Mutex - Critical section Problem

Consideriamo un sistema che consiste dei processi $\{P_0, P_1, \ldots, P_n\}$; ogni processo ha una sezione di codice chiamata "sezione critica" dove il processo potrebbe cambiare variabili condivise, aggiornare tabelle, scrivere su file...

Quindi, quando un processo è nella sua sezione critica, nessun altro processo è abilitato ad entrare nella propria. Questo vuol dire che non ci sono mai due processi che eseguono la propria sezione critica allo stesso momento.

Il termine MUTEX, ovvero mutual exlusion, indica un processo di sincronizzazione fra processi o thread CONCORRENTI con cui si impedisce che più task paralleli accedano contemporaneamente ai dati in memoria.

Implementazioni

L'implementazione più comune è sicuramente quella dei **Monitor**:

I monitors sono un'ASTRAZIONE ad alto livello che forniscono un meccanismo per la sincronizzazione tra processi. Un **monitor type** presenta delle operazioni definite dal programmatore che consentono la mutua esclusione.

Al contrario dei **semafori** i monitors provvedono automaticamente a fornire la mutua esclusione; infatti se un semaforo non viene usato correttamente dal programmatore, potrebbe malfunzionare.

Sintassi

```
monitor monitor_name {
    procedura P1 (...) {...}
    procedura P2 (...) {...}
    ...
    inizializzazione code (...) {...}
}
```

- Una procedura inizializzata all'interno di un monitor può accedere solo alle variabili dichiarate localmente all'interno del monitor.

- Inoltre, le variabili dichiarate localmente al monitor possono essere accese solo dalle procedure locali.

- Di conseguenza, il monitor si assicura che solo una procedura alla volta acceda allo stesso file, e quindi solo una proc può essere attiva alla volta.

Variabili condition - monitors

In ordine di ottenere la sincronizzazione tra processi con i monitors, dobbiamo definire le variabili condition.

Vengono dichiarate nel seguente modo: condition x, y

Le uniche operazioni che possono essere invocate sulle variabili condition sono: x.wait() x.signal();
l'operazione x.wait() significa che il processo che invoca questa operazione è sospeso finché un altro processo non invoca x.signal().

Quindi, se un processo vuole usare la variabile x, non potrà usarla finché un altro processo non la rilascia. Quando il processo che stava usando quella variabile rilascia la variabile, un altro processo che l'ha richiesta potrà usarla.
Se non ci sono processi sospesi, non viene ripreso nulla.

TLB Translation look-Aside Buffer

Il TLB è un buffer per le traduzioni; questo tipo di buffer è particolarmente piccolo, a differenza della cache che invece, essendo più grande, è usata per salvare dati.

Il TLB è contenuto all'interno del MMU; sappiamo che con la memoria virtuale, lo spazio viene organizzato in Pagine; queste pagine vengono acceso tramite la Tabella delle pagine. Il problema, però, è che questa tabella può risultare a volte lenta, quindi il TLB non è altro che una forma di cacheing della tabella.

All'interno del TLB è solitamente presente un'altra tabella (molto più piccola dell'originale) che mappa gli indirizzi virtuali a quelli fisici.

Come funziona?
Quando viene ricercata una chiave (indirizzo virt.) se questa è presente all'interno del TLB si accede direttamente alla memoria fisica. Se invece questa non è presente la si va a ricercare all'interno della tabella delle pagine (operazione assai più lunta) e la si copia nel TLB qualora servisse nuovamente.