Sistemi operativi I

Leonardo Ganzaroli

Indice

Introduzione

Questi appunti sono derivanti principalmente dalle slide del corso di *Sistemi operativi I* che ho svolto durante la laurea Triennale di informatica all'università "La Sapienza".

N.B. Questo corso è il naturale proseguimento di *Architettura degli elaboratori*, quindi molte cose saranno date per scontate.

1 Processi

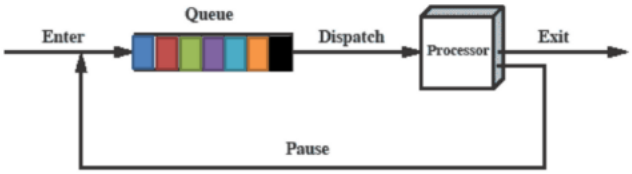
Definizione La traccia di un processo è la sua sequenza di istruzioni che vengono eseguite.

Definizione Il Process Image di un processo è l'insieme di:

- **PCB**
- **Programma sorgente**
- **Dati**
- **Stack delle chiamate**

Per gestire opportunamente i processi il SO mantiene tutti i processi attivi in una tabella, per distinguerli viene assegnato un codice identificativo (PID) ad ognuno.

Definizione Il dispatcher è un programma che sospende un processo per farne eseguire un altro, usa una o più code per alternare i processi:



Questo presuppone che ogni processo si trovi in un certo stato in ogni istante, scegliendo quali e in che modo si passi da uno stato all'altro porta a diversi modelli, un possibile modello a 5 stati è:

Per mantenere queste informazioni aggiuntive si estende il PCB in modo che contenga anche:

- Identificatori
  - **PID**
  - **PPID**
  - **Utente proprietario**
- Informazioni sullo stato del processore
  - **PSW**
  - **Contenuto registri**
- Informazioni per il controllo del processo
- Supporto per strutture dati
- Risorse usate
- Permessi speciali
- Metodi di comunicazione con altri processi

Ogni processore ammette almeno 2 modi di esecuzione per i processi:

1. **Utente**

   Usato per i programmi avviati dall'utente, non permette di svolgere alcune operazioni.

2. **Sistema**

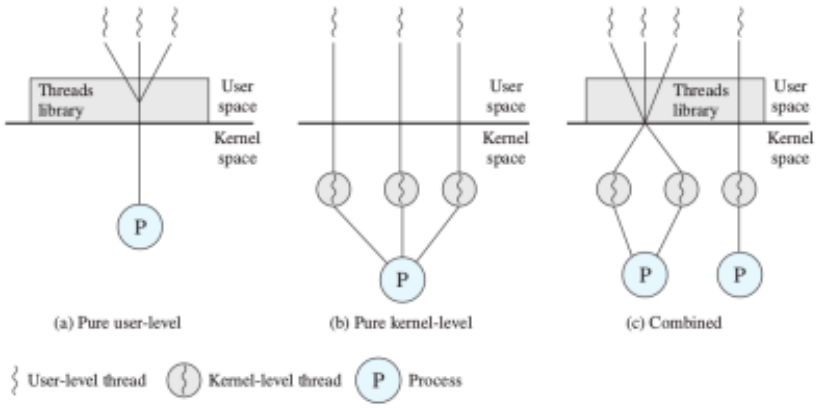
   Usato dal Kernel, permette di avere pieno controllo ed accedere ad ogni locazione di memoria.

I processi utente possono essere portati in modalità sistema temporaneamente, questo accade quando c'è bisogno di gestire un'interruzione.

Il SO stesso è trattato come ogni altro programma, l'unica differenza sta nel fatto che possiede dei privilegi più alti. Molto spesso lascia volontariamente il posto ad altri processi per poi riprenderselo tramite interrupt. In particolare i suoi processi vengono eseguiti a livello utente.

Il Kernel invece viene eseguito al di fuori dei processi ed ha la sua zona di memoria dedicata.

I Thread condividono tutte le risorse del processo padre tranne la stack delle chiamate, possono essere di 3 tipi ed ognuno ha i suoi vantaggi e svantaggi:



1.1 Scheduling

Lo scopo dello scheduling è quello di assegnare i processi da eseguire ad ogni processore, deve farlo nel modo più efficiente possibile ma allo stesso tempo equo per tutti i processi.

Ne esistono 4 tipi diversi:

1. **I/O**

   Decide l'assegnazione dei dispositivi I/O.

2. **A lungo termine**

   Decide quali processi vengono aggiunti a quelli da eseguire.

3. **A medio termine**

   Decide l'aggiunta di processi nella RAM.

4. **A breve termine** (dispatcher)

   Decide quale processo tra quelli pronti viene eseguito.

L'ultimo è quello eseguito più frequentemente essendo chiamato in risposta agli eventi, i modi per valutare le diverse politiche utilizzabili in questo caso si basano su diversi criteri:

- Utente
  - Prestazionali
    - **Tempo di ritorno**
      Tempo passato tra la creazione ed il completamento di un processo.
    - **Tempo di risposta**
      Tempo passato tra la sottomissione di una richiesta e l'inizio della risposta.
    - **Deadline**
      Numero di deadline rispettate.
  - Non prestazionali
    - **Predicibilità**
      Variabilità nei tempi di risposta/ritorno.
- Sistema
  - Prestazionali
    - **Throughput**
      Numero di processi completati per unità di tempo.
    - **Uso del processore**
      Tempo di utilizzo del processore.
  - Non prestazionali
    - **Equità**
      Equità nella gestione dei processi.
    - **Gestione delle priorità**
      Rispetto delle diverse priorità dei processi.
    - **Bilanciamento delle risorse**
      Uso delle risorse.

1.1.1 Politiche

Definizione La funzione di selezione si occupa di scegliere il processo da mandare in esecuzione.

Definizione La modalità di decisione specifica in quali istanti di tempo viene invocata la precedente, può essere:

- **Preemptive**

  Si può interrompere il processo in qualsiasi momento.

- **Non preemptive**

  Si può interrompere solo alla fine dell'esecuzione o in presenza di richieste I/O.

Principali politiche di Scheduling:

- **FCFS**
  - Alla fine dell'esecuzione si prende il primo processo della coda
  - Non preemptive
  - Processi lunghi favoriti
  - Processi CPU-Bound favoriti
- **Round robin**
  - Usa un clock
  - Ogni processo ha uno slot temporale
  - Alla fine del suo tempo il processo viene rimesso nella coda
  - Preemptive
- **Round robin virtuale**

  Come il precedente ma presenta una coda aggiuntiva ad alta priorità, vengono messi in questa coda i processi dopo il completamento di una richiesta I/O con il loro tempo di slot residuo.

- **SPN**
  - Si sceglie il processo con tempo stimato di esecuzione minore
  - Non preemptive
  - Processi corti favoriti
- **SRT**

  Come il precedente ma preemptive rispetto all'arrivo di un nuovo processo.

Nel caso di architetture multiprocessore bisogna gestire anche l'assegnazione dei processi:

- **Statica**

  Gliene viene assegnato uno alla sua creazione.

- **Dinamica**

  La sua esecuzione può variare durante la sua vita.

2 Concorrenza

Definizione Un'operazione atomica è una sequenza indivisibile di comandi.

Definizione Una sezione critica è una parte di codice che presenta un accesso esclusivo ad una risorsa condivisa.

Definizione La mutua esclusione è un requisito che impone che un solo processo alla volta sia in una sezione critica.

Definizione Una race condition è la violazione della mutua esclusione.

Definizione Il deadlock si verifica quando due o più processi non possono proseguire con l'esecuzione perché si aspettano l'un l'altro.

Definizione Il livelock si verifica quando due o più processi cambiano continuamente il loro stato (senza fare niente di utile) in risposta agli altri.

Definizione La starvation si verifica quando un processo *Ready* non viene mai scelto dallo scheduler.

Comunicazione	**Relazione**	**Problemi**
Nessuna	Competizione	Mutua esclusione, deadlock, starvation
Memoria condivisa	Cooperazione	Mutua esclusione, deadlock, starvation, coerenza dati
Comunicazione primitiva	Cooperazione	Deadlock, starvation

Tabella 1: Interazione tra processi

2.1 Mutua esclusione

Per garantire la mutua esclusione ci sono diversi metodi:

- **Disabilitazione interruzioni** (Solo monoprocessore)

  Se un processo può decidere di non essere interrotto allora nessuno lo può interrompere nella sezione critica. (Possibile peggioramento prestazioni)

- **Istruzioni speciali**

  Si usano delle istruzioni speciali (atomiche) ed una variabile per permettere ad un solo processo alla volta di entrare nella sezione critica.

- **Semafori**

  Una struttura contenente una coda di processi ed un contatore, ha 3 operazioni atomiche associate:

  1. **Initialize**
  2. **P**
  3. **V**

  Quando un processo richiede l'accesso alla risorsa:

  1. Chiama P
  2. Se la risorsa è totalmente occupata viene bloccato e messo in coda
  3. Quando un altro processo finisce chiama V
  4. Se la coda non è vuota sveglia un processo e rimuovilo da essa

  Se il contatore assume solo i valori $0, 1$ il semaforo viene detto binario.

  In base al modo di prelevare i processi dalla coda si identificano 2 tipi:

  1. Sem. forte (FIFO)
  2. Sem. debole (Non specificato)

- **Alg. Dekker** (2 processi)

  Usa 2 flag (una per processo) e una variabile *turno*, $flag_i = true$ indica che il processo i potrebbe essere nella sua fase critica.

  Prima della sezione critica un processo controlla la flag dell'altro e se è *true* aspetta, se il valore di *turno* è il numero dell'altro processo imposta la propria flag $false$.

  Dopo la sezione critica un processo imposta la sua flag $false$ e *turno* al numero dell'altro processo.

- **Alg. Peterson** (2 processi)

  Versione semplificata del precedente, generalizzazione più semplice.

2.2 Interazione

I processi possono comunicare tra di loro (e potenzialmente sincronizzarsi) tramite l'invio di messaggi, lo scambio avviene tramite le operazioni atomiche:

- **Send**
- **Receive**

Lo scambio di un messaggio può essere:

- **Bloccante**

  I 2 processi coinvolti si bloccano fino alla ricezione del messaggio.

- **Non bloccante**

  La ricezione può essere bloccante o meno, l'invio non lo è.

Per indirizzare il messaggio ci sono 2 metodi:

1. **Diretto**

   Si specifica un identificatore, ogni processo ha una sua coda.

2. **Indiretto**

   I messaggi vengono inviati ad una mailbox condivisa, va creata esplicitamente.

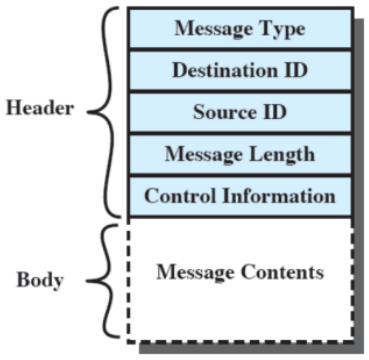


Figura 1: Formato tipico messaggio

In generale tutti i metodi visti sono intercambiabili tra loro per quanto riguarda la condivisione di risorse.

2.3 Deadlock

Le risorse si possono classificare in:

- **Riusabili**
  - Usabili da un solo processo alla volta
  - Non vengono "consumate"
- **Non riusabili**
  - Vengono create e distrutte

Il deadlock nei 2 casi può avvenire rispettivamente se:

- Un processo che sta usando una risorsa ne richiede un'altra
- Viene richiesta una risorsa non ancora creata

Lo stato delle risorse e dei processi può essere rappresentato tramite un grafo in cui:

- Un cerchio rappresenta un processo
- Un rettangolo rappresenta una risorsa
- Un pallino nel rettangolo rappresenta un'istanza della risorsa
- Un arco Processo → Risorsa è una richiesta
- Un arco Risorsa → Processo è un'assegnazione

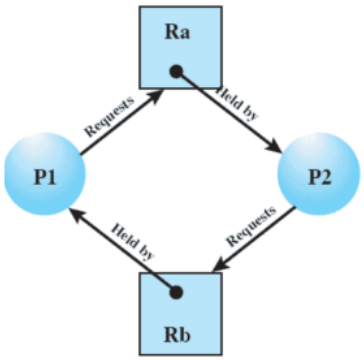


Figura 2: Esempio di deadlock

Oltre a cercare di prevenirlo si può provare ad evitarlo (ad esempio con l'algoritmo del banchiere), nel caso si verifichi comunque si possono terminare i processi coinvolti, usare punti di ripristino oppure prendere forzatamente le loro risorse.

3 I/O

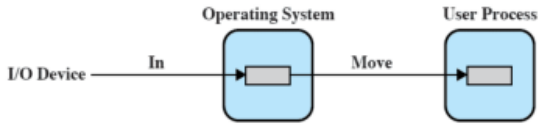
Data l'esistenza di svariati tipi di dispositivi I/O che possono presentare caratteristiche anche estremamente diverse tra loro risulta complicato catalogarli, a grandi linee però si possono dividere in:

- Comunicazione diretta con l'utente
- Comunicazione con dispositivi elettronici
- Comunicazione con dispositivi remoti

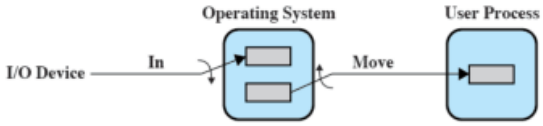
Un'altra possibilità è l'uso di una gerarchia che presenta delle similitudini con i protocolli usati nelle reti:

1. **Dispositivi locali**
2. **Dispositivi di comunicazione**
3. **File System**

Per evitare situazioni di deadlock dovute a richieste I/O delle pagine in memoria si può usare il *buffering*, ossia avere dei buffer in memoria (spazio di sistema). Questo potenzialmente permette di avere l'input già pronto prima della richiesta e di posticipare l'output.



Singolo



Doppio

Circolare

3.1 Gestione del disco

Dato il suo funzionamento il disco rigido presenta dei tempi di ricerca/scrittura che possono risultare lunghi, usando opportune politiche di gestione delle richieste si può ottimizzare il processo:

- **RSS**

  Scheduling random.

- **FIFO**
- **LIFO**
- **Priorità**
- **SSTF**

  Si sceglie la richiesta che provoca il minore spostamento della testina.

- **SCAN**

  Si scelgono le richieste in modo che la testina vada sempre in un verso, quando arriva al bordo procede nel verso opposto.

- **C-SCAN**

  Come il precedente ma nel ritorno non si svolgono richieste.

- **FSCAN**

  Ogni nuova richiesta deve aspettare che tutte le precedenti vengano eseguite, si usano 2 code per fare ciò.

- **N-step SCAN**

  Generalizzazione del precedente con più di 2 code.

Un ulteriore meccanismo prevede di usare un buffer in memoria per mantenere alcune settori del disco, le politiche di rimpiazzo sono le stesse della Cache.

3.1.1 RAID

Definizione Uno strip è un insieme di settori.

Definizione Uno stripe è un insieme di strip.

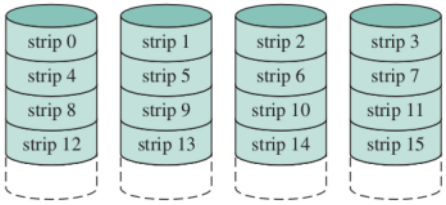
Definizione Un blocco è uno strip potenzialmente grande.

Definizione Il *Redundant Array of Independent Disks* è un metodo che permette di usare un insieme di dischi come se fossero uno.

Le configurazioni principali sono:

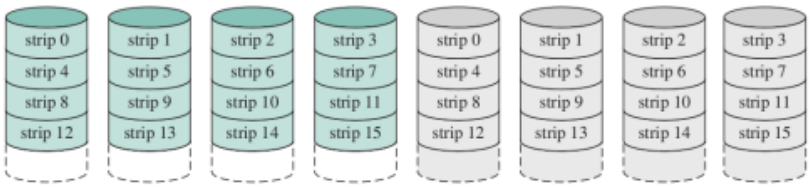
- **RAID0**

  I dati vengono distribuiti per avere un accesso più efficiente.



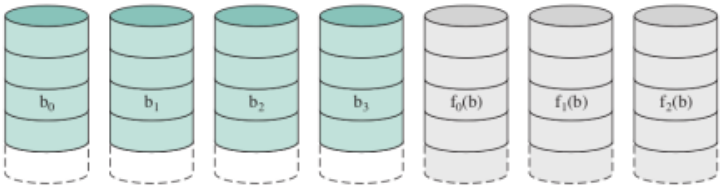
- **RAID1**

  Come il precedente ma i dati vengono duplicati.



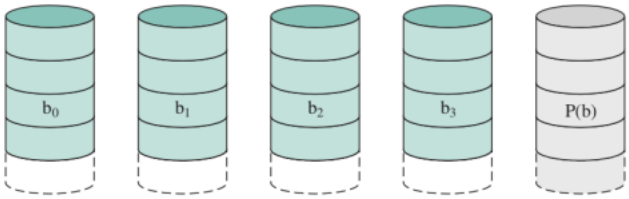
- **RAID2**

  Ridondanza tramite codice di Hamming.



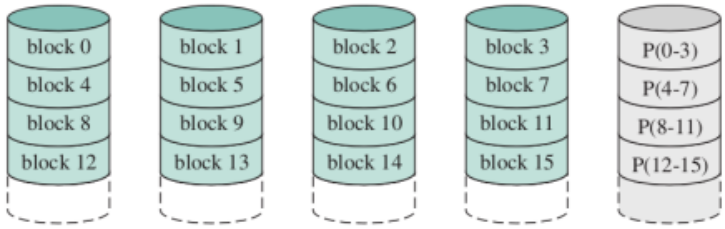
- **RAID3**

  Ridondanza (byte) tramite un disco che contiene le parità.



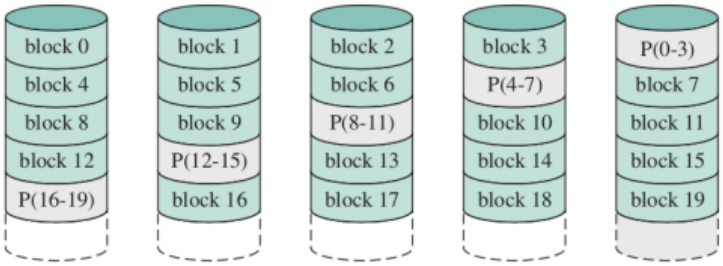
- **RAID4**

  Generalizzazione del precedente.



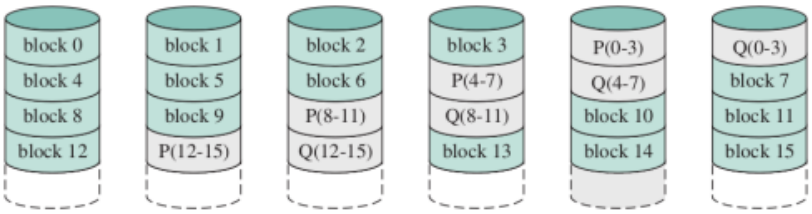
- **RAID5**

  Come il precedente ma senza un disco dedicato, le parità sono distribuite.



- **RAID6**

  Come il precedente ma le parità sono duplicate.



4 File system

Definizione Campo = uno o più dati base.

Definizione Metadato = dato che descrive una proprietà di un altro dato.

Definizione Record = insieme di campi correlati.

Definizione File = insieme di record correlati.

Definizione Database = collezione di dati correlati.

Definizione Volume = disco logico = insieme di settori.

Definizione Il *File Management System* si occupa di fornire servizi all'utente ed alle applicazioni che permettono di interagire con i file.

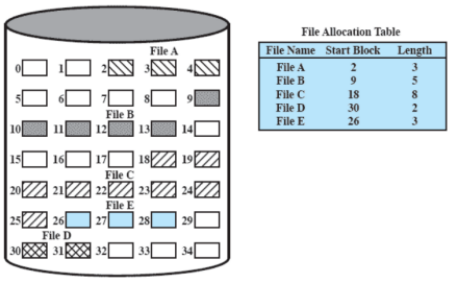
Definizione Una directory è un file speciale che contiene altri file, in particolare deve contenere tutte le loro informazioni e fornire un mapping tra un nome ed il file corrispondente.

L'assegnazione di memoria secondaria ad un file comporta gli stessi problemi visti per quella primaria, similmente si può usare un'allocazione statica/dinamica con blocchi fissi/variabili.

Per allocare i blocchi effettivi ci sono diversi metodi:

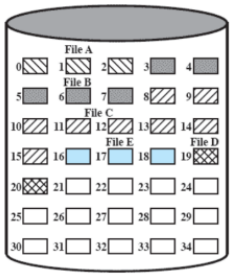
- **Contigua**

  I blocchi necessari sono tutti in sequenza, possibile frammentazione esterna.



File Allocation Table

File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3



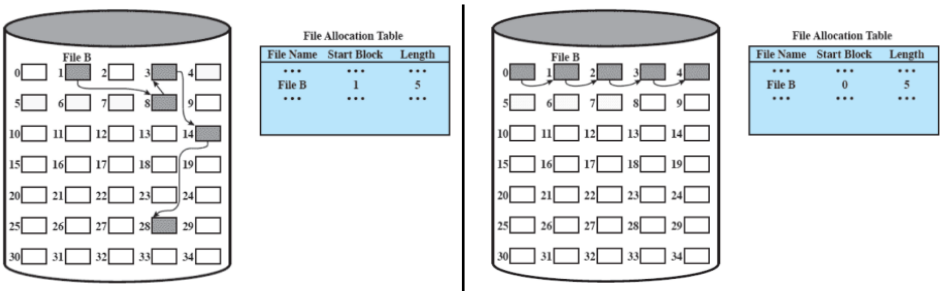
File Allocation Table

File Name	Start Block	Length
File A	0	3
File B	3	5
File C	8	8
File D	19	2
File E	16	3

Dopo compattazione

- **Concatenata**

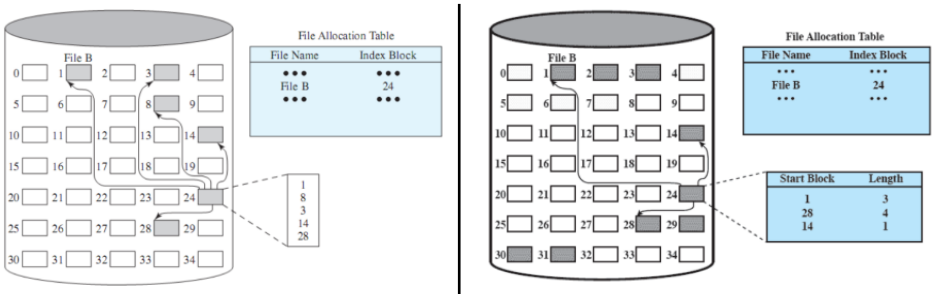
  Ogni blocco contiene un puntatore al prossimo.



Dopo consolidamento

- **Indicizzata**

  Viene usato un blocco indice che contiene gli indirizzi degli altri blocchi.



Porzioni fisse

Porzioni variabili

Per far funzionare questi metodi bisogna mantenere una lista dei file ed una per i blocchi liberi, quest'ultima può essere implementata tramite:

- **Lista dei blocchi liberi**
- **Tabella di bit**

  Vettore con un bit associato ad ogni blocco, 0 se libero.

- **Porzioni libere concatenate**

  Si concatenano i blocchi liberi con i puntatori, bisogna specificare la dimensione del blocco puntato.

- **Indicizzazione**

  Tutto lo spazio libero viene visto come un file.

Definizione Il Journaling è una tecnica per preserva l'integrità dei dati in caso di problemi, prima di effettuare un'operazione viene scritta la stessa in una zona designata in memoria (log). In questo modo se per esempio non viene portata a termine per uno spegnimento improvviso il SO la porterà a termine alla successiva accensione leggendo il log.