Dijkstra e il problema dei Filosofi a Cena

Cinque filosofi siedono davanti a un piatto di spaghetti **con un solo bastoncino a destra e uno solo a sinistra.**
Nel problema si deve immaginare **che ciascun filosofo possa mangiare SOLO CON DUE BASTONCINI anziché uno soltanto.**
Cinque sono i filosofi, cinque sono i piatti di spaghetti e cinque sono i bastoncini.

Si deve pensare che:

- la vita di un filosofo consista di periodi alterni di cibo e pensiero
- ciascun filosofo abbia bisogno di due bastoncini per mangiare
- che i bastoncini vengano presi una per volta.

Dopo essere riuscito a prendere due bastoncini, il filosofo mangia per un pò, poi lascia i bastoncini e ricomincia a pensare.

Ecco il problema: **trovare una regola che impedisca lo stallo (**deadlock**) o la morte per fame (**starvation).

Tanto per capirci meglio:

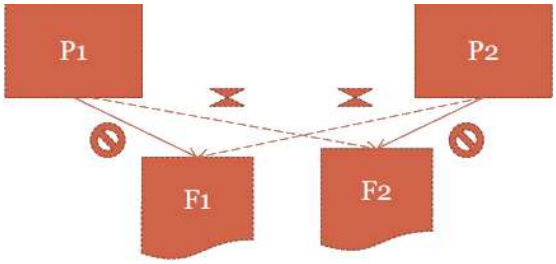
- il **deadlock** corrisponde a un filosofo che tiene in mano un bastoncino senza mai riuscire a prendere l'altro, cioè: il filosofo F1 aspetta di prendere il bastoncino che ha in mano il filosofo F2, che a sua volta aspetta il bastoncino che ha in mano il filosofo F3, e così via all'infinito.
- la **starvation** può verificarsi indipendentemente dal deadlock se uno dei filosofi non riesce mai a prendere tutte e due i bastoncini.

Finché i cinque filosofi non si metteranno d'accordo sulla sequenza della presa e del rilascio dei bastoncini, prima o poi il Sistema complessivo entrerà in stallo e si bloccherà inevitabilmente.

Spostando il problema in tema di **Risorse Informatiche** all'interno del nostro comunissimo PC di casa, i cosiddetti 'blocchi del Sistema' che ci fanno innervosire e ci costringono a chiudere le sessioni di lavoro con l'odiato comando **Ctrl Alt Canc**, corrispondono proprio ai cinque filosofi che non sanno più destreggiarsi con lo scambio dei bastoncini nella cena.
Già! E' vero che il PC è sempre più potente ma le sue risorse sono quelle che sono e il software è sempre più ingombrante, ragion per cui un deadlock o una starvation ci può anche stare!

Volendo fare i raggi X al nostro PC, va detto che il blocco di una risorsa è una tecnica comune per garantirne l'accesso da parte di un programma o di una sua partizione, ma se quella risorsa è già stata bloccata, il programma aspetta fino a quando verrà sbloccata.
Quando però il blocco coinvolge più di una risorsa, è possibile arrivare all'estremo, cioè al deadlock.

Per esempio, se due programmi P1 e P2 girano contemporaneamente tenendo bloccato uno stesso file F1, si verificherà certamente che entrambi aspetteranno lo sblocco del file da parte dell'altro programma, ovviamente invano e all'infinito.
Come altro esempio si consideri il caso di un programma che ha bisogno di due file da elaborare. Se due programmi di questo genere bloccano un file ciascuno, entrambi aspetteranno invano che l'altro programma sblocchi l'altro file.

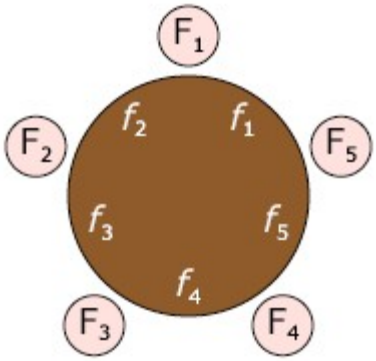


Abbiamo detto che questo problema si inquadra nell'ambito del **controllo della concorrenza** ma è anche un problema di **sincronizzazione**, tipo quello utilizzato nei nodi dei semafori ferroviari.

L'esempio di cui stiamo parlando fu descritto nel 1965 dall'informatico olandese **Edsger Dijkstra** che se ne servì, appunto, per descrivere un problema classico di sincronizzazione.
Dijkstra era nato a Rotterdam nel '30 ed è morto nel 2002 a Nuenen, la città in cui visse Van Gogh alla fine dell'800.
E' rimasto celebre per il suo più importante contributo all'informatica noto come **l'algoritmo di Dijkstra** nel concetto semaforico.

Tornando a noi e alla cena coi filosofi: qual è una possibile soluzione semplice alla sincronizzazione?

Eccola qua, la vedete nell'immagine:



1. Numerare i bastoncini ed esigere che vengano prese in ordine numerico crescente. In questa soluzione i filosofi sono denominati F1, F2, F3, F4 e F5, mentre i bastoncini alla loro sinistra sono rispettivamente f1, f2, f3, f4 e f5.
2. Il primo filosofo F1 dovrà prendere il primo bastoncino f1 prima di poter prendere il secondo bastoncino f2. I filosofi F2, F3 e F4 si comporteranno in modo analogo, prendendo sempre il bastoncino fi prima del bastoncino fi+1. Rispettando l'ordine numerico ma invertendo l'ordine delle mani, il filosofo F5 prenderà prima il bastoncino f1 e poi il bastoncino f5. Si crea così un'asimmetria che serve ad evitare il blocco.

Un'ottima soluzione di un metodo di **mutua esclusione**.

Già … alle volte non si pensa al fatto che i problemi di logica pura che tanto ci creano imbarazzo e ci fanno irritare tutte le volte che ci vengono proposti, in realtà sono la chiave di più di una soluzione di problemi molto più complessi di pubblica utilità.
Di come dovessero scambiarsi i bastoncini Socrate e Platone, sarete d'accordo con me che ben poco ci importava, ma che da quello stupido problema si sviluppasse poi il controllo dei processi informatici, la cosa non fa che stupirci sempre più.

Regole per consentire ad un n agenti (processi) di ottenere accesso esclusivo ad m risorse:

1. Tutti gli agenti devono richiedere accesso alle risorse in ordine crescente, prima cioè di richiedere l'accesso ad una risorsa di ordine maggiore, l'agente deve aver ottenuto l'accesso alle risorse di ordine minore di cui ha bisogno.
2. Prima di richiedere l'accesso a una risorsa di ordine minore, l'agente deve rilasciare le risorse di ordine maggiore alle quali sta accedendo.

La programmazione concorrente nasce per gestire i Sistemi Concorrenti cioe' sistemi in grado di supportare più utenti (o programmi) contemporaneamente
Sistemi intrinsecamente concorrenti:

- Sistemi Real Time
- Sistemi operativi
- Gestione di basi di dati

Applicazioni potenzialmente concorrenti. Uso di algoritmi paralleli per computazioni:

- su grandi quantità di dati
- con grande mole di calcolo
- vincoli di tempo reale.

Per sistema **concorrente** intendiamo:

- un sistema software
- implementato su vari tipi di hardware
- che porta avanti contemporaneamente una molteplicità di attività diverse
- tra di loro correlate
    - possono cooperare ad un risultato comune
    - possono competere per risorse condivise

- l concetto di concorrenza è contrapposto a quello di sequenzialità.
- In un sistema sequenziale i processi vengono eseguiti uno per volta e non si verifica alcuna forma di interazione tra essi durante l'esecuzione.
- Un programma sequenziale:
    - sequenza di dichiarazioni e istruzioni che verranno
    - eseguite sequenzialmente
- Un programma concorrente:
    - un insieme di programmi sequenziali che verranno
    - eseguiti in parallelo

La programmazione concorrente

- Avanzamento alternato
- Reale sovrapposizione

Cobegin
    P1
    P2
    …
    Pn
Coend

Competizione e collaborazione tra processi

Si ha **interferenza** quando due processi per evolvere **hanno bisogno della sessa risorsa**. Si deve garantire:

- mutua esclusione
- attesa limitata
- avanzamento

Si ha **cooperazione** quando sono logicamente interconnessi, cioè quando uno per evolvere deve usare una **risorsa che viene prodotta dall'altro**.

Si ha lo **stallo** quando ciascuno si trova in attesa di una risorsa che deve essere generata dall'altro.