SISTEMI OPERATIVI

Gestione del Processore
Sincronizzazione dei Processi

Lezione 2 – Variabili di lock

Vincenzo Piuri

Università degli Studi di Milano

Sommario

Sincronizzazione di processi concorrenti

Approcci a livello di istruzioni

- Variabile di turno
- Algoritmi per la sincronizzazione mediante turno
- Variabile di lock
- Supporti hardware per le variabili di lock

Variabile di turno

Variabile condivisa che definisce

il turno di uso della risorsa

cioè a quale processo spetta il diritto di uso in un certo istante.

Sincronizzazione di due processi concorrenti mediante variabile di turno

Vediamo tre approcci per l’accesso

in mutua esclusione alla rispettiva sezione critica

mediante una variabile di turno

Algoritmo 1

```
public class Algorithm_1 implements MutualExclusion
{
        private volatile int turn;

        public Algorithm 1() {
                turn = TURN 0;
        }
        public void enteringCriticalSection(int t) {
                while (turn != t)
                        Thread.yield();
        }
        public void leavingCriticalSection(int t) {
                turn = 1 - t;
        }
}
```

Garantisce mutua esclusione
Impone stretta alternanza dei processi
Non garantisce progresso

Algoritmo 2

```
public class Algorithm_2 implements MutualExclusion
{
        private volatile boolean flag0, flag1;
        public Algorithm 2() {
                flag0 = false; flag1 = false;
        }
        public void enteringCriticalSection(int t) {
                 if (t == 0) {
                        flag0 = true;
                        while(flag1 == true)
                                Thread.yield();
                }
                else {
                        flag1 = true;
                        while (flag0 == true)
                                Thread.yield();
                }
        }
        public void leavingCriticalSection(int t) {
                    if (t == 0)  flag0 = false; else flag1 = false;
        }
}
```

Non impone stretta alternanza dei processi
Non garantisce progresso
Possibile attesa infinita

Algoritmo 3

```
public class Algorithm_3 implements MutualExclusion
{
        private volatile boolean flag0;
        private volatile boolean flag1;
        private volatile int turn;
        public Algorithm_3() {
                flag0 = false;
                flag1 = false;
                turn = TURN_0;
        }
        public void enteringCriticalSection(int t) {
                int other = 1 - t;
                turn = other;
                if (t == 0) {
                        flag0 = true;
                        while(flag1 == true && turn == other)
                                Thread.yield();
                }
                else {
                        flag1 = true;
                        while (flag0 == true && turn == other)
                                Thread.yield();
                }
        }
        public void leavingCriticalSection(int t) {
                 if (t == 0)  flag0 = false; else flag1 = false;
        }
}
```

Garantisce mutua esclusione
Garantisce progresso

Variabile di lock

Variabile condivisa che definisce

lo stato di uso di una risorsa

cioè quando è in uso da parte di un processo

(cioè quando un processo è nella sua sezione critica).

Lock = 0 → risorsa libera

= 1 → in uso

Uso ad interruzioni disabilitate

Acquisizione della risorsa:

- disabilito le interruzioni
- leggo la variabile di lock
- se la risorsa libera (lock=0), la marco in uso ponendo lock=1 e riabilito le interruzioni
- se la risorsa è in uso (lock=1), riabilito le interruzioni e pongo il processo in attesa che la risorsa si liberi

Rilascio della risorsa:

- pongo lock=0

Hardware per la sincronizzazione

Istruzione atomica

TEST-AND-SET

- legge la variabile di lock e la pone in un flag del processore
- pone lock = 1
- se il flag (= vecchio valore di lock) vale 0, la risorsa era libera, altrimenti era già occupata e il processo deve attendere

In sintesi

- **Abbiamo visto**:

  il concetto e l'uso

  di variabili di turno e di lock

  per realizzare la mutua esclusione
  nell'accesso alle rispettive sezioni critiche
  di processi concorrenti per risorse condivise

- **Ricordiamo che questi sono**

  approcci a livello di istruzione

  e quindi richiedono grande attenzione da parte del programmatore per un uso corretto