

Il semaforo

Definizioni

- **Semaforo:** strumento linguistico di basso livello per risolvere problemi di sincronizzazione.
- **Struttura dati semaforo:** un semaforo è rappresentato da una variabile di tipo *semaphore*, definita come variabile intera non negativa, cui è possibile accedere solo tramite le due operazioni *P* e *V*
 - Un semaforo è condivisibile da due o più processi;
 - Le operazioni sono sezioni critiche

Specifica operazioni

```
semaphore s = i; // i >= 0

void P(semaphore s):
    region s << when (vals > 0) vals--; >>

void V(semaphore s):
    region s << vals++; >>
```

- P sospende il processo chiamante se $val = 0$;
- V risveglia un processo se $val = 0$.

Proprietà

```
// Costanti utilizzate
int I;           // Valore >= 0 con cui il semaforo è inizializzato
int nv,         // Numero di volte che l'operazione V è stata eseguita
int ni;         // Numero di volte che l'operazione P è stata INIZIATA
int np;         // Numero di volte che l'operazione P è stata COMPLETATA
int bloccati;   // Numero di processi sospesi sul semaforo

// #1: Relazione tra valore di semaforo e chiamate alle operazioni
val = I + nv - np;

// #2: Relazione di invarianza (derivata da #1 e da val >= 0)
np <= I + nv;

// #3: Relazioni derivate
bloccati = ni - np;
val > 0 => bloccati == 0;
bloccati > 0 => val == 0;
```

Tipologie di semaforo

- Semafori di mutua esclusione
- Semafori evento
- Semafori binari composti
- Semafori condizione
- Semafori risorsa
- Semafori privati

Semafori di mutua esclusione

Definizione: semaforo binario, può assumere solo i valori 0 e 1.

<code>semaphore mutex = 1;</code>	
<code>void op1() { P(mutex); << corpo della funzione >>; V(mutex); }</code>	<code>void op2() { P(mutex); << corpo della funzione >>; V(mutex); }</code>

Proprietà dei mutex

<pre>// Proprietà 1: C'è massimo un processo nella sezione critica Th: 0 <= Nsez <= 1; Hp: I = 1; // Semaforo inizializzato a 1 Tutti hanno come prologo P(mutex) Tutti hanno come epilogo V(mutex) Dim: Nsez = np - nv; // Direttamente dalle ipotesi val = I + nv - np >= 0; // Proprietà semafori 1 + nv - np >= 0; 1 >= np - nv; 1 >= Nsez; // Inoltre il protocollo prevede sempre una P e una V np >= nv; np - nv >= 0; Nsez >= 0; // Unendo le due... 0 <= Nsez <= 1; // C.V.D. // Proprietà 2: Assenza di deadlock Th: Non esiste uno stato in cui: val = 0 && Nsez == 0; // Condizione deadlock Hp: I = 1; // Semaforo inizializzato a 1 Tutti hanno come prologo P(mutex) Tutti hanno come epilogo V(mutex) Dim: // Per assurdo, suppongo vera la condizione di deadlock val = I + nv - np; // Proprietà semafori 0 = 1 + nv - np; // Da prima condizione di deadlock Nsez == 0 => np - nv == 0 => np = nv; // Proprietà 1 0 = 1 + 0; // ASSURDO! // Proprietà: Se il mutex è libero, // non ci sono processi in attesa e il prossimo potrà entrare Th: Nsez == 0 => bloccati == 0 && val > 0; Hp: I = 1; // Semaforo inizializzato a 1 Tutti hanno come prologo P(mutex) Tutti hanno come epilogo V(mutex) Dim: Nsez == 0 => np = nv; // Supponiamo per assurdo che val == 0 val = I + nv - np; 0 = 1 + 0; // ASSURDO!</pre>

Semafori evento

Descrizione: semafori binari utilizzati per imporre un vincolo di precedenza tra le operazioni dei processi.

Esempio

<pre>// Esempio: op1 esegue solo dopo op2 semaphore event = 0;</pre>	
<pre>void op1() { P(event); << corpo della funzione >>; }</pre>	<pre>void op2() { << corpo della funzione >>; V(event); }</pre>

Esempio: Problema del rendez-vous

- P esegue pa() e pb()
- Q esegue qa() e qb()
- Le operazioni xb() possono essere eseguite solo quando entrambe le xa() sono finite
- Soluzione:
 - P: pa(), v(semb), p(semq), pb()
 - Q: qa(), v(semq), p(semb), qb()

Semafori binari composti

Definizione: insieme di semafori binari usato in modo che uno solo di essi sia inizializzato a 1 e tutti gli altri a 0 e ogni processo che li usa esegue sempre sequenze che iniziano con la P su uno di questi e termina con la V su un altro.

Esempio: due processi P1 e P2 utilizzano un buffer condiviso per scambiare dati di tipo T; il buffer va acceduto in maniera mutuamente esclusiva, P2 può leggere solo quando un dato è stato inserito e P1 può scrivere solo quando il buffer è vuoto.

<pre>semaphore vuoto = 1; semaphore pieno = 0;</pre>	
<pre>void invio(T dato) { P(vuoto); << corpo della funzione >>; V(pieno); }</pre>	<pre>T ricezione() { P(pieno); << corpo della funzione >>; V(vuoto); }</pre>

Semafori condizione

Definizione: utilizzato per sincronizzare i processi su una condizione.

Esempio con attesa circolare

<pre>// op1() : region R << when(C) S1; >> - Modello con attesa circolare semaphore mutex = 1; semaphore sem = 0; int csem = 0;</pre>	
<pre>void op1() { P(mutex); while (!C) { csem++; V(mutex); P(sem); } }</pre>	<pre>void op2() { P(mutex); S2; if (csem > 0 && C) { csem--; V(sem); } }</pre>

<pre> P(mutex); } S1; V(mutex); } </pre>	<pre> } V(mutex); } </pre>
--	----------------------------

Esempio con passaggio del testimone

<pre> // op1() : region R << when(C) S1; >> - Modello con passaggio di testimone // Pi efficiente ma risveglia solo un processo per volta semaphore mutex = 1; semaphore sem = 0; int csem = 0; </pre>	
<pre> void op1() { P(mutex); if (!C) { csem++; V(mutex); P(sem); csem--; } S1; V(mutex); } </pre>	<pre> void op2() { P(mutex); S2; if (csem > 0 && C) V(sem); else V(mutex); } </pre>

Semafori risorsa

Definizione: semafori generali, possono assumere qualsiasi valore ≥ 0 , sono impiegati per allocare risorse equivalenti: il valore del semaforo indica il numero di risorse disponibili.

Esempio: produttori e consumatori con buffer di dimensione N (ogni cella è una risorsa equivalente).

Semafori privati

Definizione: semaforo privato ad un processo quando solo tale processo può eseguire la P su quel semaforo (la V la può invece fare anche un altro). Viene inizializzato a 0.

- Usato per particolari politiche di allocazione di risorse;
- Chi acquisisce una risorsa, se non ha una particolare condizione di sincronizzazione, si sospende sul suo semaforo privato
- Chi rilascia una risorsa, sveglierà uno dei processi sospesi.