**Introduzione**

**Sistemi Operativi - definizione di S/O**

Il sistema operativo ha due funzionalità: estende la macchina e gestisce le risorse.

E’ un insieme di utilità progettate per…

1. Offrire all’utente un’astrazione più semplice e potente della macchina assembler

• Concetto di “macchina virtuale” (macchina estesa) che sia più facile da programmare dell’hardware sottostante.

– ambiente virtuale dove eseguire applicazioni

– originariamente per sistemi multi-utente

• Più semplice da usare (es., senza bisogno di conoscenze di microprogrammazione) senza che il programmatore debba preoccuparsi le caratteristiche a basso livello dell’hardware.

• Più potente (es., usando la memoria secondaria per realizzare una più ampia memoria principale virtuale)

2. Gestire in maniera ottimale le risorse fisiche e logiche dell’elaboratore

• Gestione delle risorse risorse rispetto al tempo e allo spazio:

* rispetto al tempo: programmi o utenti diversi fanno a turno per usare una risorsa (CPU)
* rispetto allo spazio: invece di alternarsi ad ognuno viene assegnata parte della risorsa (memoria, disco rigido, ecc)

Ottimalità e la minimizzazione dei tempi di attesa e la massimizzazione dei lavori svolti per unità di tempo

**Processi – definizione di processo**

Un processo è un programma in esecuzione e corrisponde a:

1. L’insieme ordinato di stati assunti dal programma nel corso dell’esecuzione (sulla sua macchina virtuale) - Processo come “**automa a stati**”
2. L’insieme ordinato delle azioni effettuate dal programma nel corso dell’esecuzione (sulla sua macchina virtuale) - Processo come “**attore**” (operatore di azioni)

**Realizzazione di processo**

Spazio di indirizzamento logico:

– La memoria della macchina virtuale che il processo può leggere e scrivere – Memoria virtuale organizzata a pagine e/o segmenti

• Programma eseguibile

• Dati del programma – Organizzazione dell’informazione in forma di file

• Aree di lavoro

**Caratteristiche di processi**

In un sistema coesistono processi utente e di S/O. Possono cooperare tra loro ma hanno privilegi diversi, i processi avanzano concorrentemente

Il S/O assegna loro le risorse necessarie secondo diverse politiche di ordinamento

• A divisione di tempo

• A livello di priorità (urgenza)

I processi possono dover comunicare e sincronizzarsi tra loro. Il S/O deve fornire i meccanismi e i servizi necessari .

Un processo può creare processi “figli”

– Esempio: un processo interprete di comandi (shell) lancia un processo figlio per eseguire un comando di utente.

I processi vengono

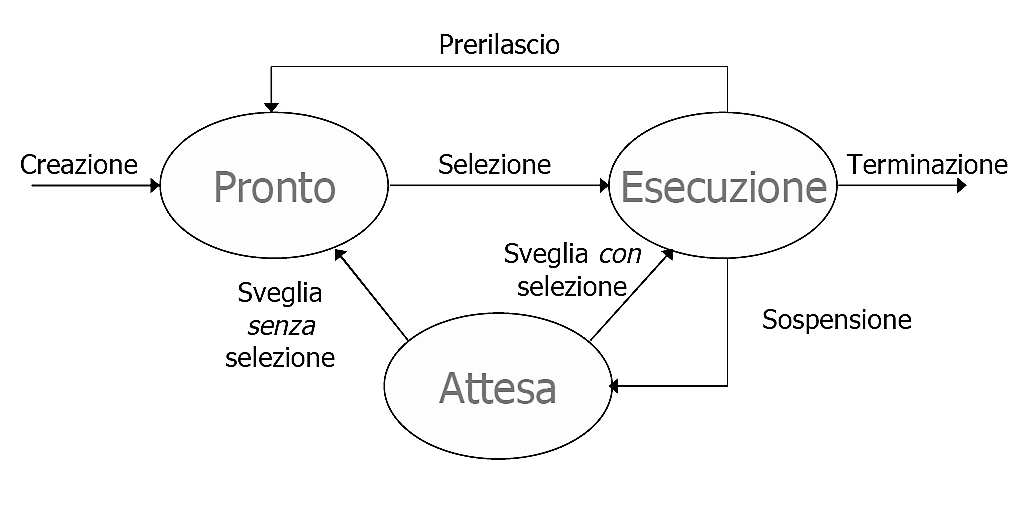
– **Creati** per eseguire un lavoro

– **Sospesi** per consentire l’esecuzione di altri processi

– **Terminati** al compimento del lavoro assegnato

• Un processo figlio che sopravvive alla terminazione del processo padre è detto “orfano” e può essere molto dannoso

**Stati di avanzamento di processo**



**Gestore dei processi**

Costituisce il cuore o nucleo del S/O (**kernel**) – Gestisce ed assicura l’avanzamento dei processi

• Stato di avanzamento

– In esecuzione, pronto per l’esecuzione, sospeso in attesa di un evento (una comunicazione, la disponibilità di una risorsa, …)

• La scelta del processo da eseguire ad un dato istante si chiama ordinamento (scheduling)

• Il gestore decide il cambio di stato dei processi

– Per divisione di tempo

– Per trattamento di eventi (es., risorsa libera / occupata)

Compiti del nucleo di S/O:

– Gestire l’avanzamento dei processi

• Registrando ogni transizione nel loro stato di attivazione

– Gestire le interruzioni esterne (all’esecuzione corrente) causate da

• Eventi di I/O

• Situazioni anomale rilevate da altri processi o componenti del S/O

– Consentire ai processi di accedere a risorse di sistema e di attendere eventi

La politica di ordinamento deve essere equa (fair → fairness)

– Processi pronti per eseguire devono avere l'opportunità di farlo

– Processi in attesa di risorse devono avere l'opportunità di accederle

I meccanismi e servizi di comunicazione e sincronizzazione devono essere efficaci

il destinatario in un tempo breve e in modo sicuro

**Risorse - definizione di risorsa**

Risorsa è qualsiasi elemento fisico (hardware) o logico (realizzato a software) necessario alla creazione, esecuzione e avanzamento di processi.

Le risorse possono essere:

– Durevoli (es., CPU)

– Consumabili (es., memoria fisica)

– Ad accesso divisibile o indivisibile

• Divisibile se tollera alternanza con accessi di altri processi

• Indivisibili se non tollera alternanza durante l’uso

– Ad accesso individuale o molteplice

• Molteplicità fisica o logica (virtualizzata)

**Risorsa CPU**

Risorsa indispensabile per l’avanzamento di tutti i processi.

A livello fisico (hardware) corrisponde alla CPU.

A livello logico (sotto gestione software) può essere vista come una macchina virtuale.

Offerta dal S/O alle sue applicazioni.

**Risorsa memoria**

Scrittura: risorsa ad accesso individuale

Lettura: risorsa ad accesso multiplo

La gestione software la virtualizza (usandola insieme alla memoria secondaria) attribuendone l’accesso ai vari processi secondo particolari politiche.

Se virtualizzata, diventa riutilizzabile e prerilasciabile

– Altrimenti consumabile e indivisibile

– Gestione velocizzata con l’utilizzo di supporto hardware

**Risorsa I/O**

Risorse generalmente riutilizzabili, non prerilasciabili, ad accesso individuale.

La gestione software ne facilita l’impiego nascondendone le caratteristiche hardware e uniformandone il trattamento.

L’accesso fisico ha bisogno di utilizzare programmi proprietari e specifici – BIOS

**Caricamento del S/O**

Il S/O può risiedere:

• Permanentemente in ROM

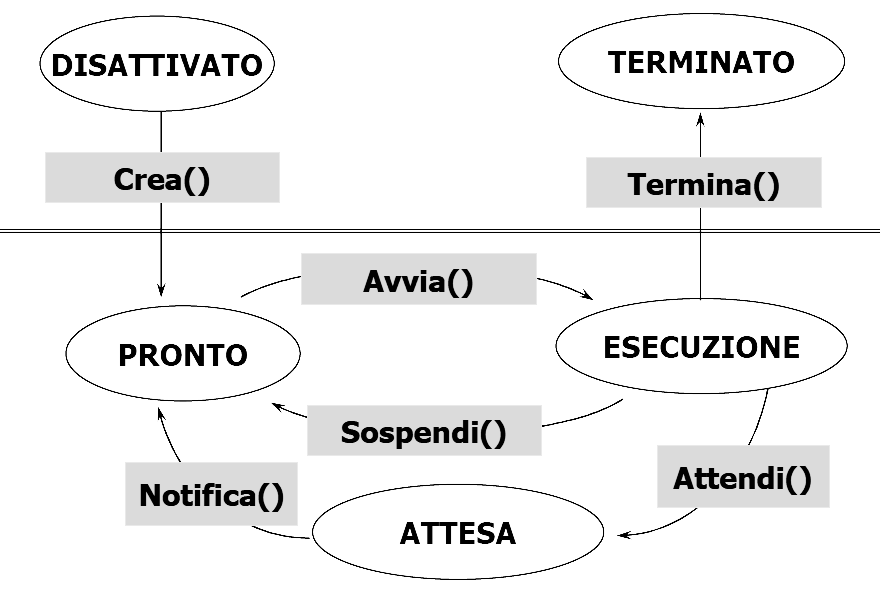
– Soluzione tipica di sistemi di controllo industriale e di sistemi dedicati

• In memoria secondaria per essere caricato (tutto o in parte) in RAM all’attivazione di sistema (bootstrap)

– Adatto a sistemi di elevata complessità oppure predisposti al controllo (alternativo) da parte di più istanze di S/O

– In ROM risiede solo il caricatore di sistema (bootstrap loader)

**Stati di avanzamento di processo**



**Stati**

* **DISATTIVATO**:

Il programma è in memoria secondaria. Unsupervisore lo carica in memoria mediante unachiamata di sistema che crea una struttura di

controllo di processo (Process Control Block, PCB)

* **PRONTO**:

Il processo, pronto per l’esecuzione, rimane in attesadel suo turno

* **ESECUZIONE**:

Il processore è stato attribuito al processoselezionato, la cui esecuzione avanza

* **ATTESA**:

Il processo è sospeso in attesa di una risorsa attualmente non disponibile o di un evento non ancora verificatosi

* **TERMINATO**:

Il processo ha concluso regolarmente le sue operazioni e si predispone ad abbandonare la sua macchina virtuale

**Transizioni**

* **Crea()**

Assegna una macchina virtuale a un nuovo processo, aggiornando la lista dei processi pronti (ready list)

* **Avvia()**

Manda in esecuzione il primo processo della lista dei pronti

* **Sospendi()**

Il processo in esecuzione ha esaurito il suo quanto di tempo e torna in fondo alla lista dei pronti

* **Attendi()**

Il processo richiede l’uso di una risorsa o l’arrivo di un evento e viene sospeso se la risorsa è occupata o se l’evento non si è ancora verificato

* **Notifica()**

La risorsa richiesta dal processo bloccato è di nuovo libera o l’evento atteso si è verificato. Il processo ritorna nella lista dei pronti

* **Termina()**

Il processo in esecuzione termina il suo lavoro e rilascia la macchina virtuale

**Strutture di rappresentazione**

Modello di processo realizzato tramite struttura a tabella (Process Table) – Array di strutture.

Ogni processo è rappresentato da un descrittore (Process Control Block) contenente :

– Identificatore del processo

– Contesto di esecuzione (stato interno) del processo

– Stato di avanzamento del processo

– Priorità (iniziale ed attuale)

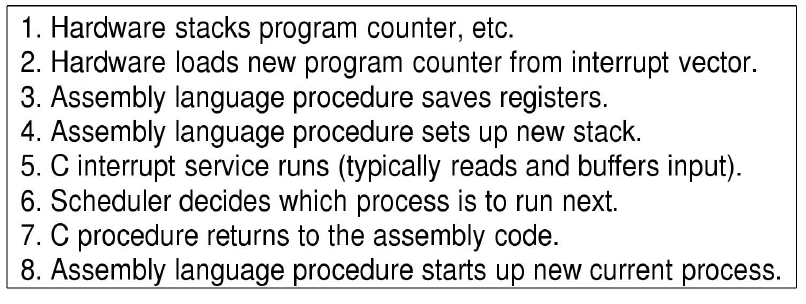
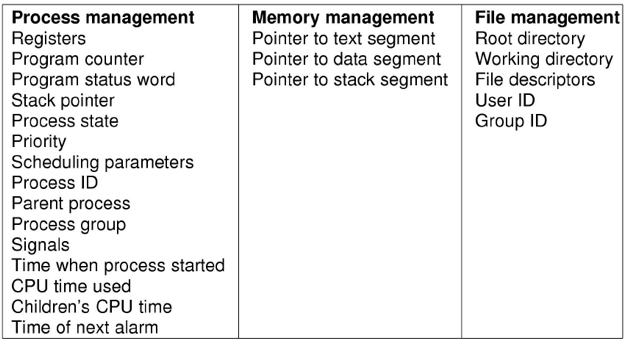
– Diritti di accesso alle risorse e privilegi

– Puntatore al PCB del processo padre e degli eventuali processifigli

– Puntatore alla lista delle risorse assegnate alla macchina virtuale del processo

– … Vedi Fig. 2.4 nel libro

**Campi di una Process Table Passi principali seguenti interrupt**



**Ordinamento di processi**

Diversi metodi utili per determinare quando porre un processo in stato di esecuzione in sostituzione di un altro (switch)

– **Scambio cooperativo** (cooperative o non pre-emptive switching)

• Il processo in esecuzione decide quando passare il controllo al processo successivo – Windows 3.1

– **Scambio a prerilascio**: il processo in esecuzione viene rimpiazzata da:

• Un processo pronto a priorità maggiore (priority-based pre-emptive switching) → Sistemi detti “a tempo reale”

• All’esaurimento del suo quanto di tempo (time-sharing pre-emptive switching) → Unix, Windows NT (misti)

**Dispatcher**

Il componente che avvia processi all’esecuzione (ma non il selettore, scheduler!) viene detto dispatcher

– Deve essere molto efficiente perché gestisce ogni scambio

– Deve salvare il contesto del processo in uscita, installare quello del processo in entrata (context switch) e affidargli il controllo della CPU

L’efficienza del dispatcher si misura in

– Percentuale di utilizzo della CPU

– Numero di processi avviati all’esecuzione per unità di tempo

– Durata di permanenza di un processo in stato di pronto

I processi in stato di pronto sono accodati in una struttura detta lista dei pronti (**ready list**).

La più semplice gestione della lista è con tecnica a coda (**First-Come-First-Served**, FCFS).

– Il primo processo ad entrare in coda sarà anche il primo avviato all’esecuzione

– Facile da realizzare e da gestire

– La garanzia di esecuzione di altri processi (fairness) dipende dalla politica di scambio - Lo scambio cooperativo non offre garanzie

Le attività di un processo tipicamente comprendono sequenze di azioni eseguibili dalla CPU intervallate da sequenze di azioni di I/O.

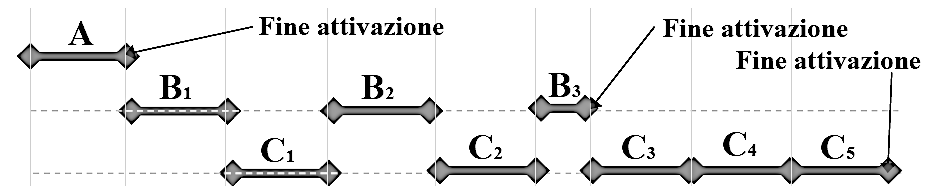
I processi si possono dunque classificare in

– CPU-bound: Comprendenti attività sulla CPU e di durata molto lunga

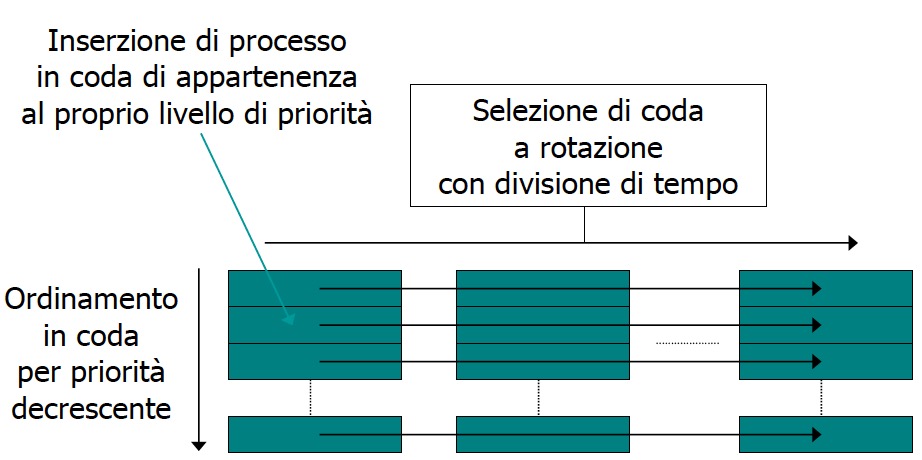
– I/O-bound: Comprendenti attività di breve durata sulla CPU, intervallateda attività di I/O molto lunghe

La tecnica FCFS penalizza i processi della classe I/O-bound

Imponendo la suddivisione di tempo (**time-sharing)** sulla politica FCFS si deriva una tecnica di rotazione detta **round-robin**.

Vediamone l’applicazione su tre processi A, B e C con tempi di esecuzione 2, 5 e 10 ms e quanto di tempo 2 ms

**Politica a rotazione con priorità**



**Dispatcher - 2**

A ogni singolo processo possiamo attribuire una priorità individuale che denota il suo livello di privilegio nel sistema.

Processi diversi possono poi essere categorizzati per attributi (p.es., CPU-bound, I/O-bound).

Possiamo allora istituire una coda per ciascuna categoria di processo e ordinarla a priorità.

Stabiliamo poi una politica di ordinamento tra code (p.es.: round-robin).

Otteniamo una politica di ordinamento a livelli

– Rotazione tra code e con priorità entro ciascuna coda

Possiamo anche facilmente (e utilmente) definire una politica duale alla precedente

– Istituiamo una coda per ogni livello di priorità attribuita ai processi

– Selezioniamo la coda a priorità più elevata

– Applichiamo la politica a rotazione (round-robin) sul processo selezionato

– Otteniamo la politica a priorità con rotazione

• Selezione prioritaria tra code e a rotazione equa entro ciascuna coda

**Politica a priorità con rotazione**

