I processi

Cos'è un processo?

Un processo è una attività, controllata da un programma, che si svolge su un processore.

- Il **programma** è una entità *statica* che descrive la sequenza di istruzioni che devono essere svolte.
- Il **processo** è una entità *dinamica*, che rappresenta la particolare esecuzione di un programma, quindi evolve nel tempo.
- Il **processore (o CPU):** è la principale risorsa del sistema di elaborazione che consente l'evoluzione di un processo.

Per esempio la ricetta di cucina può essere considerato un programma che diventa un processo quando, assegnate tutte le risorse (utensili e ingredienti), sarà effettivamente eseguita dal processore (il cuoco). In realtà si possono generare più processi: l'approvvigionamento degli utensili e degli ingredienti (eventuale acquisto), preparazione della ricetta scomposta in diverse fasi, presentazione finale della pietanza.

In linea di massima un processo può essere definito come un programma in esecuzione composto dalla coppia (E,S)

- E: **codice eseguibile**
- S: **stato** del processo

Un programma può dare origine a diversi processi. Un certo processo è associato ad un unico programma.

Affinché un processo avanzi è necessaria l'assegnazione delle risorse a lui necessarie, fra cui la **CPU**.
Se la CPU è una sola, solo uno fra i processi esistenti è nello stato di esecuzione. Gli altri sono in qualche altro stato, ma non sono in esecuzione.
Normalmente (nei sistemi multiprogrammati) sono presenti sulla macchina decine di processi (osserva il task manager di Windows).
Un processo nasce e occupa le risorse del sistema. In particolare occupa la RAM, che è il punto di partenza per potere andare in esecuzione. Infatti la CPU è in grado si eseguire codice solo se quest'ultimo è in RAM.
I processi del kernel sono sempre in RAM, essendo "pronti" per essere eseguiti.

La componente principale dell'elaboratore, e anche la più costosa, è la CPU (Central Processing Unit) che è un esecutore sequenziale di operazioni. Tutti i programmi hanno bisogno della CPU per essere eseguiti, ma questa è unica e quindi viene "contesa" dai vari processi in esecuzione. Riveste allora particolare importanza quella parte del sistema operativo chiamato **gestore dei processi** che si occupa dell'assegnazione della CPU ai singoli processi che ne richiedono l'uso. Quindi la gestione dei processi è affidata a quei moduli del sistema operativo che formano il nucleo e che sono sempre residenti in memoria centrale. Il nucleo, chiamato anche **Kernel** (nocciolo), è la parte del sistema operativo più vicina alla macchina ed è strettamente dipendente dall'hardware. Le funzioni fondamentali del nucleo sono:

- Gestire il ciclo di vita dei processi: avvio e terminazione dei processi
- assegnare la CPU ai diversi processi: selezionare quale tra i processi in stato di "pronto" deve essere mandato in esecuzione
- sincronizzare i processi: gestire la cooperazione e la concorrenza tra i processi nell'accesso alle risorse
- sincronizzare i processi con l'ambiente esterno

Il nucleo comprende tutte le **routine di risposta alle interruzioni** e le **procedure che assegnano la risorsa CPU** ai diversi processi. Le norme che regolano queste assegnazioni vengono chiamate **politiche di scheduling.** Fanno quindi parte del nucleo:

- lo **schedulatore dei lavori;**
- lo **schedulatore dei processi;**
- il **controllore del traffico;**
- il **gestore delle interruzioni**;
- le **procedure di sincronizzazione e comunicazione** tra più processi necessarie per lo scambio di dati e informazioni.

Ogni elaboratore ha una propria configurazione hardware, così come ogni sistema operativo ha differenti caratteristiche. La configurazione (hardware e software) che si trova più di frequente è quella di un sistema di elaborazione **multiprogrammato** e **interattivo**, dotato di **un solo processore centrale**, di uno o più processori dedicati (canali di I/O) e delle periferiche standard (dischi, stampanti….)

In **memoria centrale** (RAM) in un dato istante possiamo trovare un certo numero di processi uno dei quali *sta evolvendo*, altri sono *pronti* per essere serviti, altri devono *attendere* la terminazione di operazioni di I/O. Altri ancora sono momentaneamente *parcheggiati* su memoria di massa. L'esecuzione di un programma genera almeno un processo, costituito dalla sequenza delle operazioni del programma. Un processo che deve essere eseguito può trovarsi, in ogni istante, in uno "**stato**", cioè in una condizione particolare determinata dalle esigenze del processo stesso e dallo stato globale del sistema (cioè le altre componenti hardware e software).

> Ogni **stato corrisponde a una situazione** in cui si trova il programma da eseguire (è in esecuzione, è in attesa della stampante, è in attesa di un dato da inserire da tastiera ecc.)
>
> Il **passaggio da uno stato all'altro** è deciso dal **sistema operativo** sulla base delle **politiche di schedulazione**. A ogni passaggio corrisponde una transazione che è effettuata tramite l'esecuzione di uno specifico programma del sistema operativo (controllore del traffico).

Descriviamo il ciclo di avanzamento di un programma seguendone tutta la sua storia, dal momento in cui il lavoro è proposto al sistema a quello in cui termina la sua esecuzione.
Come esempio di processo che deve essere svolto si può considerare quello di effettuare un viaggio in aereo dove *la situazione in cui si trova il passeggero* viene considerato il suo ***stato*** ed è determinato non solo dalle esigenze del passeggero ma anche dalle condizioni esterne: ad esempio *si trova all'aeroporto* in attesa dell'imbarco in aereo, che può essere pronto o deve ancora arrivare ed è in ritardo, oppure il passeggero *è in volo*, oppure ancora *si trova in un aeroporto di transito* per uno scalo o, finalmente, *è arrivato* a destinazione.

Processo utente e supervisore

Un **processo utente** è un processo generato da un programma scritto dall'utente.

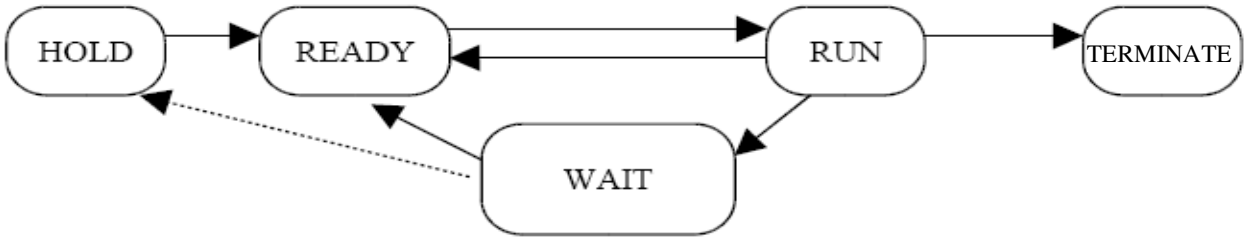
Un **processo supervisore** è un processo generato da un programma del Sistema Operativo.

Tra i processi in esecuzione sulla CPU ci sono quindi i processi utente, ma anche i **processi supervisore** del SO e questi ultimi hanno una **priorità** e una **importanza maggiore** dei processi utente
Ad esempio quando si chiede attraverso un programma scritto in C++ di acquisire un dato da tastiera con l'istruzione *cin …* oppure si visualizza un messaggio sul monitor con *cout* …, in realtà il nostro programma chiede l'intervento al sistema operativo per poter colloquiare opportunamente con le due periferiche menzionate, tastiera e monitor, e il processore entra in stato supervisore per trattare le richieste del nostro programma perchè comincia ad eseguire le routine preposte appunto alla gestione della tastiera e del monitor. Al termine dell'operazione richiesta potrà riprendere l'esecuzione del nostro programma (processo utente).

Gli stati di un processo

Gli **stati possibili** nei quali si può trovare un processo sono:

- **HOLD (parcheggio)**: il programma (chiamato job) è stato proposto al sistema e **attende di essere caricato in memoria centrale**. Esso è *in attesa di esecuzione* e *si trova in memoria di massa* .
- **READY (pronto)**: il programma è diventato **processo** e *si trova in memoria centrale*, pronto per essere eseguito. Esso **attende che gli venga assegnata la CPU**.
- **RUN (esecuzione)**: il **processo è in esecuzione** (in ogni istante un solo processo si trova in questo stato) perché gli è stata assegnata la CPU.
- **WAIT (attesa)**: il processo è in attesa (ad esempio deve attendere la fine di un'operazione di I/O).
- **TERMINATE (terminazione)**: il processo è terminato e può essere tolto dalla memoria centrale.



Stati di un processo

A ogni **stato** è associata una **lista** che contiene l'elenco di **tutti i processi** che si trovano in quel particolare stato in quel momento.

Transizione HOLD–READY

Avviene quando il lavoro viene caricato in memoria centrale. All'inizio, il Job scheduler, che ha ricevuto una richiesta di esecuzione di un programma, ha creato il **lavoro** (job) corrispondente al programma e lo ha messo in stato di Hold. Al momento opportuno lo **schedulatore dei lavori** sceglie tra i lavori in stato di Hold quello da **caricare nella memoria centrale**, se c'è un'area sufficientemente grande per contenerlo, passandolo così nello stato di Ready. La scelta del processo da portare in stato di "pronto" viene fatta in base all'ordine di arrivo dei processi oppure in base alla priorità dei processi o altro (politiche di schedulazione).

Transizione READY–RUN

Avviene quando al processo viene assegnato il **processore**; la scelta del processo da portare in stato di Run è fatta dallo **schedulatore dei processi** in base alla politica di schedulazione adottata dal gestore del processore.

Transizione RUN–READY

Questa transizione avviene quando il processo termina il time slice assegnatogli (time sharing), oppure quando arriva una interruzione esterna.

Transizione RUN–WAIT

Avviene quando il processo chiede un sevizio del S.O. ad esempio un'operazione di I/O (visualizzazione su monitor, richiesta di inserimento dati da tastiera, richiesta di una stampa ….)

Transizione WAIT–READY

Avviene quando un'interruzione esterna segnala il completamento dell'operazione per cui il processo è in attesa.

Transizione RUN–TERMINATE

Quando l'esecuzione del processo termina.

Transizione WAIT–HOLD

Non esiste sempre. E' la "schedulazione di medio livello", nella quale il S.O. decide di recuperare dello spazio di memoria "swappando" un processo su disco. E' una operazione lenta perché impone il salvataggio su disco dello stato del processo, e viene utilizzata più raramente possibile.

LO SCHEDULATORE DEI LAVORI ***(O SCHEDULATORE A LUNGO TERMINE)***

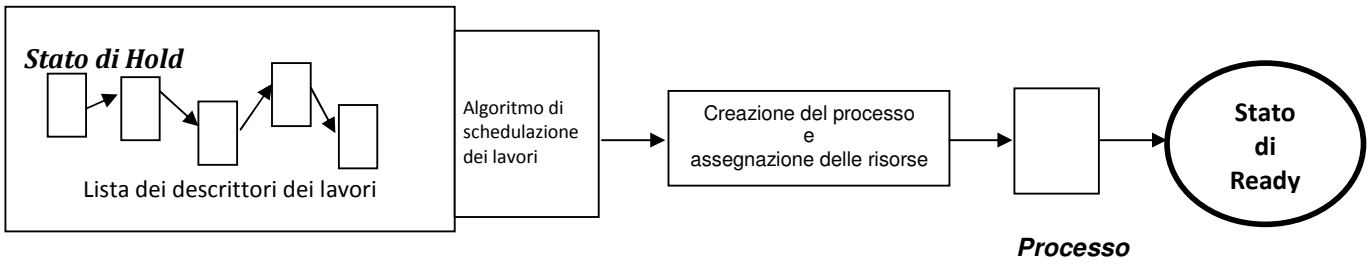
Determina in ogni istante il numero di lavori che risiedono in memoria centrale e quindi condiziona e controlla il grado di multiprogrammazione del sistema.

- Tiene aggiornate apposite *tabelle* che contengono le informazioni dello stato di ciascun lavoro (job)
- Sceglie ogni volta quale lavoro deve acquisire le risorse per avanzare, selezionandolo da una lista dei lavori in stato di *New*
- Quando un job viene scelto genera un processo
- Quando un processo termina sovrintende e coordina il recupero delle risorse assegnate a quel processo.

Per fare questo consulta i **descrittori dei lavori.**

Ogni **descrittore** contiene le seguenti informazioni:

- il *nome* del lavoro
- la *data e ora* di accettazione
- la *priorità*
- il *tempo* previsto di CPU
- *l'elenco delle risorse da assegnare staticamente* (dedicate per l'intero tempo di elaborazione)
- *l'elenco delle risorse da assegnare dinamicamente* (assegnate al momento dell'uso e rilasciate subito dopo)
- l'indicatore dello *stato di avanzamento* del lavoro (step). Dato che un lavoro può generare più processi viene indicato ogni processo già concluso e quello eventualmente in esecuzione.
- il puntatore al lavoro successivo nella lista



Politiche di schedulazione dei lavori

First In First Out: I lavori vengono scelti nell'ordine in cui sono giunti. In questo caso può succedere che i tempi di risposta siano imprevedibili e spesso c'è una scarsa ottimizzazione delle risorse.

Priorità statica: in questo caso viene assegnata ad ogni lavoro che arriva una priorità calcolata in base ad alcuni parametri, ad esempio al tempo presunto di utilizzo di alcune risorse. C'è però il rischio che alcuni lavori, quelli con priorità basse, restino troppo in attesa se arrivano continuamente lavori con priorità più alta.

Priorità dinamica: in questo caso viene assegnata ad ogni lavoro che arriva una priorità calcolata in base ad alcuni parametri, come nel caso precedente, ma con il crescere del tempo di attesa viene incrementata la priorità; si tiene quindi conto dell'anzianità del lavoro per evitare che rimanga troppo tempo nella lista.

SCHEDULATORE DEI PROCESSI *(O SCHEDULATORE A BREVE TERMINE)*

Determina in ogni istante il processo che deve essere portato da stato di pronto in stato di esecuzione, assegnandogli quindi la risorsa processore.

Per fare questo consulta i **descrittori dei processi (PCB - Process Control Block).**

Ogni **descrittore** contiene le seguenti informazioni:

- il *nome* del processo
- *l'indirizzo del descrittore del processo che l'ha generato* (es.: per un processo di I/O viene posto in questo campo l'indirizzo del processo che l'ha richiamato)
- *la lista degli indirizzi dei descrittori dei processi richiamati*
- *lo stato di avanzamento*
- *l'immagine nel processore* (contenuto dei registri di lavoro, del registro dei flags, del program counter,…)
- *l'immagine nella memoria* (le informazioni sulle aree di memoria assegnate al processo)
- *le informazioni relative alle altre risorse* assegnate al processo
- *il tempo* di CPU già usato, i limiti di tempo, …
- il puntatore al processo successivo nella lista

Tutti i descrittori dei processi che si trovano nello stesso stato di avanzamento sono collegati tra loro in modo da formare *liste bidirezionali multiple*.

Politiche di schedulazione dei processi

Esistono diversi algoritmi di schedulazione dei processi, ma sicuramente una buona politica deve considerare le seguenti necessità del sistema di elaborazione:

- *Rendere massimo il tempo di utilizzo della CPU*: il processore deve essere mantenuto attivo il più possibile, evitando che in un certo istante la lista dei processi in stato di pronto sia vuota;
- *Servire il maggior numero di processi nell'unità di tempo* rendendo quindi massimo il *throughput* di sistema e cioè il numero di processi completati nell'unità di tempo;
- *Valutare i tempi di esecuzione dei processi.* Viene misurato il tempo che un processo rimane attivo dal momento della sua generazione fino alla sua terminazione, tenendo conto sia dei tempi passati in attesa che di quelli impiegati nell'uso delle diverse risorse;
- *Rendere il più possibile limitato il tempo di attesa nello stato di pronto;*
- *Minimizzare il tempo di risposta.* In un sistema interattivo il tempo di risposta dipende sia dall'utente che fornisce i dati, sia da ciò che deve essere eseguito, sia dalla velocità di risposta dei dispositivi di output. Allora è sicuramente più significativo valutare il tempo che intercorre tra una singola richiesta e la prima risposta prodotta;
- *In un sistema multiutente, soddisfare il maggior numero di utenti;*
- *Minimizzare l'overhead di sistema (sovraccarico di sistema),* considerando che l'assegnazione della CPU da un processo ad un altro prevede sempre l'esecuzione della procedura di salvataggio dello stato del vecchio processo e il caricamento dello stato del nuovo processo (*context switch*).

L'algoritmo di rotazione tra i processi in esecuzione deve essere il più equo ed efficiente possibile.
C'è da tenere presente anche che per un corretto funzionamento del sistema è opportuno avere sia processi che usano molto la CPU, sia processi che usano molto le periferiche (I/O). In questo modo si evitano tempi di risposta inaccettabili per i programmi interattivi in un caso e inattività del processore nell'altro.

Anche lo schedulatore dei processi usa precisi criteri per operare la scelta di quale processo far avanzare.
Tratteremo successivamente alcune di queste politiche.