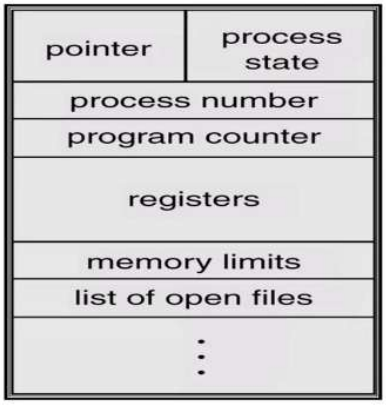
Processi e thread 2 Immagine di processo+ PCB

lunedì 13 ottobre 2025 18:21

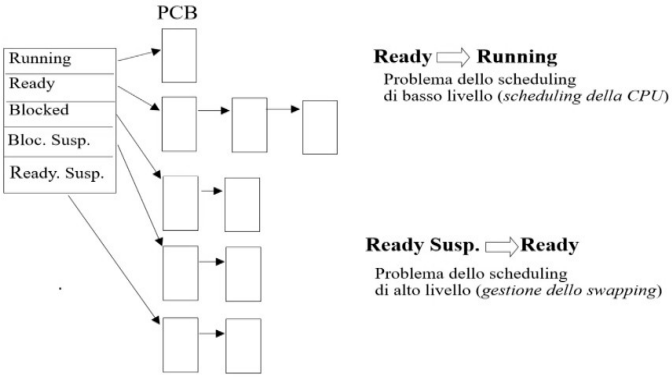
Vediamo ora altro concetto importante : **istanza di un processo** ovvero l'insieme di tutte le info che caratterizzano un processo : il programma in cui applicazione risulta istanza , i dati inclusi all'interno dell'address space , la stack area (all'interno dell'address space) , ulteriore stack area di sistema (dovuta alla system call -> quindi ulteriore stack area per determinare il flusso e valore/posizione delle variabili del software di sistema che stiamo eseguendo) , ed una collezione di attributi **Process Control BLock** (PCB) ovvero informazioni che il software di sistema utilizza per andare a controllare l'esecuzione del processo. Sono metadati . Tipicamente rappresentati in blocchi di memoria (principale e/o secondaria) . Nota : alcune di queste informazioni dell'**immagine** del processo sono sempre mantenute in memoria principale : tutte quelle informazioni che riguardano la parte sistema (PCB e stack area del sistema): sono residenti in RAM. Addentriamoci ora nel PCB , vedendo gli attributi basici :

1. Identificatori
    a. Del processo in oggetto e di processi relazionati (padre ed eventuali figli)
    b. Di solito è un codice numerico
2. Stato del processo
    a. Posizione corrente del processo : in quale stato si trova
3. Privilegi
    a. Possibilità di questa applicazione di accedere a servizi e privilegi
4. Registri(Contesto di esecuzione)
    a. Fotografia della CPU(contesto di esecuzione), ripristinando lo stato dopo
5. Informazioni di scheduling
    a. Qual' è l'importanza di questa applicazione rispetto alle altre
6. Informazioni di stato
    a. Relazioniamo applicazione rispetto agli eventi , registrando alcune informazioni all'interno del suo PCB

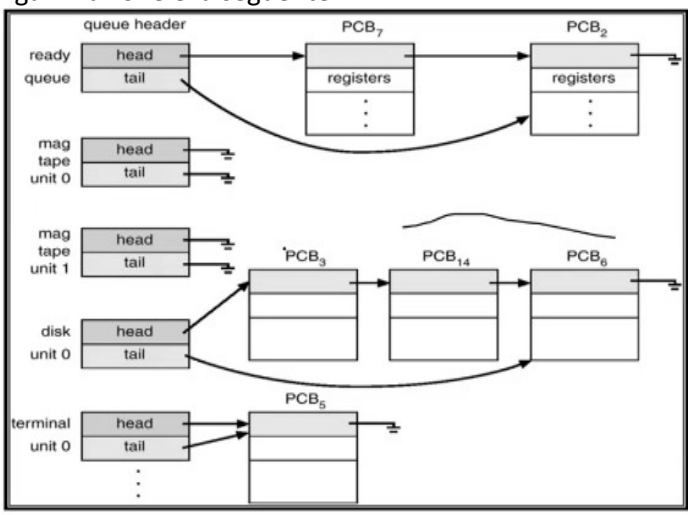
Vediamo ora come è fatto il Process Control Block in dettaglio :



Nota : per lista di file aperti si intende un riferimento ad altra tabella indicizzata tramite le info passate alle syscall, mentre per quelle della memoria rappresentano come applicazione può usare il suo address space . Per quanto riguarda invece il pointer , ci permette di puntare a qualcos'altro : liste di PCB. In dettaglio :

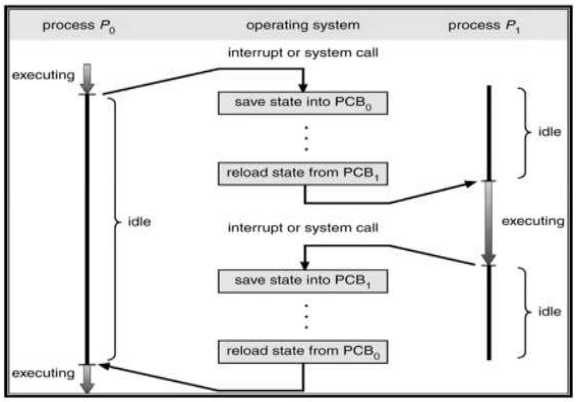


Una variante di questa organizzazione è la seguente :



Vediamo ora a vedere due concetti importanti :

1. **Cambio contesto**
   a. Cambio lo stato della CPU , magari cambiando anche processo attivo
   b. Svolto dal software del sistema, magari perché arriva interrupt
   c. Si ha salvataggio dello stato del processo corrente all'interno di un PCB, aggiornando il PCB con lo stato in cui lo portiamo, inserendolo nella lista/coda adeguata
   d. Quindi selezioniamo il prossimo processo da schedulare ,aggiornando il suo PCB
   e. Ripristino il contesto del processo schedulato
   f. Vediamo un esempio:
   i.



   g. Di solito è dovuto al time-sharing, oppure interruzione i/o, oppure per un fault di memoria (deattivo il processo corrente), oppure anche per attivazione di systemcall
2. **Cambio modo di esecuzione**
   a. Accediamo a privilegi di livello superiore (eseguo blocco di codice kernel): da modalità user a modalità kernel, magari in seguito a syscall
   b. Quindi si ha la possibilità di eseguire istruzioni che in modalità utente non sono possibili
   c. Le cause possono essere o l'attivazione di una funzione kernel oppure gestione di una routine di interruzione , chiaramente salvando/ripristinando la porzione di contesto
   d. Nota : **non si ha implicazione diretta tra entrambi**