'infinito, è necessario premere la combinazione: **Ctrl + C**.