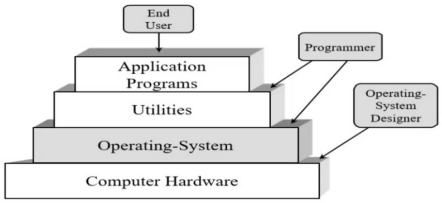
Introduzione1 Tipi di sistemi operativi

martedì 30 settembre 2025 11:38

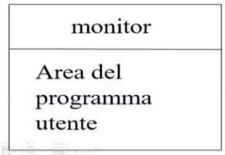
Andiamo a vedere i **sistemi operativi** ovvero dei componenti software .



Quindi con questo schema cerchiamo di andare a vedere la relazione tra il sistema operativo ed un programma che vogliamo farci girare. In dettaglio :

1. Application programs
    a. Software applicativo che usa il software del sistema operativo ed inoltre può usare anche le utility ed hardware
    b. Possibili chiamate a software esterno(software sistema operativo)
    c. Possibili chiamate ad utility , le quali a loro volta potrebbero chiamate il sistema operativo
2. Utilities
    a. Possono essere identificate come librerie (moduli già scritti)
3. Operating System
    a. Software del sistema operativo
    b. Implementa serie di servizi o task che possono essere triggerati (mandati in esecuzione)
    c. È caratterizzato da
        i. Semplicità
            1. Posso utilizzate moduli già scritti per invocare attività per lavorare su parti specifiche di hardware , oppure possiamo avere situazione mista : ovvero in certi punti del software utilizziamo software per lavorare su hardware ed il altri punti utilizziamo le utility
        ii. Efficienza
            1. Codifico attività in modo che i task (attività) siano ottimizzati
        iii. Flessibilità
            1. Portabilità dell'applicazione
    d. Quindi rappresenta una **facilitazione per l'utilizzo del sistema hardware e per lo sviluppo di applicazioni**
4. Computer Hardware
    a. Componenti fisici per fare eseguire i programmi come per esempio la CPU

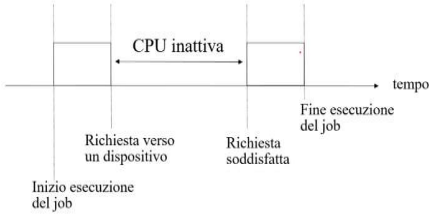
Vediamo ora un concetto molto importante : la **virtualizzazione** delle risorse : ovvero rappresentazione di un oggetto tramite software . Quindi utilizzando questo approccio si usano le risorse virtuali piuttosto che quelle reali , ed il legame tra questi due tipi di risorse è mantenuta dal sistema operativo in modo trasparente (mascherando in dettagli e la struttura). Nota : le risorse reali sono assegnate in **modo esclusivo** , limitando così il parallelismo nell'esecuzione delle applicazioni, il quale viene bypassato se si utilizzano quelle virtuali , come per esempio sistema multi utente oppure unico utente ma molteplici applicazioni. Vediamo ora un po' di storia dei sistemi operativi :

1. **Elaborazione seriale (anni 40-50)**
    a. Elaborazione seriale (job) da eseguire: un solo job alla volta
    b. Tutto a carico dell'utente
        i. Caricare il compilatore del programma all'interno dell'architettura hardware, fornendo il programma da compilare come programma da analizzare
        ii. La compilazione ci genera un programma compilato salvato da qualche parte
        iii. Il programma compilato può essere collegato ad altri oggetti
        iv. Collegamento tra i vari moduli e li carico in memoria del sistema hardware
    c. Nota : **non c'era il software** : si utilizzavano lettori di schede perforate . Quindi aumentava il tempo speso per le attività
    d. Migliorie : sviluppo di librerie di funzioni , linker , loader , debugger
2. **Sistemi operativi batch (anni 50-60)**
    a. 

    b. Dove nella zona superiore vi era mantenuto il software di un componente : quindi per monitor si intende l'istanza di questo sistema operativo batch : c'è del software che vede se ci sono dei job da mandare in esecuzione , i quali job vengono caricati nella parte sottostante . Oltre a fare ciò ha al suo interno i driver dei dispositivi ovvero moduli software che permettono di pilotare delle componenti hardware
    c. **Il monitor risiede in memoria**
    d. Nella parte bassa (area programma utente) viene caricato il software del programma incluse le subroutine
    e. Quindi in generale si ha che :
        i. Il programma è reso disponibile tramite un dispositivo di input
        ii. Il monitor effettua il caricamento del programma in memoria e lo avvia
        iii. Il programma restituisce il controllo al monitor in caso di terminazione e/o errore ed in caso di interazione con i dispositivi : ci fa capire le relazione tra il software del monitor e quello del sistema . Il monitor quindi riprende il controllo non quando vuole

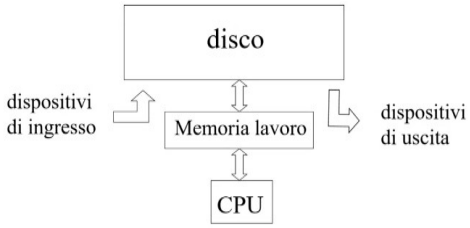
iv. Il monitor quindi è residente in memoria di lavoro
v. Il monitor esegue attività di interrelazione con specifici dispositivi prelevando software da caricare in memoria

f. Vediamo ora i limiti di questa architettura:

i. 

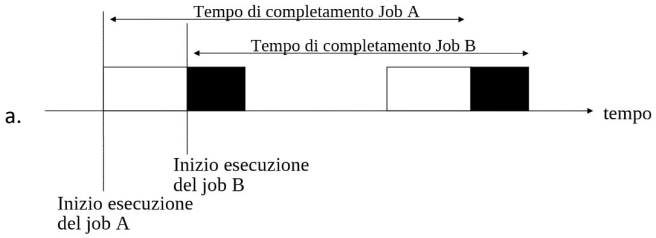
ii. Si nota che si ha un solo job in esecuzione alla volta il che potrebbe portare ad un sotto utilizzo della CPU dovuto alla lentezza dei dispositivi
iii. Nella zona CPU inattiva non si ha che la CPU non lavora, ma non la stiamo sfruttando per farle svolgere istruzioni macchina in modo corretto o fruttuoso : per esempio busy-waiting

g. Vediamo ora una serie di tecniche per bypassare questa lentezza :
i. **Spooling (Operazione di periferica online simultanea)**

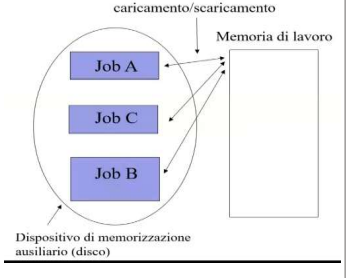
1. 

2. Quindi l'operazione on-line simultanea viene fatta rispetto all'esecuzione di un programma che sta utilizzando la CPU: il programma è già in RAM. Quindi l'operazione di periferica simultanea riguarda il disco di memorizzazione : memoria disco per mantenere info , quindi si può sia lavorare con dati in memoria che con dati nel disco.
3. Attenzione però : la velocità di scrittura/lettura di dati dal disco è determinata dalla velocità del dispositivo di ingresso
4. Quindi il disco viene inteso come **buffer tampone** ovvero una memoria tampone che velocizza i caricamenti verso la memoria , ma anche gli scaricamenti verso l'output
5. Quindi si ha una diminuzione della percentuale di attesa della CPU
6. Ed è possibile quindi avere contemporaneità di input ed output di job distinti

3. **Sistemi batch multi programmati (multitasking)**

a. 

b. Quindi permette la gestione simultanea di più job
c. È il monitor che cambia il flusso per portare l'esecuzione ad altro job, **ma non sappiamo quando farlo**
d. Le istruzioni del job devono essere prelevate(fetch) dalla RAM
e. Quindi la memoria di lavoro viene utilizzata per memorizzare i job secondo uno schema di :

Partizioni multiple (RAM) :ognuno di questi job può interagire con il monitor	
Partizione singola (Disco):appena finisce un job entra il prossimo (non è detto che vi sia ordine)	

f. Facciamo ora un paragone :

	job1	job2	job3	
Tipo	calcolo	calcolo+I/O	calcolo+I/O	
Durata	5 min	15 min	10 min	
Utilizzo disco	no	no	si	
Utilizzo terminale	no	si	no	
Utilizzo stampante	no	no	si	

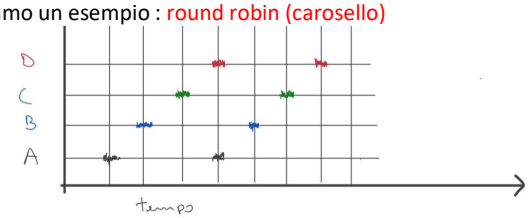
i.

	monoprogrammazione	multiprogrammazione
Utilizzo pocessore	17%	33%
Utilizzo del disco	33%	67%
Utilizzo stampante	33%	67%
Tempo totale	30 min	15 min
Throughput	6 job/ora	12 job/ora
Tempo di risposta medio	18 min	10 min

g. Un suo limite si ha quando si ha esecuzione di job con frequenti richieste di dispositivo : **quindi si ha l'impossibilità di avere applicazioni interattive** (vado a utilizzare software di sistema per interagire con i dispositivi) : magari dati due job A e B il primo non riesce a riprendere il controllo , quindi potrebbe verificarsi un sotto utilizzo di risorse

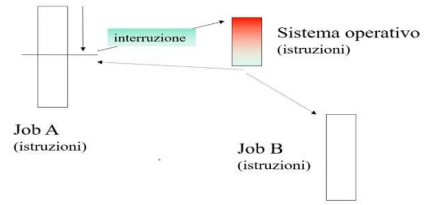
4. **Sistemi operativi time sharing (anni 60/70)**
   a. Questi sistemi condividono il tempo
   b. Il tempo di CPU viene assegnato ai vari job viene assegnato secondo un opportuno algoritmo di scheduling (pianificazione), il quale viene stabilito dal monitor
   c. Quindi c'è la possibilità di interruzione del job anche se non c'è una richiesta verso un dispositivo **(preemption)**
   d. Vediamo un esempio : round robin (carosello)

i.



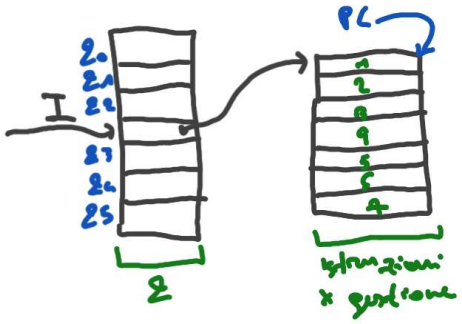
   ii. Dove i processi vengono eseguiti uno per volta e per un certo tempo di CPU : si garantisce cosi la consistenza dei dati
   e. Supporto al time sharing
      i. Quando si ha un job e facciamo il fetch delle istruzioni, potrebbe capitare che un qualcosa dica alla CPU di non fare il fetch delle istruzioni : le **interruzioni** ovvero il ritorno del controllo al monitor (base per costruire sistemi time sharing)

ii.

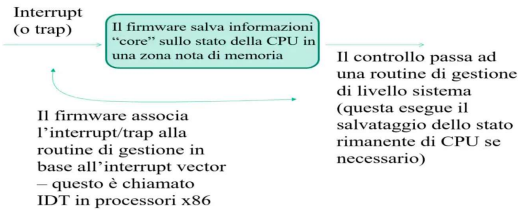


      iii. Se eseguo il job A e lo interrompo, posso avere due casi :
         1. Il sistema operativo comincia a fare il fetch delle istruzioni del job B (carosello -> round robin)
         2. Il sistema operativo continua a fare il fetch delle istruzioni del job A si ha che il job A ha maggiore priorità rispetto al job B
      iv. Come vengono gestite le interruzioni ?
         1. Si trasferisce il controllo ad un modulo software che si occupa della gestione: **routine di interrupt**
         2. E questo trasferimento avviene tramite un **interrupt vector** ovvero una struttura dati dove registriamo gli indirizzi di memoria di quali sono le routine per poter gestire le interruzioni
         3. Quindi l'architettura oltre ad implementare questo interrupt vector , deve anche fornire meccanismi di salvataggio dell'indirizzo della prossima istruzione del programma interrotto (PC->Program counter) e di tutti i valori dei registri della CPU

4.



         5. **Quindi ogni sistema operativo moderno è interrupt driven**
         6. Riassumendo quindi :

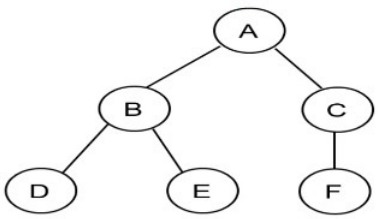


a.

   f. **L'applicazione attiva viene chiamata processo** : programma in esecuzione che opera sui dati applicativi ed è caratterizzato anche dal

contesto di esecuzione (informazioni necessarie al sistema operativo per schedularlo). Possibile interazione tra più processi o con sistema operativo (insieme di processi/programmi/moduli software).

g. Possibile gerarchia tra processi :

i. 

ii. **A è parent e B è child**

5. **Sistemi real time**
   a. In questi sistemi un programma esegue specifici task entro dei limiti temporali (deadlines) . Si classificano in :
   b. **Hard real time**
      a. Le deadlines devono essere rispettate
      b. Si utilizzano principalmente in processi di controllo industriali
      c. Dati ed istruzioni vengono tipicamente memorizzate in componenti ad accesso veloce o memorie a sola lettura (ROM)
      d. Chiaramente non è supportato dai sistemi operativi time sharing
   c. **Soft real time**
      a. Le deadlines dovrebbero essere rispettate il più possibile , ma se non avviene non si hanno gli stessi problemi dell'hard real time
      b. Di solito si utilizzano in applicazioni che richiedono funzionalità di sistema avanzate (multimediali)

Quindi in generale i servizi classici di un sistema operativo sono:

1. Gestione dei processi
2. Gestione memoria principale e secondaria
3. Gestione file
4. Gestione i/o
5. Protezione delle risorse