Immagine Processi

martedì 28 marzo 2023 17:27

Immagine di un processo

L'immagine di un processo è l'insieme di tutte le possibili informazioni che caratterizzano l'esercizio di un processo.

Essa è definita da:

- **Programma**

Programma di cui il processo risulta istanza.

- **DATI**

I suoi dati, che sono all'interno dell'address space.

- **Stack Area.**

Di supporto di gestione delle chiamate a funzione.

- **Stack Area di sistema**

Quando chiamiamo una System-Call il sistema utilizza un'altra area di memoria per andare a scrivere informazioni che riguardano il flusso d'esecuzione e determinare anche la posizione di variabili locali del software di sistema che stiamo eseguendo.

- **Collezione degli attributi.**

È una serie di informazioni di controllo comunemente chiamata PROCESS CONTROL BLOCK (PCB).

Sono informazioni che il software di sistema utilizza per andare a controllare l'esecuzione di questo processo.

Queste informazioni sono tipicamente rappresentate all'interno di blocchi della memoria.

Di tutte queste informazioni che rappresentano l'immagine di un processo, alcune sono sempre mantenute comunque in memoria principale, queste sono tutto ciò che riguarda la parte sistema.

Il PCB è sempre mantenuto all'interno della memoria di lavoro e lo stack di sistema.
Queste sono informazioni/aree di memoria che utilizziamo quando esegue il software di sistema.

- Tale immagine viene tipicamente rappresentata in blocchi di memoria (contigui o non) che possono risiedere o in memoria principale o sull'area di Swap
- Una porzione e' mantenuta in memoria principale per controllare efficientemente l'evoluzione di un processo anche quando esso risiede sull'area di Swap

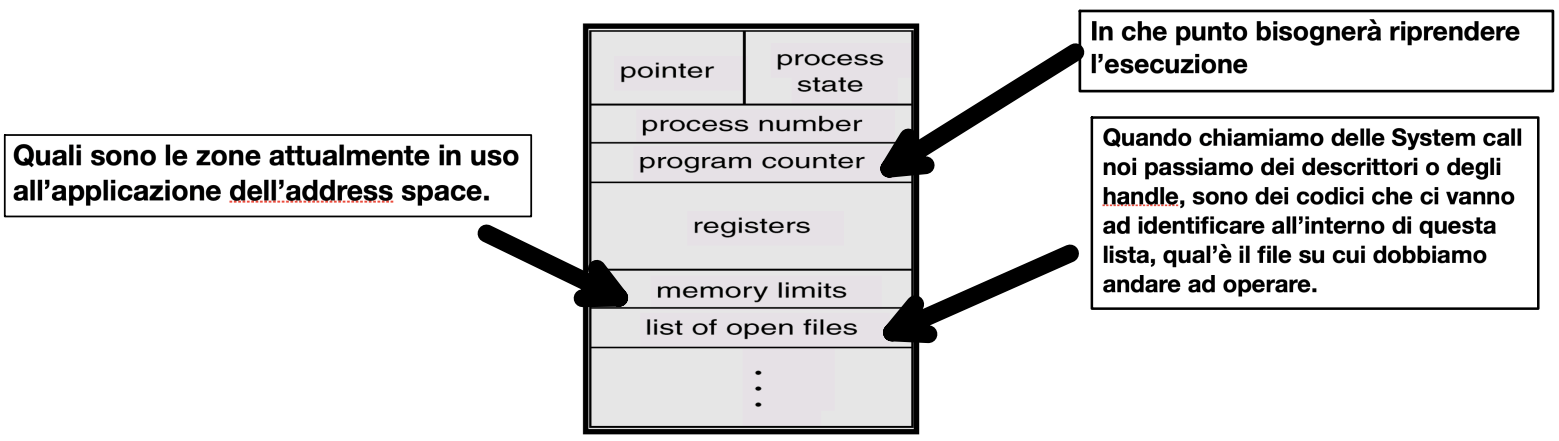
Tutto il resto è caricato in memoria secondaria.

PCB: ATTRIBUTI BASICI

Identificatori	Del processo in oggetto e di processi relazionati (padre, eventuali figli)
Stato del processo	Posizione corrente nel precedente diagramma di rappresentazione
Privilegi	In termini di possibilita' di accesso a servizi e risorse
Registri (contesto di esecuzione)	Contenenti informazioni associate allo stato corrente di avanzamento dell'esecuzione (es. il valore del program counter, i puntatori allo stack, i registri del processore)
Informazioni di scheduling	Tutto cio' che il sistema necessita di sapere per poter arbitrare l'assegnazione del processore ai processi che si trovano nello stato **Ready** (problema dello *scheduling della CPU*)
Informazioni di stato	Evento atteso

Il PCB è una tabella di informazioni alla quale il software può accedere tramite System Call!

Schema di un Process Control Block (PCB)



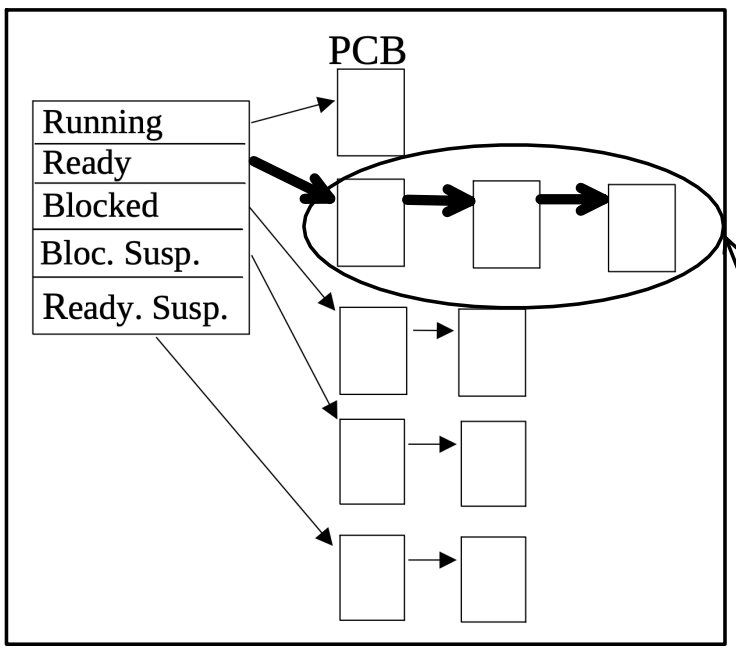
Le informazioni contenute variano a seconda delle implementazioni, ma in generale sono presenti:

- Program counter
- Area per il salvataggio dei registri general purpose, di indirizzamento
- Area salvataggio registro di stato
- Area di salvataggio per i flag
- Stato corrente di avanzamento del processo (Pronto, In Esecuzione, Bloccato)
- Identificatore unico del process

- Un puntatore al processo padre
- Puntatore ai processi figli se esistenti
- Livello di priorità
- Informazioni per il memory management(Gestione della memoria) (in particolare memoria virtuale) del processo
- Identificatore della CPU su cui è in esecuzione
- Informazioni per lo scheduling (gestione) del processo, come il tempo di run (esecuzione) o wait (attesa) accumulato o tempo stimato di esecuzione rimanente
- Informazioni di accounting di un processo
- Segnali che pendono
- Informazioni sullo stato di I/O del processo
- Registro nel quale è presente un puntatore alla page table

Durante la commutazione di contesto, è necessario salvare in memoria centrale lo stato di esecuzione del processo che viene fermato. Queste informazioni vengono memorizzate proprio nel PCB del processo, e sarà sempre dal PCB che esse verranno ricaricate quando si dovrà proseguire l'esecuzione.

Con la zona "pointer" del PCB, possiamo costruire liste di PCB!



Se vogliamo identificare TUTTE le applicazioni che sono attualmente nello stato "Ready", avremo sicuramente un primo puntatore che punterà al primo PCB di un'applicazione READY, che a sua volta ci permetterà di identificare un secondo PCB, un terzo PCB, e così via…

Quando il kernel del S.O deve eseguire un blocco di software per gestire le applicazioni e vuole andare ad identificare quali sono le applicazioni "ready", basta andare all'interno di quella lista.

All'interno dei PCB abbiamo le "informazioni di scheduling" che ci indicano le attività che devono essere eseguite dal software di sistema operativo per andare eventualmente ad assegnare la CPU alle nostre applicazioni, e per portare un processo da "Ready" a "Running", il software del Sistema va dentro quella lista e sceglie, e la scelta si basa proprio leggendo queste informazioni di scheduling dentro il PCB di ogni processo.

Cambio di contesto

Supponiamo di avere la CPU all'interno di un sistema e su questa CPU decidiamo che, sebbene prima su questa CPU avevamo in esercizio il processo A, adesso dobbiamo mandare in esercizio il processo B: stiamo cambiando il contesto d'esecuzione dal processo A, al processo B.

Quando abbiamo un cambio di contesto che ovviamente viene svolto dal software del sistema, ad esempio quando arriva un interrupt da timer, la CPU che era dedicata al processo A deve essere dedicata ad un altro processo, perché è scaduto il tempo.

Bisogna salvare il contesto corrente del processo che sta per lasciare la CPU e aggiornare il PCB del processo corrente definendo anche in che stato lo portiamo. Cambiando lo stato bisogna inserire il PCB nella lista adeguata, la lista associata a quello stato.

Ora bisogna selezionare il processo da schedulare, quindi scegliere tra gli altri processi, a quale di quesi assegnare la CPU. In questo caso aggiorniamo il PCB del processo che selezioniamo, indicando che lo mandiamo "running", e ripristino il contesto di questo processo schedulato, quindi rimettere in CPU le informazioni che avevamo

- salvataggio del contesto corrente
- aggiornamento del PCB del processo corrente (definizione dello stato)
- inserimento del PCB nella lista/coda adeguata
- selezione del processo da schedulare
- aggiornamento del PCB del processo schedulato (cambio di stato)
- ripristino del contesto del processo schedulato

salvato all'interno di questo PCB.

Cambio di modo di esecuzione

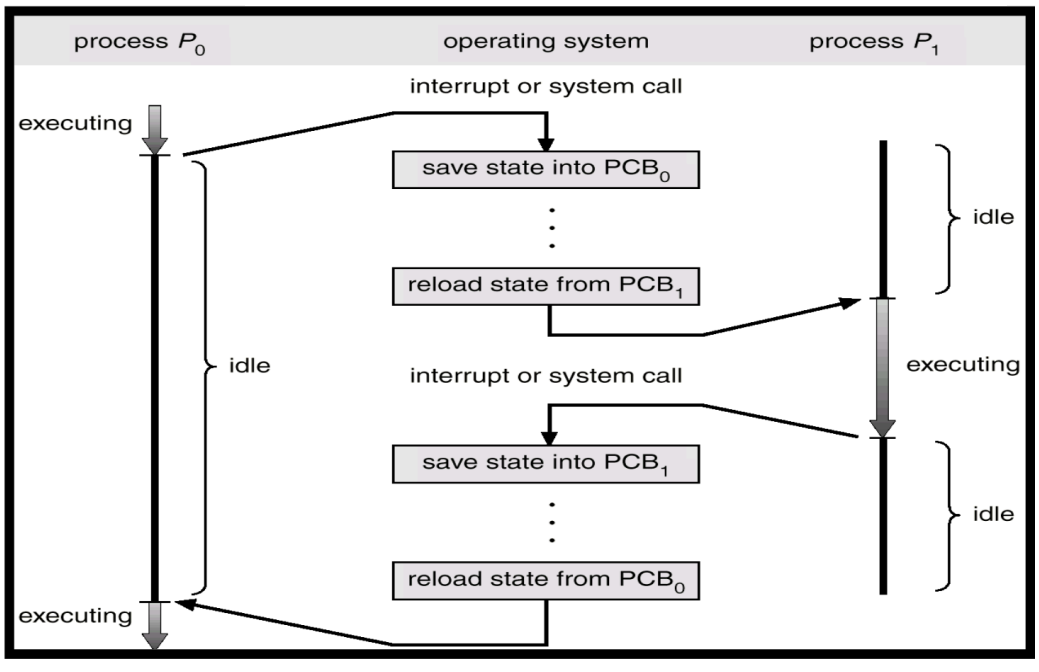
Quando noi accediamo a privilegi di livello superiore passando ad eseguire blocco di codice del Kernel.

Eravamo ad esempio in modalità user, stavamo compilando le istruzioni del programma che avevamo mandato in esercizio, e ad un certo punto passiamo ad eseguire qualche cosa che fa parte del Kernel.

Quand'è che avviene?
Tipicamente avviene quando noi chiamiamo una System Call!

Ma attenzione: in running c'è sempre la nostra applicazione.
Andiamo ad eseguire istruzioni che cambiano il Current Privilege Level (CPL), ad esempio l'istruzione sys_call!
Abbiamo possibilità di eseguire istruzioni che non sono ammesse in modalità utente.

ESEMPIO RIASSUNTIVO



● Nessuna implicazione diretta ●

Cambio di modo ⟺ **Cambio di contesto**

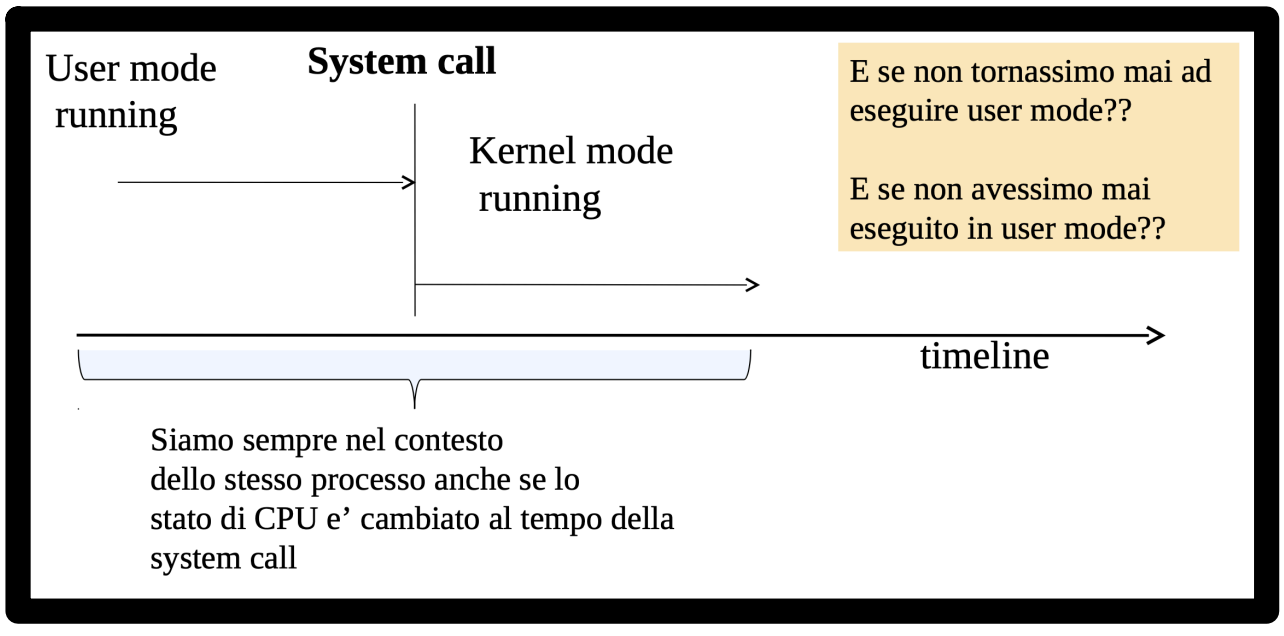
● **Cause di cambio di contesto**
- interruzione di clock (time-sharing), viene attivato lo scheduler per cedere il controllo ad un altro processo
- interruzione di I/O, con possibile riattivazione di un processo a piu' alta priorita'
- fault di memoria (per sistemi a memoria virtuale), con deattivazione del processo corrente

● **Cause di cambio di modo di esecuzione**
- attivazione di una funzione kernel
- gestione di una routine di interruzione

⇒ Salvataggio/ripristino di **porzione di contesto**

Si può avere un cambio di modo senza cambio di contesto: ad esempio sono a livello user, ho un cambio di modo e passo l'esecuzione a livello kernel, dopo ritorniamo ad eseguire la modalità user, al punto di esercizio che avevamo lasciato. Tutto questo senza cambio di contesto. L'applicazione lavora in modalità user-kernel.

Si può cambiare il contesto senza avere un cambio di modo, posso portare la cpu a lavorare da un processo A ad un processo B. Supponiamo di avere un'applicazione attiva in modalità user e poi tramite una system call scendiamo in modalità kernel, e se non tornassimo mai più ad eseguire user?
E se la CPU esegue solo in USER-MODE? Questi tipi di processi si chiamano DEMONI.
E quindi posso avere tanti demoni che prendono il controllo a livello kernel della CPU, perciò posso avere un cambio di contesto ma non un cambio di modo.



E se non tornassimo mai ad eseguire user mode??

E se non avessimo mai eseguito in user mode??