

Il Sistema Operativo

Corso di *Informatica Generale*

(Roberto BASILI)

Sistemi Operativi

Cos'è un sistema operativo ?

- **Un sistema operativo è:**
 - La astrazione logica del livello macchina hardware e microprogrammata
 - Un insieme di programmi che rende "utilizzabile" un sistema di elaborazione.
 - Il gestore delle risorse di un sistema di elaborazione
 - L'interfaccia utente di un qualsiasi elaboratore

Utenti di un SO

- Nella classe degli utenti di un sistema operativo troviamo:
 - Operatori dei sistemi di elaborazione
 - Programmatori di Sistema
 - Programmatori di Applicazioni
 - Personale Amministrativo
 - Utenti
 - *Programmi*
 - *Sistemi/Dispositivi Hardware*

Gestione Risorse

- Un Sistema Operativo gestisce in genere un sistema le cui risorse essenziali sono:
 - Uno o più Processori
 - Memoria
 - Dispositivi di Memorizzazione di Massa
 - Dispositivi di I/O
 - Dati

Gestione Risorse (2)

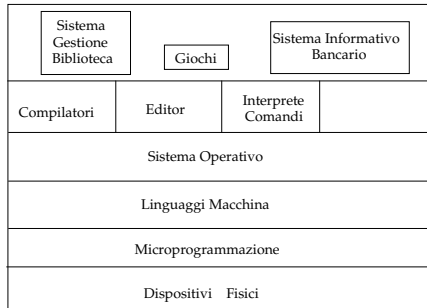
- Garantire l'utilizzo del sistema significa:
 - Fornire una astrazione adeguata dei singoli dispositivi agli utenti dei livelli macchina superiori
 - Controllare l'accesso condiviso a tutte le risorse
 - Garantire tolleranza agli errori

Gestione Risorse (3)

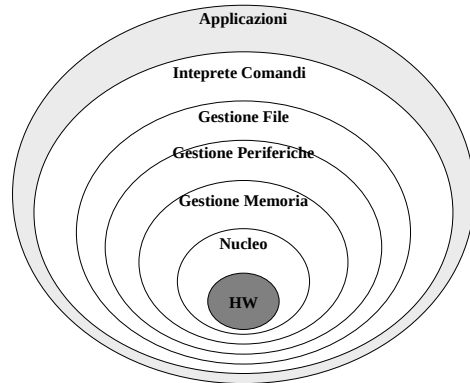
In quanto gestore delle risorse un S.O. svolge le seguenti funzioni:

- Fornisce una **interfaccia all'utente** (comandi/grafici/icone)
- Regola la **condivisione di risorse** tra gli utenti
- Effettua la **allocazione delle risorse** agli utenti che le richiedono
- Garantisce una **astrazione delle funzioni di I/O**
- **Gestisce Errori**: Rilevamento, Reazione, Eliminazione, Ripristino

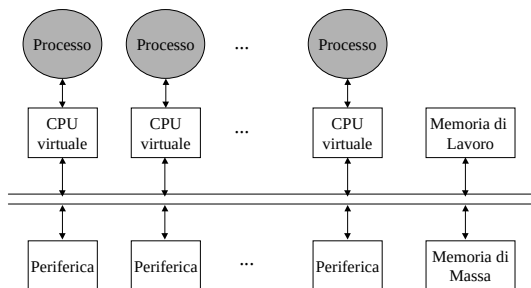
Sistema Operativo: macchina astratta



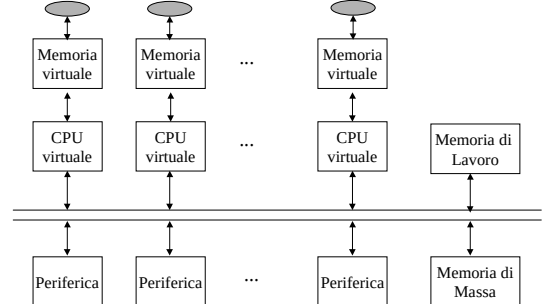
Modello A Livelli - Macchine Virtuali



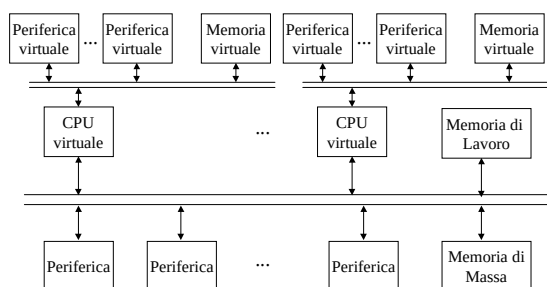
Macchine Virtuali - Nucleo



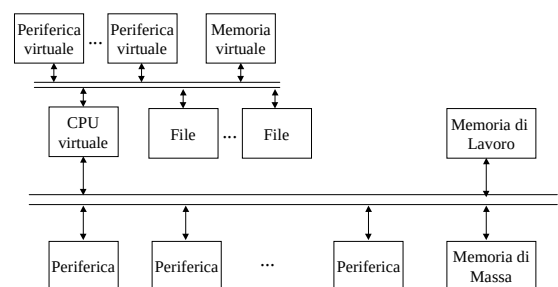
Macchine Virtuali - Gestore Memoria



Macchine Virtuali - Gestione Periferiche



La gestione della Memoria di massa



Interfaccia Utente

Ogni Sistema Operativo ammette una componente che viene generalmente denominata interprete dei comandi o *shell*. La *shell* garantisce una serie di primitive ad alto livello (i comandi) che supportano l'interazione tra l'utente e la macchina, permettendo cioè:

- l'attivazione dei comandi;
- l'accesso alla memoria di massa;
- l'accesso ai dispositivi di I/O;

Interfaccia Utente: Esempi

- Esempi tipici:

- **Command.COM** nel Ms-DOS

- `c:\miadir> type pippo.txt`

- `c:\miadir> dir /p`

- `c:\miadir> mkdir documenti`

- **Cshell** in Unix

- `/usr/local/user> cat pippo.txt`

- `/usr/local/user> ls -al | more`

- `/usr/local/user> md documenti`

Interfaccia Utente

- Le Interfacce grafiche: Windows

Multiprogrammazione (Parallelismo)

- La disponibilità di hardware sempre più potente e costoso, e la variabilità dei compiti dei programmi condussero negli anni '70 all'introduzione del concetto di *multiprogrammazione*.
- In un sistema multiprogrammato, più programmi possono essere eseguiti contemporaneamente anche se il processore è in grado di eseguire sequenzialmente le istruzioni di un solo programma per volta.

Multiprogrammazione (Parallelismo)

- La residenza in memoria di più programmi attivi, che si alternano nel controllo della CPU, è detta *multiprogrammazione*.
- La nozione di *processo* cattura adeguatamente il concetto di programma attivo.

Multiprogrammazione (Parallelismo)

- In un ambiente dotato di multiprogrammazione quindi è compito del sistema operativo di gestire l'insieme dei processi attivi in un certo istante:
 - decidere se il processo che sta usando la CPU può continuare, ed eventualmente sospenderlo
 - decidere il successivo processo a cui concedere l'uso della CPU
 - gestire una coda di processi attivi

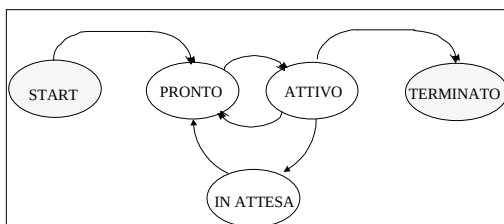
Processo

- Un processo e' un programma *in esecuzione*, descritto cioè da un suo stato:
 - istruzione successiva da eseguire
 - stato dei propri dati
 - dallo stato di eventuali processi ad esso correlati (ad es. *figli*)
- Tra i compiti principali di un SO c'e' quello di *attivare, interrompere temporaneamente o concludere programmi* cioè *creare, sospendere o terminare processi*.

Processo

- Un processo sequenziale puo' essere:
 - ATTIVO: mentre le istruzioni del suo programma vengono eseguite
 - IN ATTESA: mentre attende che alcuni eventi abbiano luogo (per esempio, che un altro processo rilasci la stampante)
 - PRONTO: quando attende di essere assegnato ad un processore, cioè attivato.

Processi



Stato del Processo

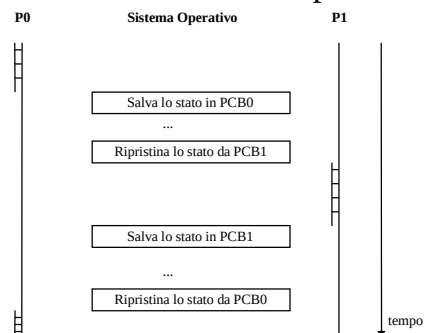
- Ogni processo e' rappresentato dal suo Blocco di Controllo dei Processi (**Process Control Block**) che contiene informazione riguardo a
 - Stato del processo
 - Program Counter
 - Registri della CPU
 - Informazioni per la schedulazione (priorita', puntatore al successivo, ...)
 - Informazioni per la gestione della memoria del processo
 - Informazioni per l'account (addebito)
 - Stato dei canali di I/O, dei file aperti, delle richieste

Stato di un processo

- Il PCB di un processo

Puntatore	Stato del Processo
	PID Processo
	Program Counter
	Registri
	Limiti di Memoria
	File Aperti
	...
	...

Avanzamento di un processo



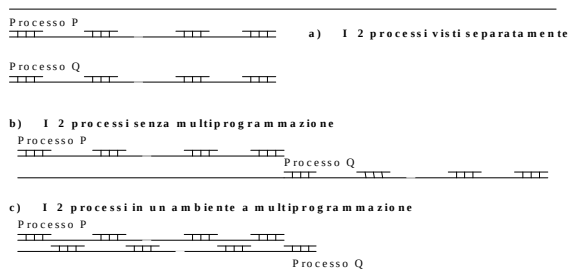
Comunicazione tra Processi

- Nei sistemi uniprocessore anche se molti processi sono attivi solo UNO può ottenere il controllo della CPU. E' necessario definire metodi per decidere:
 - quale processo
 - per quanto tempo
 merita o può ottenere il controllo della CPU.
- Tale compito è detto **SCHEDULING**.

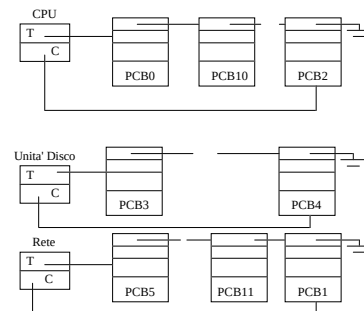
Scheduling

1. Esegui un processo P fino a quando non può evolvere oltre (ad es. richiede una risorsa non disponibile)
2. Allora togli al processo P il controllo della CPU
3. Scegli tra i processi PRONTI il nuovo processo Q a cui affidare il controllo della CPU
4. Torna in 1 con P uguale a Q.

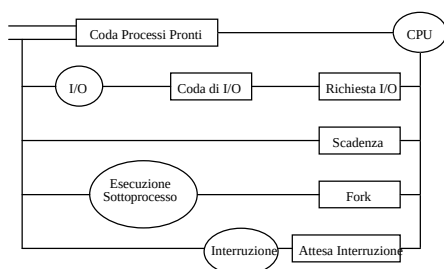
Scheduling



Scheduling



Scheduling



Scheduling

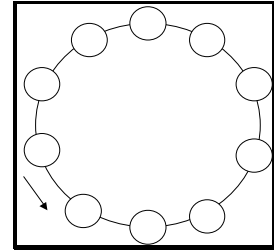
- Lo *scheduling* può aver luogo in una delle quattro situazioni seguenti:
 - Il passaggio di un processo da ATTIVO in PRONTO
 - Il passaggio di un processo dallo stato di IN ATTESA ad uno stato di PRONTO (ad es. è avvenuto un *interrupt*)
 - Il passaggio di un processo dallo stato di ATTIVO ad uno stato di IN ATTESA (ad es. è avvenuto un *interrupt*)
 - Un processo termina

Algoritmi di Scheduling

- Le tecniche algoritmiche per la gestione dello *scheduling* si possono raggruppare in alcune classi piu' importanti:
 - algoritmi FIFO
 - algoritmi Shortest-Job-First
 - algoritmi basati su priorit 
 - algoritmi *Round-Robin*

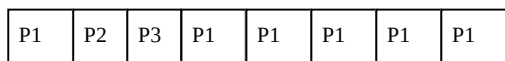
Algoritmi di Scheduling

- Round-Robin
 - coda circolare
 - quanto di tempo, τ
 - ad ogni processo non piu' di un quanto τ



Round-Robin

Processo	Tempo di CPU
P1	24
P2	3
P3	3



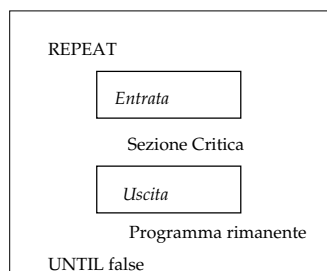
4 ms 3 ms Tempo medio d'attesa = 5.6 ms

Scheduling: problemi tipici

Corse Critiche

- Sono *corse critiche* quelle situazioni in cui il risultato della esecuzione delle istruzioni di due programmi che si contendono l'accesso ad una risorsa dipende dall'ordine di esecuzione delle situazioni dei due programmi.
- La sequenza delle istruzioni che contenendo accesso a risorse condivise possono generare corse critiche e' detto *sezione critica*.
- Tipiche sezioni critiche sono relative a problemi di *mutua esclusione* tra i processi.

Corse Critiche



Corse Critiche

- Soluzioni
 - Con *mutua esclusione* (meno efficienti)
 - Variabili di Lock
 - Senza *mutua esclusione*
 - Semafori
 - Monitor
 - Scambio di messaggi

Scheduling: problemi tipici

Starvation

- *Starvation* si riferisce alla situazione in cui un processo che non può avanzare, poiché in attesa di una risorsa, attende indefinitamente senza mai terminare la propria computazione.
- Come sottolineato gli algoritmi di scheduling consentono di evitare tali situazioni (es. Round-Robin)

Interazione tra Processi

- Modalità di un processo
 - modo utente
 - modo supervisore
- Processi foreground
- Processi background

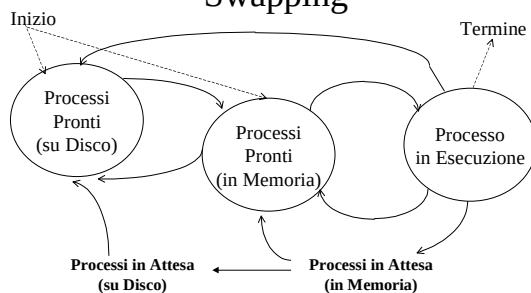
Gestore Memoria

- Il sistema operativo garantisce ai programmi utente e supervisor
 - il caricamento di essi in memoria centrale
 - la ottimizzazione dell'uso della memoria, mantenendo in memoria solo la componente dei dati e del programma utile in una certa fase di elaborazione
- Il gestore della memoria garantisce quindi uno spazio di indirizzamento virtuale ai programmi più grande dello spazio fisico di memoria disponibile

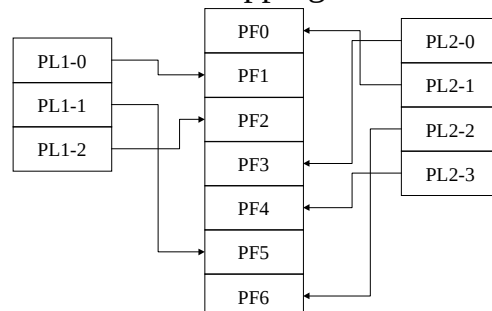
Gestione della Memoria: swapping

- Lo spostamento di processi dalla memoria al disco e viceversa è necessaria in sistemi *multiprogrammati* e *timesharing* poiché la memoria è insufficiente per contenere il numero (impredicibile) dei processi utente. Questa attività è detta SWAPPING (scambio).
- Due tecniche che consentono un efficace gestione della memoria in presenza di *swapping* sono la gestione virtuale della memoria (o *memoria virtuale*) e la *segmentazione*.

Swapping



Swapping



Gestione della Memoria

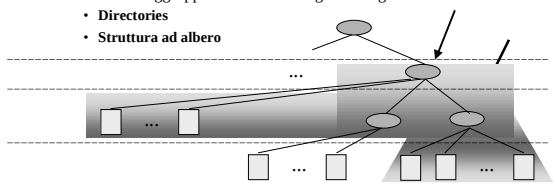
- **Segmentazione**
 - I vari processi non hanno una dimensione fisica qualsiasi
 - Per evitare un uso ridondante della memoria essi vengono suddivisi in differenti componenti di dimensione fissata e di uso potenzialmente condiviso tra processi diversi (per es. librerie grafiche)
 - I componenti vengono detti segmenti:
 - codice
 - dati
 - pila
 - I segmenti vengono gestiti in modo paginato

Gestione Memoria di Massa

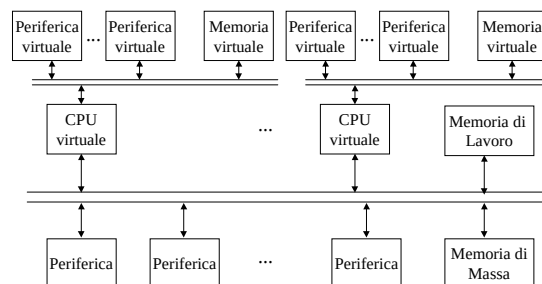
- Le operazioni tipiche effettuate sulla memoria di massa sono:
 - Recupero di Informazione precedentemente memorizzata
 - Eliminazione della Informazione
 - Modifica/Aggiornamento dei dati
 - Copia o trasferimento (es. da Hard-disk a floppy-disk)
- Il gestore della memoria di massa che garantisce una visione logica delle suddette operazioni e' anche detto **File system**

Il File System

- **Localizzazione delle Informazioni**
 - Ogni informazione memorizzata e' organizzata in un File (archivio)
 - Nome
 - Estensione
 - I file sono raggruppati in insiemi e organizzati gerarchicamente
 - Directories
 - Struttura ad albero



Gestione Periferiche di I/O



Gestione Periferiche I/O

- I/O indipendente dall'hardware sottostante
 - driver fisici
 - specifici al tipo di dispositivo (HD vs. CD-ROM)
 - specifici al tipo di costruttore di un certo HW
 - driver logici
 - realizzano primitive di accesso (I/O) indipendenti dal tipo di dispositivo (files su CD-ROM e su HD)
 - sono da installare/configurare oppure **Plug&Play**
 - garantiscono la affidabilita' del processo (generalmente asincrono)

Gestione Periferiche I/O

- Tipici dispositivi sono
 - video
 - stampante/i
 - tastiera
 - CD-ROM
 - Porte esterne
 - dispositivi dedicati (es. tastiere MIDI)
 - rete telefonica
 - audio

Gestione Periferiche di I/O

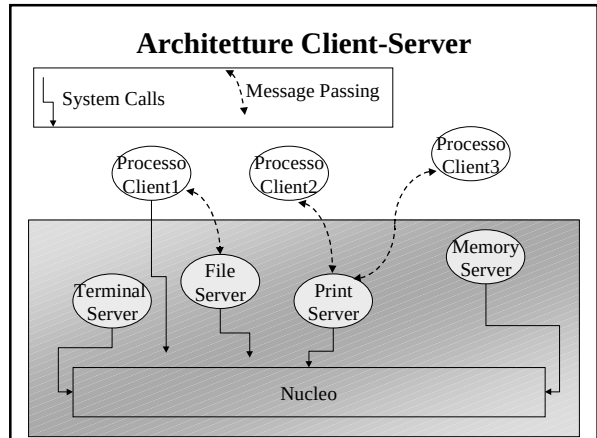
- Un esempio: la stampa
 - Una stampante non è condivisibile
 - La stampa viene *virtualizzata* in stampanti dedicate
 - I conflitti vengono gestiti mediante lo **spooling**
 - un richiesta di accesso (per la stampa) viene accodata
 - le stampe vengono effettuate mediante una politica FIFO fino allo svuotamento della coda (detta coda di stampa)
 - la attesa attiva in memoria permane finché una nuova richiesta viene accodata

Modelli Organizzativi

- Modello Monolitico
 - modo utente
 - modo supervisore
- Modelli a strati
 - Nucleo come unico strato dipendente dall'HW
 - processi utente responsabili delle risorse
- Modelli basati su macchine astratte

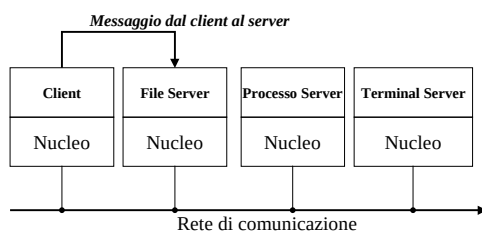
Modelli Organizzativi

- Modello Client-Server
 - Finalità: un nucleo semplice
 - Modalità
 - processi di sistema visti come processi utente (*processi server*)
 - processi utente come clienti dei processi server (*processi client*)
 - Il nucleo garantisce le comunicazioni tra i vari processi



Architetture Client-Server

- Modello Client-Server in sistemi distribuiti



Windows NT

- Integra Client-Server e stratificazione
- Workstation e Server
- Modalità kernel
 - tutta la memoria e istruzioni sono disponibili
 - vengono garantite le comunicazioni tra processi
- Modalità utente
 - per le applicazioni (es. Word, compilatori, ...)

