

Esercizio 4

a) Descrivere brevemente l'algoritmo MFQ con Affinity Scheduling per sistemi multiprocessore.

b) Una struttura dati è condivisa da 4 thread T1, T2, T3, T4. I thread T1 e T2 modificano la struttura condivisa, i thread T3 e T4 accedono alla struttura in sola lettura. Per l'accesso alla struttura dati si adotta la seguente politica:

- Inizialmente accede il primo thread che ne fa richiesta sia esso lettore o scrittore;
- Quando la struttura dati è in uso da T3 o da T4 (o da entrambi), se i thread T1 e T2 decidono di accedere alla struttura dati vengono messi in attesa. I thread in attesa vengono riattivati solo quando sia T3 sia T4 hanno completato l'accesso alla struttura dati. Se T1 e T2 sono entrambi in attesa allora viene riattivato uno dei due. Se né T1 né T2 sono in attesa allora la struttura dati ritorna disponibile a tutti i thread e si torna alla situazione iniziale;
- Quando la struttura dati condivisa è utilizzata da T1 (o da T2), tutti gli altri thread che richiedono l'accesso vengono messi in attesa. Quando T1 (oppure T2) termina il proprio accesso, verifica prima se T2 (o T1) è in attesa e in caso affermativo lo riattiva. Altrimenti, se sono in attesa, vengono riattivati T3 e/o T4. Se nessun thread è in attesa allora la struttura dati ritorna disponibile a tutti i thread e si torna alla situazione iniziale.

Si chiede di implementare, utilizzando esclusivamente i semafori come meccanismo di sincronizzazione, i metodi: `accediT1T2()`, `rilasciaT1T2()`, `accediT3T4()`, `rilasciaT3T4()`.

• Soluzione

↳ Passo 1: Identificare i problemi

Abbiamo 4 thread che accedono ad una struttura dati condivisa con le regole specifiche

- T₁ e T₂ (scrittori): solo uno alla volta può accedere
- T₁ e T₂ (lettori): possono accedere contemporaneamente, ma solo se nessuno scrittore è attivo / in attesa

⇒ Bisogna garantire mutua esclusione tra lettori e scrittori, con una gestione ordinata delle attese

↳ Passo 2: Identificare le variabili necessarie

Bisogna tenere traccia dello stato della struttura dati e gestire l'accesso con semafori e contatori:

• Semafori

→ mux: controlla l'accesso in mutua esclusione alle variabili condivise

→ waitT1T2: gestisce l'attesa degli scrittori

→ waitT3T4: gestisce l'attesa dei lettori.

- Contatori

→ $T_1 T_2$: indica se uno scrittore è nella sezione critica (\emptyset o 1)

→ $T_1 T_2 \text{ waiting}$: conta gli scrittori in attesa

→ $T_3 T_4$: conta i lettori attivi nella sezione critica

→ $T_3 T_4 \text{ waiting}$: conta i lettori in attesa

⇒ Obiettivo: Usare questi semafori e contatori per rispettare la politica di accesso richiesta.

↳ Passo 3: Definire le comportamenti degli scrittori (T_1 e T_2)

Gli scrittori devono rispettare le seguenti condizioni:

1. Se nessun lettore / scrittore è attivo → entra subito

2. Se ci sono lettori attivi o un altro scrittore → va in attesa su $\text{wait } T_1 T_2$

3. Quando uno scrittore termina, se ce ne sono altri in attesa, ne attiva 1 solo

4. Se nessuno scrittore è in attesa, attiva lui; lettori in attesa

- Passaggi per accedi($T_1 T_2 C$):

→ Entrare in mutua esclusione ($P(C_{\text{mux}})$)

→ Controllare se la sezione critica è libera ($T_3 T_4 == 0$, $T_1 T_2 == 0$)

• Se sì → lo scrittore entra ($T_1 T_2 == 1$)

• Se no → incrementa $T_1 T_2 \text{ waiting}$ e si mette in attesa

($P(\text{wait } T_1 T_2)$)

→ Uscire dalla mutua esclusione ($V(C_{\text{mux}})$)

- Passaggi per rilascia($T_1 T_2 C$)

→ Entrare in mutua esclusione ($P(C_{\text{mux}})$)

→ Segnalare che lo scrittore ha finito ($T_1 T_2 == 0$)

→ Se ci sono scrittori in attesa → aviarne uno ($V(\text{wait } T_1 T_2)$)

→ Se non ci sono scrittori in attesa → aviarne lui; lettori ($V(\text{wait } T_3 T_4)$)

→ Uscire dalla mutua esclusione ($V(C_{\text{mux}})$)

↳ Passo 4: Definire il comportamento dei lettori

I lettori devono rispettare queste condizioni:

1. Se non ci sono scrittori attivi / in attesa \rightarrow entra subito.

2. Se c'è uno scrittore attivo/in attesa \rightarrow devono aspettare (waitT3T4)

3. Quando un lettore esce, se è l'ultimo lettore e ci sono scrittori in attesa \rightarrow deve arrivare uno scrittore

- Passaggi per accedi: $T3T4C$)

- \rightarrow Entrare in mutua esclusione (PCmux)

- \rightarrow Controllare se ci sono scrittori in attesa ($T1T2\text{waiting} > 0$)

- o attivi ($T1T2 == 1$)

- Se no \rightarrow può entrare subito ($T3T4+t$)

- Se sì \rightarrow incrementa $T3T4\text{ waiting}$ e si mette in attesa (PCwaitT3T4)

- \rightarrow Uscire dalla mutua esclusione (VCmux)

- Passaggi per rilascio $T3T4C$)

- \rightarrow Entrare in mutua esclusione (PCmux)

- \rightarrow Decrementare $T3T4$

- \rightarrow Se è l'ultimo lettore e ci sono scrittori in attesa, arrivare

- uno scrittore (VCwaitT1T2)

- \rightarrow Uscire dalla mutua esclusione (VCmux)

• Dalla Logica allo PSEUDOCODICE

↳ Passo 1: Dichiarazioni delle variabili locali

Definisco le variabili globali: che servono per la sincronizzazione

Γ $\text{sem_mux} = 1;$ // controlla l'accesso alle variabili condivise Γ

$\text{sem_waitT1T2} = \emptyset;$ // Attesa degli scrittori

$\text{sem_waitT3T4} = \emptyset;$ // Attesa dei lettori

// Contatori

$T1T2 = \emptyset;$ // Indica se c'è uno scrittore attivo (\emptyset o 1)

$T1T2_waiting = \emptyset$ // # scrittori in attesa

$T3T4 = \emptyset;$ // # lettori attivi

$T3T4_waiting = \emptyset;$ // # lettori in attesa

⚠ Achtung:

- "mux" viene usato come mutua esclusione per modificare i contatori
- "waitT1T2" e "waitT3T4" sono usati per mettere in attesa i Thread
- I contatori ci permettono di Tenere traccia dello stato della sezione critica

↳ Passo 2: Implementazione di accedi T1T2(C)

Questa funzione deve:

→ Verificare se ci sono lettori o scrittori attivi

→ Se la sezione critica è libera, entra immediatamente

→ Altrimenti, si mette in attesa su "waitT1T2"

Γ void accedi(T1T2C){

7

bool OK = false; // Flag per sapere se entrare subito

PC & mux); // Entra in mutua esclusione

if (T3T4 == 0 && T1T2 == 0){ // Nessun lettore / scrittore attivo

T1T2 = 1;

OK = true; // Può entrare immediatamente

{ else {

T1T2 waiting ++; // Incrementa # scrittori in attesa

{

VC & mux); // Esce dalla mutua esclusione

if (!OK){

PC & wait T1T2); // Se non può entrare subito -> va in wait

{

L {

L

⚠️ Attenzione

- PC & mux) / VC & mux) proteggono le variabili condivise
- Se la sezione critica è libera (T1T2 == 0 e T3T4 == 0), lo scrittore entra subito.
- Altrimenti, aumenta il contatore degli scrittori in attesa e aspetta su wait T1T2.

↳ Passo 3: Implementazione di rilasciaT1T2C)

Questa funzione deve:

- Segnalare che lo scrittore ha finito.
- Se ci sono scrittori in attesa, svegliazne uno solo.
- Se non ci sono, svegliare tutti i lettori.

Void rilasciaT1T2C {

PC & mux); // Entra in mutua esclusione

T1T2 = φ; // Lo scrittore esce dalla sezione critica

if (T1T2waiting > 0) { // Se ci sono scrittori in attesa

T1T2 = 1; // Un nuovo scrittore entra

T1T2waiting --; // Riduce #scrittori in attesa

VC & waitT1T2); // Risveglia uno scrittore

else {

while (T3T4waiting > 0) { // Se non ci sono scrittori, sveglia i lettori

T3T4waiting --;

VC & waitT3T4);

{

{

↳ VC & mux); // Esce dalla mutua esclusione

⚠ Achtung:

- Se ci sono scrittori in attesa, ne sveglia uno solo
- Se non ci sono scrittori, risveglia tutti i lettori in attesa
- Il ciclo "while (T3T4waiting > 0)" assicura che tutti i lettori vengano riavvisati.

L> Passo 4: Implementazione di accedT3T4C()

Questa funzione deve:

- Controllare se un lettore può entrare subito o se deve aspettare
- Se non ci sono scrittori attivi o in attesa, entra immediatamente
- Altrimenti, si mette in attesa su waitT3T4

Void accedT3T4C {

```
    bool ok = false; // Flag per sapere se entrare subito
    PC & mux); // Entra in mutua esclusione
    if (T1T2 == φ && T1T2 == φ) { // Nessuno scrittore attivo o in attesa
        T3T4++;
        // Il lettore entra nella sezione critica
        ok = true; // Può entrare immediatamente
    } else {
        T3T4_waiting++;
        // Incrementa # lettori in attesa
    }
    VC & mux); // Esce dalla mutua esclusione
    if (!ok) {
        P C & waitT1T2); // Se non può entrare subito -> va in wait
    }
}
```

⚠ Achtung

- Se non ci sono scrittori attivi o in attesa, il lettore può entrare subito
- Se ci sono scrittori, il lettore si mette in attesa ($T3T4_waiting++$)

↳ Passo 5: Implementazione di rilascio T3T4 C;

Questa funzione deve:

→ Segnalare che un lettore ha finito.

→ Se e' l'ultimo lettore, avivare uno scrittore in attesa.

Void rilascia T3T4 C {

PC & mux); // Entra in mutua esclusione

T3T4--; // Il lettore esce dalla sezione critica

if (T3T4 == 0 && T1T2waiting > 0) { // Se era l'ultimo lettore e ci sono scrittori in attesa

T1T2 = 1; // Uno scrittore puo' entrare

T1T2waiting--; // Riduce # scrittori in attesa

VC & wait T1T2); // Risveglia uno scrittore

}

VC & mux); // Esce dalla mutua esclusione

Lf

-

⚠ Achtung

- Solo l'ultimo lettore puo' svegliare uno scrittore in attesa

- Il controllo "if (T3T4 == 0 && T1T2waiting > 0)" assicura che non si verifichino accessi simultanei.