

Gestione dei processi

- I) **Descrizione e controllo dei processi**
- II) **Schedulazione dei processi**
- III) **Concorrenza**

I) Descrizione e controllo dei processi

Processi

□ Definizione di Processo

❖ Programma in esecuzione con tutte le informazioni riguardanti il suo "stato"

- Programma eseguibile e dati associati (Codice e Dati)
- Contesto di esecuzione del programma
 - Tutte le informazioni necessarie affinché il S.O. possa gestirlo (Program Counter, Registri, Stack)
- Risorse necessarie ad un processo
 - CPU, Memoria, File e dispositivi di I/O

□ Tipi di Processi

❖ Processi I/O bound

- Eseguono soprattutto operazioni di I/O, richiedono poco tempo di CPU
 - Programmi gestionali

❖ Processi CPU bound

- Eseguono molte operazioni di calcolo, richiedono molto tempo di CPU
 - Programmi di tipo scientifico o ingegneristico

Funzione fondamentale del S.O. è la **gestione dei processi**, cioè allocare risorse ai processi, permettere loro di condividere e scambiare informazioni, proteggere le risorse di ciascun processo dagli altri, permettere la sincronizzazione

Processi

□ Istanze di un processo

- ❖ Più processi possono essere associati ad uno stesso programma
- ❖ Ciascun processo rappresenta l'esecuzione dello stesso codice con dati di ingresso diversi
 - Più utenti possono far eseguire diverse istanze dello stesso programma di posta elettronica o un utente può invocare più istanze dello stesso elaboratore di testo
 - Ciascuna di queste istanze è un diverso processo e, nonostante il codice del programma sia lo stesso, le sezioni dei dati sono diverse
- ❖ Più **istanze** dello stesso programma generano **processi diversi**

Nella multiprogrammazione su un singolo processore, l'esecuzione di processi avviene in **modo interlacciato nel tempo**, mentre nei sistemi multiprocessori può essere sia **interlacciata che simultanea**

Processi

□ Stato di un processo

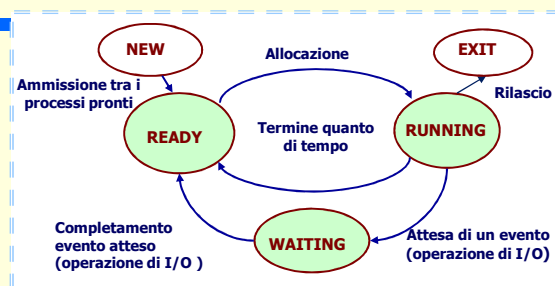
❖ Lo stato di un processo è caratterizzato da

- **numero di identificazione** del processo
- **stato di priorità** e tempo trascorso dall'ultimo stato
- **contesto di memoria, programma e dati** che devono risiedere in memoria quando il processo associato è pronto o in esecuzione
- **contesto del processore**, ossia il contenuto dei registri della CPU
 - Program Counter, Stack Pointer, e altri registri contenenti informazioni relative alla esecuzione del programma

□ Stati che caratterizzano il comportamento di un processo

1. **NEW** - **Processo appena creato** ma non ancora ammesso dal S.O. nell'insieme dei processi eseguibili
2. **READY** - **Processo pronto per l'esecuzione** ma in attesa della CPU (allocata ad un altro processo)
3. **WAITING** (Blocked) – **Processo bloccato** perché in attesa di un evento (interruzione da parte di un dispositivo, messaggio o sincronizzazione con un altro processo) o il liberarsi di una risorsa
4. **RUNNING** - **Processo in esecuzione**, ossia gli è assegnata la CPU
5. **EXIT** - **Processo** rilasciato dall'insieme dei processi eseguibili in quanto **terminato o fallito**

Processi



□ Transizione fra gli stati

- ❖ Il S.O. maschera al processo la evoluzione nel tempo della sua esecuzione eseguendo due operazioni
 1. **Salvataggio del contesto** del processo ogni volta che viene interrotto
 2. **Ripristino del contesto** quando viene ad esso riassegnata la CPU
- ❖ Il S.O. per gestire i processi ha necessità di informazioni riguardo il loro stato corrente: costruisce tabelle di informazioni relative a ciascun processo
- ❖ La **struttura dati** che contiene le informazioni utili per la descrizione dello stato di un processo è detta **descrittore del processo**

Processi

□ **Descrittore di processo**

❖ Ogni processo è rappresentato dal S.O. mediante un descrittore di processo (**Process Descriptor-PD**), detto anche blocco di controllo di un processo (**Process Control Block-PCB**)

❖ Il descrittore contiene informazioni riguardanti

- nome o identificatore del processo (**process index**)
- stato del processo
- contesto del processo (contatore di programma, registri di CPU)
- priorità del processo
- informazioni sulla gestione della memoria
 - tabelle delle pagine o dei segmenti
- informazioni sullo stato di I/O
 - lista dei dispositivi I/O assegnati al processo, elenco dei file aperti

❖ **Lista dei processi**

- **Tabella** che contiene i descrittori di tutti i processi

Puntatore	Stato del processo
Identificatore del processo	
Contatore di programma	
Registri	
Limiti di memoria	
Elenco file aperti	
.	

Processi

□ **Cambio di contesto**

❖ L'utilizzo della CPU viene commutato da un processo ad un altro mediante

- salvataggio del contesto del processo in esecuzione nel suo descrittore (**salvataggio stato**)
- inserimento del descrittore nella coda dei processi bloccati o pronti
- selezione di un altro processo dalla coda dei processi pronti e caricamento del suo nome nel registro processo in esecuzione (**short term scheduling**)
- Caricamento del contesto del nuovo processo nei registri del processore (**ripristino stato**)

❖ Il cambio di contesto può richiedere onerosi trasferimenti da/verso la memoria secondaria, per allocare e deallocare gli spazi di indirizzi dei processi (gestione della memoria)

Operazioni sui processi

- **Attività del S.O. per la gestione dei processi**
 - ❖ **Creazione e terminazione** dei processi
 - ❖ **Sospensione e riattivazione** dei processi
 - ❖ **Sincronizzazione** dei processi
 - ❖ **Comunicazione** fra processi

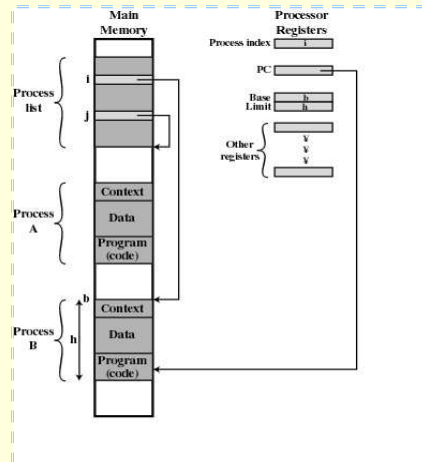
- Il S.O. (e in particolare il suo nucleo) simula il parallelismo fra processi nel caso in cui sia presente un solo processore
- In sistemi con più processori, il S.O. assegna ogni processo ad un processore diverso

Operazioni sui processi

- **Creazione di un processo**
 - ❖ I processi in un sistema si possono eseguire in modo concorrente e si devono creare e cancellare dinamicamente
 - Per ogni processo creato il S.O. inizializza il process control block e lo aggiorna dinamicamente
 - ❖ Meccanismi per la gestione dei processi
 - Durante la sua esecuzione, un processo può creare nuovi processi tramite una apposita chiamata di sistema: il processo creatore si chiama **genitore** e quello creato **figlio**
 - Ciascuno di questi processi può creare a sua volta altri processi, formando una gerarchia dei processi, **albero dei processi**
 - Il processo genitore può continuare l' esecuzione in modo concorrente con i processi figli o attendere che alcuni o tutti i suoi processi figli terminino
 - Il sistema operativo mantiene le informazioni relative alle relazioni parentali nel descrittore

Operazioni sui processi

□ Creazione dei processi



A.M.Fanelli

Architettura dei Sistemi a.a. 2008-09

11

Operazioni sui processi

□ Terminazione di un processo

- ❖ Un processo termina quando finisce la esecuzione della sua ultima istruzione
 - Il processo chiede al sistema operativo di essere cancellato usando la chiamata di sistema **exit**
 - Tutte le risorse del processo, incluse la memoria fisica e virtuale, i file aperti e le aree di memoria per l'I/O sono liberate dal sistema operativo
- ❖ Un processo può causare la terminazione di un altro processo per mezzo di una opportuna chiamata di sistema, ad esempio **abort**

□ Sospensione e riattivazione di un processo

- ❖ La sospensione e riattivazione dei processi è gestita dal S.O. mediante le **tracce** dei processi
 - L'obiettivo della multiprogrammazione è avere sempre più processi in esecuzione al fine di massimizzare l'utilizzo della CPU
 - In sistemi con singola CPU si può avere in esecuzione un solo processo alla volta: si deve commutare l'uso della CPU tra i vari processi

A.M.Fanelli

Architettura dei Sistemi a.a. 2008-09

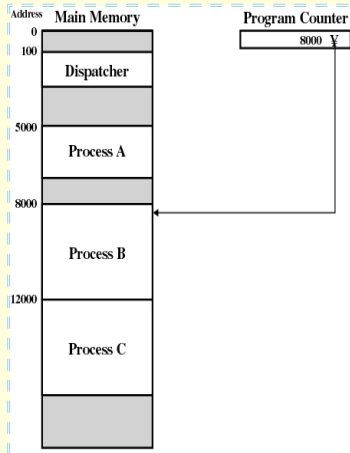
12

Operazioni sui processi

Allocatore (Dispatcher)

❖ Programma, residente in memoria, che trasferisce il processore da un processo all'altro

- Tre processi in memoria
 - processo A
 - processo B
 - processo C
- Processo B attivo



Operazioni sui processi

Traccia di un processo

❖ Il comportamento di un singolo processo è caratterizzato elencando la relativa **sequenza di istruzioni eseguite**. Tale elenco è detto **traccia** del processo

5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

(a) Trace of Process A

(b) Trace of Process B

(c) Trace of Process C

5000 = Starting address of program of Process A
 8000 = Starting address of program of Process B
 12000 = Starting address of program of Process C

Operazioni sui processi

Tracce Interallacciate

Il comportamento del processore è caratterizzato mostrando come sono **interallacciate** le tracce dei diversi processi

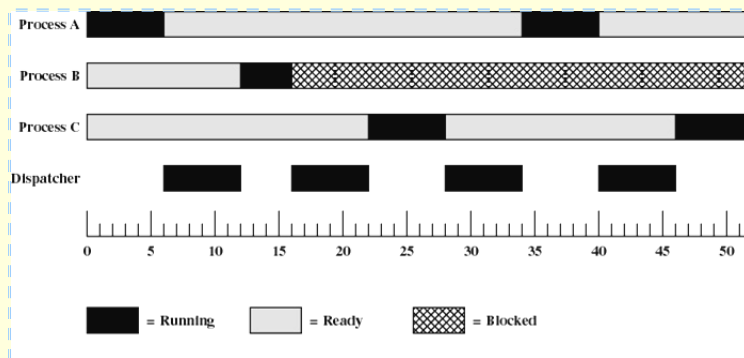
Esempio

- Indirizzo di inizio del programma allocatore:100
- Numero di istruzioni eseguite da ogni processo: al massimo 6
- Il processo B esegue 4 istruzioni, l'ultima delle quali richiede una operazione di I/O

1	5000	27	12004
2	5001	28	12005
3	5002		-----Time out
4	5003	29	100
5	5004	30	101
6	5005	31	102
	-----Time out	32	103
7	100	33	104
8	101	34	105
9	102	35	5006
10	103	36	5007
11	104	37	5008
12	105	38	5009
13	8000	39	5010
14	8001	40	5011
15	8002		-----Time out
16	8003	41	100
	-----I/O request	42	101
17	100	43	102
18	101	44	103
19	102	45	104
20	103	46	105
21	104	47	12006
22	105	48	12007
23	12000	49	12008
24	12001	50	12009
25	12002	51	12010
26	12003	52	12011
			-----Time out

Operazioni sui processi

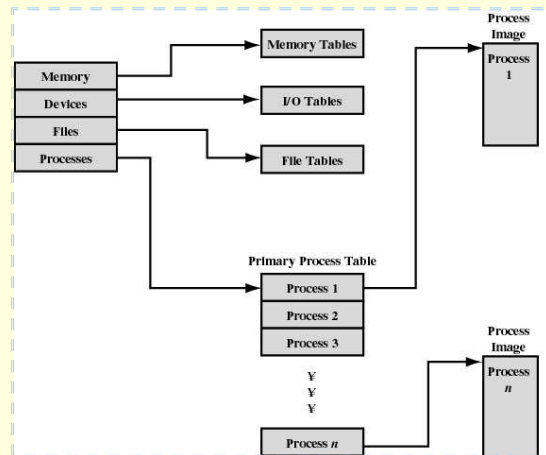
Stati dei processi



Operazioni sui processi

▣ Strutture di controllo dei processi

❖ Per gestire processi e risorse il S.O. costruisce **tabelle** di informazioni relative a ciascuna delle entità che gestisce



A.M.Fanelli

Architettura dei Sistemi a.a. 2008-09

17

Operazioni sui processi

▣ Process Image (Immagine del processo)

- ❖ **Programma utente**
 - Programma o insieme di programmi che devono essere eseguiti
- ❖ **Dati utente**
 - Insieme di locazioni per i dati, per le variabili globali o locali, per le costanti
- ❖ **Stack di sistema**
 - Uno o più stack utilizzati per memorizzare parametri ed indirizzi per le chiamate di procedure
- ❖ **Process control block**
 - Dati che servono al S.O. per controllare il processo
 - Identificatori del processo, informazioni sullo stato del processo, informazioni di controllo

A.M.Fanelli

Architettura dei Sistemi a.a. 2008-09

18

Operazioni sui processi

□ Interazione tra processi

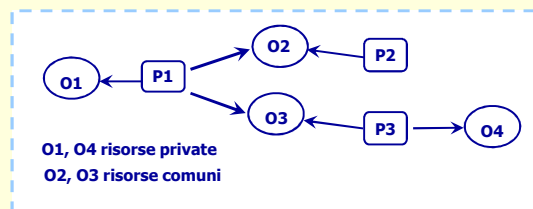
- ❖ I processi in esecuzione nel sistema possono essere indipendenti o cooperanti
 - **Processo indipendente**
 - Processo che non può influire su altri processi nel sistema o subirne l'influsso
 - **Processo cooperante**
 - Processo che influenza o può essere influenzato da altri processi in esecuzione
- ❖ Paradigma per processi cooperanti: problema del produttore-consumatore
 - Un processo produttore produce informazioni che sono consumate da un processo consumatore
 - Un programma per la stampante produce caratteri che sono consumati dal driver della stampante
 - Per permettere una esecuzione concorrente di processi consumatore e produttore occorre disporre di un **vettore di elementi** che possono essere inseriti dal produttore e prelevati dal consumatore

Operazioni sui processi

- ❖ La **esecuzione concorrente di processi cooperanti** richiede meccanismi che consentano ai processi di comunicare tra loro e di sincronizzare le proprie azioni

1. Modello ad ambiente globale

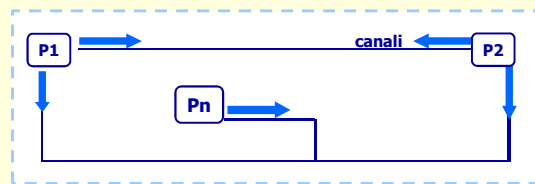
- La comunicazione tra processi può avvenire in un ambiente a memoria condivisa
- Il sistema è visto come un insieme di processi che condividono l'accesso a un vettore di memoria e ad alcune risorse



Operazioni sui processi

2. Modello ad scambio di messaggi

- Il sistema è visto come un insieme di processi ciascuno operante in un ambiente locale che non è accessibile direttamente a nessun altro processo
- La comunicazione tra processi può avvenire mediante un sistema di comunicazione tra processi (ICP)
 - ICP fornisce le operazioni **send** (messaggio) e **receive** (messaggio)
 - ICP è particolarmente utile in un ambiente distribuito dove i processi comunicanti possono risiedere in diversi calcolatori connessi in rete



Processi e Thread

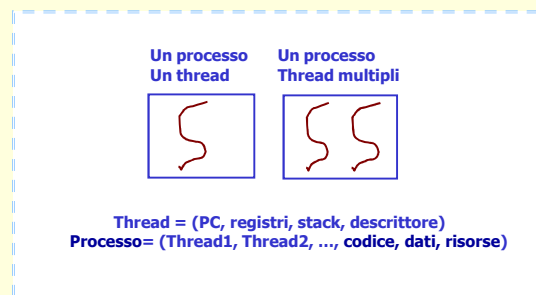
□ Processi e Thread

- ❖ Ogni processo ha il proprio spazio di indirizzamento, che contiene l'immagine del processo, e ha un flusso di esecuzione
- ❖ Il concetto di processo è basato su due aspetti indipendenti, che possono essere gestiti dal S.O. in modo indipendente:
 1. possesso delle risorse
 2. esecuzione
- ❖ La nozione di processo in molti S.O. attuali incorpora due concetti distinti ed indipendenti
 1. **Process o heavy process**
 - Processo come elemento che possiede delle risorse
 2. **Thread o lightweight process**
 - Processo come flusso di esecuzione delle istruzioni ovvero processo a cui è stata assegnata la CPU
 - Nei S.O. tradizionali ogni processo è composto da un solo thread (MS-Dos, UNIX)

Processi e Thread

□ **Processi Multithread**

- ❖ Un processo multithread, può avere **più flussi (percorsi) d'esecuzione**, thread, ognuno dei quali può eseguire un diverso insieme di istruzioni
 - Tutti i thread appartenenti allo stesso processo condividono **codice, dati, risorse di I/O**
 - Ogni thread ha il proprio stack ed il proprio descrittore
 - Tutti i S.O. moderni assumono l'esistenza di processi multithread



Processi e Thread

- ❖ La **organizzazione multithread** è vantaggiosa sia nella strutturazione delle applicazioni che per le prestazioni
 - Un applicazione si codifica come un processo comprendente più thread
 - Un programma di elaborazione di testi potrebbe avere un thread per la rappresentazione grafica, uno per la lettura dei dati immessi da tastiera ed uno per la correzione ortografica
 - Nei sistemi con più processori i thread possono essere eseguiti in parallelo
- ❖ Modi per implementare i thread
 - 1. A livello utente (user thread)**
 - Gli user thread vengono supportati sopra il kernel e sono implementati da una **Thread Library a livello utente**
 - La Thread Library fornisce supporto per la creazione, lo scheduling e la gestione dei thread senza alcun intervento del kernel
 - Il S.O. gestisce solo i processi pesanti
 - 2. A livello kernel (Kernel thread)**
 - Il S.O. gestisce direttamente i thread
 - La creazione, lo scheduling e la gestione dei thread sono implementati a livello kernel
 - La implementazione risultante è più lenta, perché richiede passaggio da livello utente a livello supervisore