Le risorse

Cos'è una risorsa?

E' qualcosa di necessario al processo per poter procedere nella sua evoluzione (passaggi di stato).

Si possono distinguere:
- Risorse **fisiche** (processore, memoria, disco, stampante,...)
- Risorse **astratte** (un file, una struttura dati,.....)

Le risorse possono essere assegnate al processo
- Per tutta la durata del processo, ***in modo statico,*** e vengono recuperate dal S.O. solo quando il processo termina.
- Durante l'evoluzione del processo, ***in modo dinamico***. In questo caso il processo chiede la risorsa una o più volte quando gli necessita e la rilascia dopo averla utilizzata.

Se il processo *può essere forzato* a rilasciare una risorsa, la risorsa è chiamata **prerilasciabile**.

Il processore è una risorsa gestita in modo dinamico

Le risorse possono anche esistere in **più copie** (classi di risorse omogenee), come i blocchi di memoria, insiemi di settori sul disco, diverse stampanti con caratteristiche comuni ecc…. Il S.O. deve decidere ***quale*** risorsa di una stessa classe assegnare al processo.

Ogni risorsa o classe di risorse viene **descritta da tabelle** con le informazioni sullo stato: se è *libera* o *occupata*, a quale processo è assegnata, ecc….. Queste tabelle sono tenute aggiornate dal S.O.

La gestione delle risorse è uno dei compiti più difficili del S.O.
Per questo motivo il S.O. è costituito da diverse parti (strati) ognuno dei quali si occupa alcune tipologie di risorse:

- La risorsa CPU è gestita dal ***gestore del processore*** (parte del nucleo).
- La risorsa memoria RAM è gestita dal ***gestore della memoria***.
- Le periferiche sono gestite dal **gestore delle periferiche** (tastiera, mouse, hard disk, monitor, stampante)
- I file e le directory sono gestiti dal ***gestore delle informazioni*** o ***File System***.

Il Gestore del processore**:** Decide a quale processo assegnare la CPU.
Recupera la CPU quando si rende disponibile.
Gestisce le comunicazioni fra i processi.

Il Gestore della memoria **:** Aggiorna lo stato della aree libere e occupate.
Sceglie quali aree allocare (assegnare) a ogni processo.
Le alloca e le recupera quando il processo non ne ha più bisogno.

Il Gestore delle periferiche **:** Assegna la periferica e inizia i processi di I/O.
Recupera la periferica.

Il Gestore delle informazioni**:** Gestisce le FAT (File Allocation Table).
Gestisce i privilegi.
Assegna spazio di disco ai processi (p.es. per creare un file).
Recupera spazio (p.es. per cancellare un file).
Crea e cancella file o directories.

Politiche di assegnazione delle risorse

In genere una risorsa può essere assegnala ad un solo processo per volta, in **mutua esclusione**.
Se più processi chiedono contemporaneamente la stessa risorsa il S.O. deve gestire le richieste con una opportuna politica di assegnazione, stabilendo un ordine tra i processi. Se la risorsa chiesta da un processo non è disponibile il processo passa in *stato di attesa* fino a quando può ottenere la risorsa (richiesta bloccante); per alcune categorie di risorse la richiesta può essere gestita in modo non bloccante, cioè se il processo non può ottenere la risorsa non viene sospeso ma viene solo informato dell'esito negativo.
Iprocessi sono in **competizione** per l'uso delle risorse e quindi **interferiscono** tra di loro.

IIS.O. deve assegnare le risorse ai processi in modo che vengano usate in mutua esclusione; per realizzare questo usa i meccanismi del modello concorrente (per esempio semafori per l'accesso alle risorse).

> **La politica di assegnazione** delle risorse stabilisce un **ordine** tra i processi, in modo da garantire che la risorsa sia assegnata in **mutua esclusione** e che **tutti i processi** che la richiedono riescano ad **ottenerla in un tempo finito**.

Il tempo di attesa di ogni processo per ottenere la risorsa dipende dalle richieste degli altri processi e dal tipo di politica di gestione adottata; se non si può stabilire un limite al tempo di attesa si può verificare il caso di **attesa indefinita** (o **starvation**) in cui il processo potrebbe non riuscire mai ad ottenere la risorsa.

Tipiche **politiche** di assegnazione sono:

- **in ordine di arrivo** (FCFS - First Come First Served) che soddisfa le richieste nell'ordine in cui sono arrivate; viene realizzata con una coda FIFO (First In First Out) e assicura un valore prevedibile al tempo di attesa poiché l'attesa di ogni processo dipende dal numero delle richieste che la precedono nella coda;
- **a priorità** che soddisfa le richieste in base a un valore di priorità associato a ogni processo;

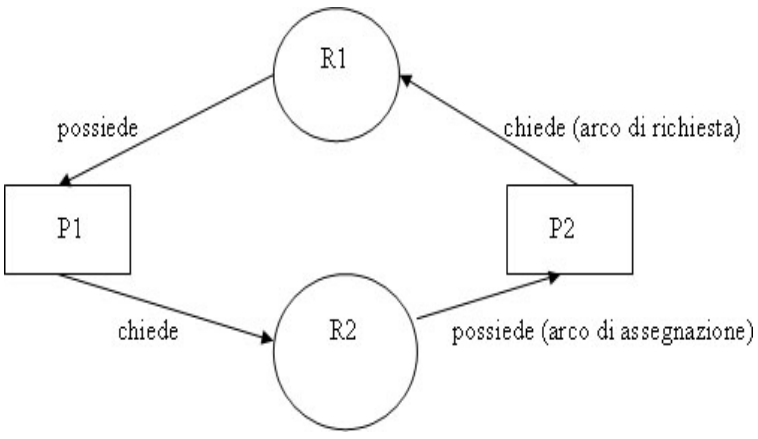
  per evitare il problema dell'attesa indefinita, la priorità viene variata in modo dinamico, aumentando la priorità delle richieste che sono in attesa da molto tempo.

Lo stallo

La presenza di più programmi che competono per l'uso delle risorse può causare situazioni di **stallo o blocco critico** (**deadlock**), cioè situazioni in cui non è possibile far avanzare nessun processo.

> Una situazione di **stallo** si genera quando un processo o più processi rimangono ***indefinitamente*** **bloccati** a causa del non verificarsi delle condizioni necessarie per il loro proseguimento.

Si consideri il seguente problema. Due processi **P1** e **P2** utilizzano due risorse R1 e R2; ciascun processo necessita di entrambe le risorse per arrivare a termine. Le due risorse sono acquisite una alla volta e utilizzate in modo esclusivo. Si supponga che ad un certo instante il processo **P1** possieda la risorsa R1 e il processo **P2** la risorsa R2. Se **P1** mantenendo il controllo di R1 chiede R2, esso viene sospeso in attesa della liberazione della risorsa da parte di **P2**; tuttavia se **P2**, mantenendo il controllo di R2, chiede la risorsa R1, entrambi i processi vengono bloccati indefinitamente (**attesa circolare**) .



Più in generale dato un insieme di processi P1,P2,…Pn che usano un insieme di risorse R1,R2,…Rm si può verificare una situazione di deadlock se sono vere contemporaneamente le seguenti condizioni:

a) Ciascuna risorsa può essere usata da un solo processo alla volta (**mutua esclusione**)
b) I processi trattengono le risorse che già possiedono mentre chiedono risorse addizionali
c) Le risorse già assegnate non possono essere sottratte (**senza prerilascio)**
d) Esiste una situazione di attesa che si chiude in se stessa (**attesa circolare**); ossia alcuni processi formano una catena chiusa tale che ciascun processo della catena controlla almeno una delle risorse richieste dal successivo.

Il S.O. può gestire le situazioni di stallo in due modi:

- **prevenendo** il verificarsi delle condizioni che portano allo stallo
  - Una **prima tecnica di prevenzione** consiste nell'imporre a ciascun processo di richiedere in blocco tutte le risorse di cui necessita; se le risorse non sono tutte disponibili al momento della richiesta il processo non ne ottiene alcuna e viene posto in uno stato di attesa fino a quando tutte le risorse sono disponibili. Dopo averle usate il processo può rilasciare le risorse in un ordine qualsiasi.
    Si noti che la condizione b) non è più soddisfatta
  - Una **seconda tecnica di prevenzione** prevede che se una richiesta per una risorsa viene rifiutata, il processo debba rilasciare tutte le risorse che aveva già acquisito e sospeso; non è più soddisfatta la condizione c)
  - Una **terza tecnica di prevenzione** prevede che a ciascuna risorsa sia assegnato un numero d'ordine e che i processi possano richiedere le risorse una per volta per numero d'ordine crescente. In questo caso non si verifica più una catena circolare, condizione d).
- **riconoscendo** che si è verificato uno stallo e **risolvendo** il problema
  - Il modo più drastico consiste nell'interrompere tutti i processi in stallo
  - Un metodo migliore consiste nell'interrompere un processo alla volta, finché non si risolve la situazione

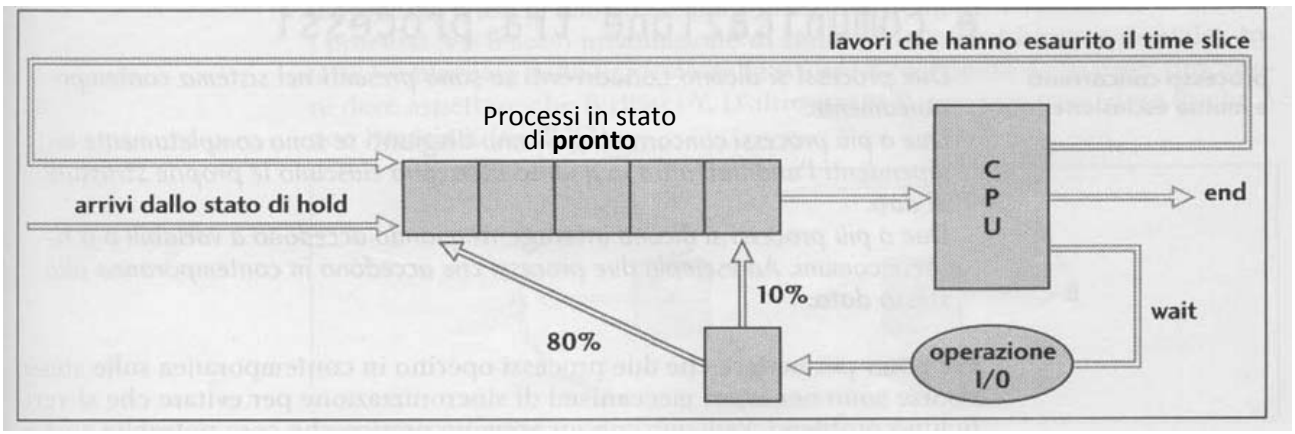
LA SCHEDULAZIONE DEL PROCESSORE (o schedulatore a breve termine = scheduler della CPU)

Riprendiamo l'argomento della **schedulazione dei processi** e in particolare vediamo alcune delle possibili politiche per la scelta del processo che dallo stato di "pronto" (ready) deve passare in stato di "esecuzione" (run) con la conseguente assegnazione del processore al processo scelto.

Politiche di schedulazione del processore (con prerilascio)

- **round robin:** il processore viene assegnato a turno, per un intervallo di tempo stabilito (**time slice o quanto di tempo**), ai processi nell'ordine in cui questi ne hanno fatto richiesta.
  La **coda** è gestita con il metodo FIFO e quindi tutti gli inserimenti provenienti dalla coda di hold, dalla coda di wait o quelli causati dall'esaurimento di un time slice precedente avvengono al fondo della coda

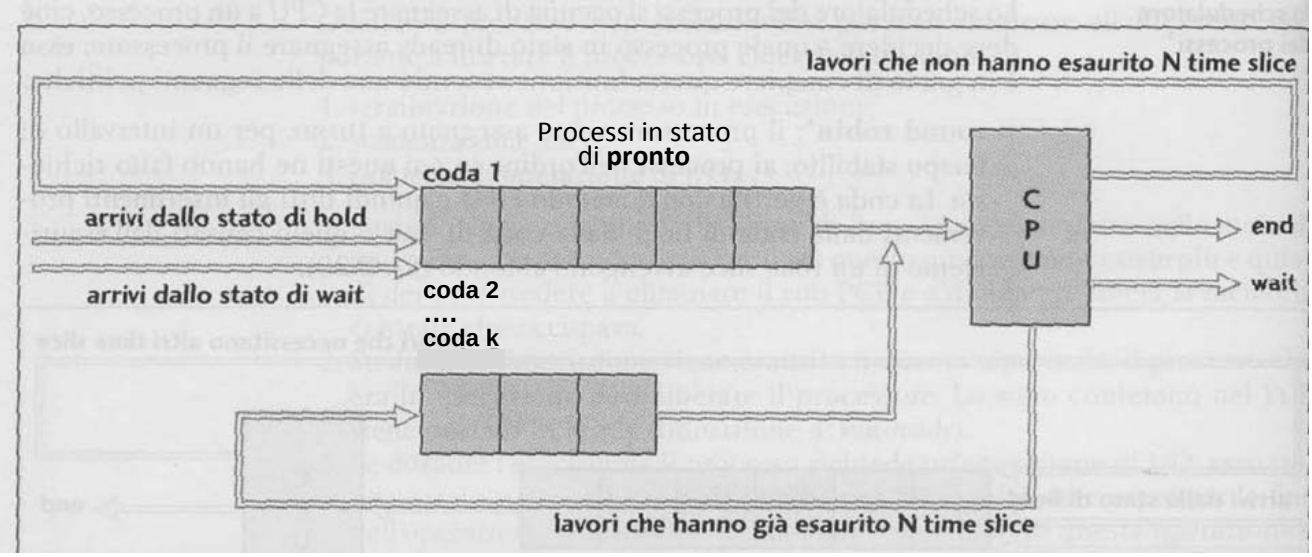
- **round robin a percentuale di tempo:** con la tecnica precedente i processi che richiedono molte operazioni di I/O sono penalizzati rispetto a quelli che ne richiedono di meno perché il rientro dallo stato di wait significa l'inserimento del processo al fondo della coda. La tecnica a percentuale di tempo è una variante del round robin: il processo non verrà inserito necessariamente al fondo, ma la posizione di rientro nella coda dipende dalla percentuale di tempo di CPU utilizzata. Questo significa che se un processo lascia il processore perché richiede un I/O, e quindi non ha ancora esaurito il proprio time slice, il suo rientro nella coda non sarà più al fondo ma tanto più avanti quanto minore è la percentuale di tempo di CPU utilizzata prima dell'interruzione.



- **round robin con priorità statica:** a ogni processo viene assegnata una priorità in fase di generazione, ad esempio in base al tempo presunto di esecuzione. Lo schedulatore dei processi terrà conto di questo valore per inserire il processo nella coda

- **round robin con priorità dinamica e code a diversi livelli di priorità:** il S.O. modifica la priorità in base al tempo di utilizzo della CPU; ai processi che rilasciano la CPU prima dello scadere del quanto temporale viene aumentato il valore della priorità.
  I processi vengono organizzati in attesa su diverse code, corrispondenti alle diverse priorità assegnate a ciascuno, e al momento di scegliere il processo da mandare in esecuzione

vengono favoriti quelli a priorità più elevata. Quando le code con priorità superiore sono vuote vengono scelti i processi delle code con priorità più bassa. L'esecuzione di un processo a priorità inferiore inoltre può essere interrotta per provvedere all'esecuzione di processi a priorità più alta che nel frattempo sono passati nello stato di "pronto".
Per evitare che i processi con priorità più bassa non ricevano mai il processore o siano interrotti frequentemente senza riuscire ad arrivare dallo stato di "esecuzione" allo stato di "terminazione", il S.O. aumenta la loro priorità tenendo conto del tempo di permanenza in stato di "pronto".



Passaggi di stato e le interruzioni

Affinché un processo possa transitare da uno stato all'altro occorre che si verifichino determinati eventi. Tali eventi devono essere comunicati all'unità di controllo della CPU tramite apposite linee hardware, utilizzando il **meccanismo delle interruzioni**.

Un'**interruzione** (o **interrupt**) è una segnalazione che arriva alla CPU per comunicarle qualcosa e può essere di natura **software o hardware.**

Si parla di **interruzioni software** quando è lo stesso processo in esecuzione che invia un'apposita istruzione nella quale specifica il tipo di comportamento che richiede alla CPU.
Si parla di **interruzioni hardware** quando la segnalazione di richiesta di interruzione arriva da parte di una **periferica** di input/output tramite un **segnale elettrico**. Tale segnale viene inviato su una linea dedicata a questo tipo di interruzioni, e giunge direttamente su un piedino del microprocessore.
Un'interruzione hardware è quindi, una segnalazione di natura tipicamente **asincrona** (cioè la CPU non è a conoscenza del momento in cui l'interruzione si verificherà); di conseguenza, tale segnalazione, proveniente dall'esterno del processore, non è sincronizzata con le attività che la CPU sta svolgendo in quell'istante.
Esempi di interruzioni asincrone sono: la fine di un processo di stampa, la pressione di un tasto sulla tastiera, un'anomalia aritmetica del processo in esecuzione (ad esempio, un overflow, un underflow o una divisione per zero).
Un altro tipo di interruzione è quella **generata dal clock del sistema** che causa lo **scadere del *quanto* di tempo** del processo in esecuzione. Se, ad esempio, un quanto è pari a 100 cicli di clock, allo scadere di ogni 100 cicli viene generata un'interruzione hardware di "quanto scaduto". Queste interruzioni sono, però, di natura **sincrona,** poiché sono eventi interni al processore, previsti e sincronizzati con le attività del processore stesso.

Gestione delle interruzioni

Quando la CPU rileva la presenza di un'interruzione (hardware o software) deve:

- **completare** l'istruzione in corso del processo in esecuzione;
- **salvare** il contesto in cui stava operando (**cambio di contesto o context switch**). In particolare deve salvare in un'apposita area tutte le informazioni necessarie a descrivere la situazione raggiunta nell'istante immediatamente prima che si verificasse l'interruzione.
- **eseguire** una particolare routine per la gestione di quel particolare tipo di interruzione
- **ripristinare**, infine, il contesto salvato e **riprendere** il processo dall'istruzione successiva a quella in cui era stato interrotto (non è detto che questo avvenga immediatamente dopo la terminazione della routine di interrupt, in quanto la CPU potrebbe essere stata assegnata nel frattempo a un altro processo che era presente nella lista in stato di "pronto" ed è passato allo stato di "esecuzione").

Context switch

Il passaggio dell'esecuzione da un processo ad un altro *richiede quindi tempo* per il salvataggio ed il caricamento dei registri e delle mappe di memoria, aggiornamento di tabelle e liste ecc.
Questa operazione, chiamata **context switch,** avviene in diverse occasioni per motivi diversi; uno dei possibili eventi che comportano un context switch è l'esaurimento del ***quanto di tempo*** che è stato assegnato ad un processo in esecuzione. Possiamo allora fare delle considerazioni sulla **durata** del **quanto di tempo.**

La durata del **quanto di tempo** (time slice):

- **non deve essere troppo breve** per evitare: **molti context switch** tra i processi con la conseguente **riduzione dell'efficienza della CPU**
- **non deve essere troppo lunga** per evitare: **tempi di risposta lunghi** per i processi interattivi con tempo di esecuzione breve che verrebbero altrimenti penalizzati dai processi che richiedono un tempo di esecuzione maggiore.