Esecuzione Concorrente di Processi

Buongiorno! Ti fornirò una spiegazione sui concetti principali dell'esecuzione concorrente di processi nei sistemi operativi.

# Mutua esclusione e sincronizzazione

La **mutua esclusione** garantisce che quando un processo accede a una risorsa condivisa, nessun altro processo possa accedervi contemporaneamente. È fondamentale per evitare condizioni di gara (race condition).

La **sincronizzazione** coordina l'esecuzione di processi che devono operare in un ordine specifico o devono attendere che altri processi completino determinate operazioni prima di procedere.

# Differente velocità di esecuzione dei processi

I processi possono eseguire a velocità diverse per vari motivi:

* Diversa priorità assegnata dal sistema operativo
* Differenti carichi di lavoro della CPU
* Interruzioni di sistema
* Differenze hardware (in sistemi distribuiti)

Questa variabilità rende cruciali i meccanismi di sincronizzazione, poiché non possiamo fare affidamento su tempistiche predeterminate.

# I semafori

I **semafori** sono variabili intere utilizzate per controllare l'accesso a risorse condivise.

* **Semaforo binario**: può assumere solo i valori 0 o 1 e viene utilizzato per implementare la mutua esclusione
* **Semaforo contatore**: può assumere valori ≥ 0 e viene utilizzato per controllare l'accesso a risorse con più istanze

Le operazioni principali sui semafori sono:

* **P() o wait()**: decrementa il valore del semaforo; se il valore diventa negativo, il processo viene bloccato
* **V() o signal()**: incrementa il valore del semaforo; se ci sono processi in attesa, uno viene sbloccato

# Monitor e scambio di messaggi

Il **monitor** è una struttura di sincronizzazione di alto livello che incapsula dati e procedure in un unico modulo. Solo un processo alla volta può essere attivo all'interno di un monitor, garantendo automaticamente la mutua esclusione.

Lo **scambio di messaggi** è un meccanismo di comunicazione tra processi basato sull'invio e ricezione di messaggi. Le operazioni principali sono:

* **send(destinazione, messaggio)**: invia un messaggio a un processo
* **receive(mittente, messaggio)**: riceve un messaggio da un processo

Lo scambio di messaggi è particolarmente utile nei sistemi distribuiti.

# Lo stallo (deadlock)

Lo **stallo** si verifica quando due o più processi sono bloccati in attesa reciproca di risorse possedute dall'altro, creando un ciclo di attesa.

Le **condizioni necessarie** per lo stallo (condizioni di Coffman) sono:

1. **Mutua esclusione**: le risorse non possono essere condivise
2. **Possesso e attesa**: i processi possono mantenere risorse mentre ne richiedono altre
3. **Non prelazione**: le risorse non possono essere forzatamente rilasciate
4. **Attesa circolare**: esiste un ciclo di processi in attesa

Le **strategie** per gestire lo stallo sono:

* **Prevenzione**: negare almeno una delle condizioni necessarie
* **Evitamento**: fare scelte di allocazione che prevengano lo stallo (algoritmo del banchiere)
* **Rilevamento e recupero**: permettere che si verifichi, rilevarlo e risolverlo
* **Ignorare il problema**: adatto solo per sistemi dove lo stallo è raro e il riavvio è accettabile