


PARTE 1 → introduzione (skip)

interruzioni

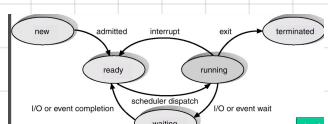
multitasking e
time - sharing

PARTE 2: STRUTTURA DEI (skip) SISTEMI OPERATIVI

di cosa si occupa un SO?

PARTE 3: GESTIONE DEI PROCESSI

PROCESSI



- QUELLO CHE SI SPosta E' IL PCB NON IL PROCESSO.

- PROBLEMA PRODUTTORE-CONSUMATORE

- IPC (messaggi o shared memory)

SCRIBULDUM

CON DIRITTO DI PRECENDERE: SO può decidere quale processo montere in esecuzione secondo un puzzle arbitrario (es: la priorità)

SENZA DIRITTO DI PRECENDERE: SO determina solo se il processo è nominato oppure quale volontariamente uscirà dalla CPU.

- THROUGHPUT: numero di processi in media completati in una certa unità di tempo.

- TEMPO DI RISPOSTA: tempo tra quando si avvia un processo e quando effettivamente comincia ad eseguire.

- TURNAROUND: tempo medio di completamento di un processo (il momento entra in coda di ready per la prima volta a quando termina).

- WAITING TIME: tempo di attesa (somma del tempo passato in RQ).

FCFS; → NON IMPLEMENTABILE!
***SJF;** → PRIORITY SCHEDULING;
ROUND ROBIN; → STARVATION
MULTILEVEL QUEUE;
MULTILEVEL FEEDBACK QUEUE;

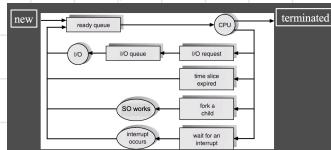
- STARVATION: in RQ c'è ogni volta un processo con priorità maggiore e precede il processo P_x non verrà mai eseguito!

- AGIJA: meccanismo di controllo alla starvation, la priorità di P_x non aumenta man mano che passa il tempo.

SCHEDULING DELLA CPU

- CONTEXT SWITCH

- CODA DI PROCESSI



THREAD

CAP 4
DOPO
CAP 3

SINCRONIZZAZIONI TRA PROCESSI

SEZIONI CRITICA

ATTESA
INFINTA

① MUTUA ESCLUSIONE: se più SO sono nella propria sezione critica ma non ne è ancora uscito, nessun altro processo P_J può entrare nella propria SC.

② PROCESSO: se un processo lascia la propria sezione critica e deve permettere ad un altro processo di entrare nella propria. EVITA DEADLOCK

③ ATTESA LIMITATA: se un processo P_i ha già eseguito la propria entry critica, evita un limite al numero di volte in cui altri SO possono entrare nelle loro SC prima che tocchi a P_i. EVITA STARVATION

KERNEL CON DIRITTO DI PRECENDERE

SEZIONI CRITICHE DEL SO

KERNEL SENZA DIRITTO DI PRECENDERE

- BUSY WAITING: un processo che attende il proprio turno per accedere alla propria SC occupa CPU inutilmente



SEMAFORI

PROBLEMI:

- PRODUTTORE-CONSUMATORI

- LETTORI E SCRITTORI

- 5 FICOSOFI

PARTE 4: GESTIONE DELLA MEMORIA

- 4.1 MEMORIA CONTRALCE

- SWAPPING:** avvicendamento di processi (utilizzando l'AREA di SWAP). Diene risorsamente ai moderni SO.
- BINDING DEGLI INDIRIZZI:** operazione di associazione di variabili ed istruzioni agli indirizzi di memoria. Si può fare in 3 fasi:

- in compilazione

- vengono generati codice assoluto o relativo;
- il compilatore deve conoscere l'indirizzo di memoria a partire del quale verrà caricato il programma, per effettuare il Binding;
- se il SO deve temporaneamente scaricare il programma da RAM, quando lo ricarica da RAM deve metterlo esattamente dove si trovava prima;

- in caricamento

- vengono generati codice dinamicamente rilocabile;
- il compilatore ad istruzione è consapevole degli indirizzi relativi rispetto all'inizio del programma, che inizia da un indirizzo virtuale;
- gli indirizzi assoluti finali vengono generati in fase di caricamento da RP;
- il Binding viene effettuato in fase di caricamento in RP: se il programma viene tolto da RP, si può ricaricarlo in una posizione diversa reapplicandone la fase di caricamento.

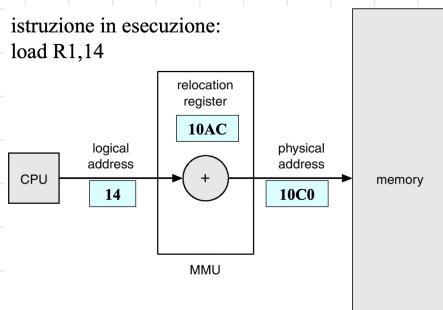
- in esecuzione

- vengono generati codice dinamicamente rilocabile;
 - si usano sempre e solo indirizzi relativi;
 - un indirizzo relativo viene trasformato in assoluto solo quando l'esecuzione viene eseguita;
 - è necessario appunto supporto HW.
- 5) BINDING DINAMICO DEGLI INDIRIZZI

uso il REGISTRO DI RICARICA

- Insieme degli indirizzi compresi tra 0 e l'indirizzo MAX dell'ultima cella di memoria occupata dal programma

⇒ SPAZIO DI INDIRIZZAMENTO LOGICO
o VIRTUAL



- Insieme degli indirizzi fisici usati dal programma

⇒ SPAZIO DI INDIRIZZAMENTO FISICO

LIBRERIE

STATICHE

vengono collegate al codice del programma e diventano parte dell'esecutibile

DINAMICHE

vengono caricate in RAM solo se il programma che le usa chiama una delle subroutine delle librerie di cui ha bisogno

TECNICHE DI GESTIONE DELLA MP

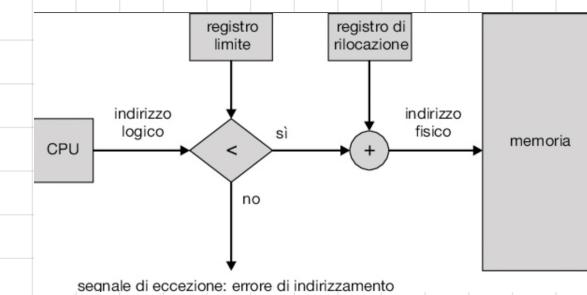
SWAPPING

- è un trattamento, non può utilizzarsi ma l'idea di fondo rimane valida

ALL. CONT. A

PART. FISSA

- memoria divide tra SO e programmi
- per proteggere la MP viene usato un **REGISTRO LIMITE** che viene installato nel SO e confrontato con ogni indirizzo logico del programma.



ALL. CONT. A

A PART. VARIABILI

- MP divide in parti di dimensione fissa non per forza tutte uguali
- il grado di multiprogrammazione è limitato dal numero di parti

- **FRAZIONATI DI INTESA:** difficilmente un processo ha esattamente le dimensioni delle parti, la parte rimanente va sprecata.

- **FRAZIONATI DI ESTESA:** somma delle spese spese della frammentazione intorno di ogni partizione

■ ALLOCAZIONE A PARTIZIONI: MULTIPLEX VARIABILI

- l'SO deve tenere tracce di tutti i buchi in RAM e assegnare ad un nuovo processo una partizione sufficientemente grande seguendo uno di questi criteri:

Best fit

Worst fit

First fit

- la frammentazione esterna rimane

- la frammentazione interna rimane perché cada troppo tenere tracce di buchi molto piccoli.



si può ricomporre la memoria ma anche questo è un processo che costa molto.

PAGINAZIONE

- METODO DI BASE

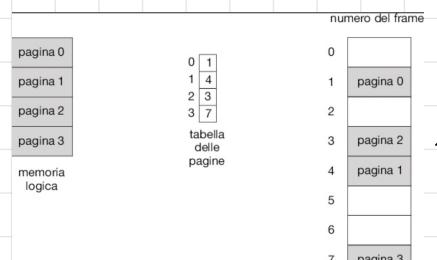
HP è divisa in "pezzi" detti FRATELLI tutti della stessa dimensione. Il spazio di indirizzamento logico è sempre visto come uno spazio contiguo ma in realtà è diviso in PAGINE di dimensione fixed che si fanno.

I frame possono non essere adiacenti.

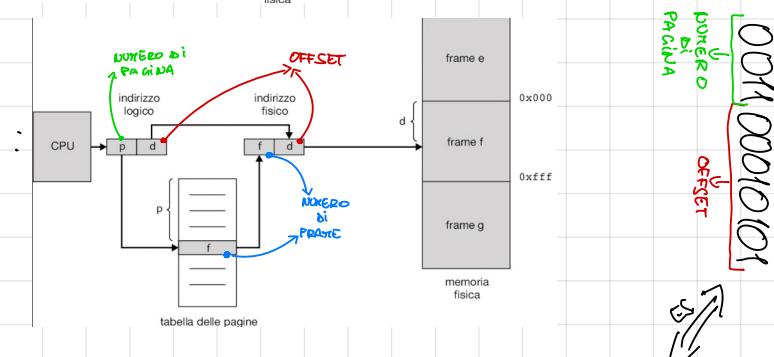
Da ogni processo è associata una TABELLA DELLE PAGINE in cui le entry sono il numero di pagina e il contenuto delle entry sono il numero di frame.

Gli indirizzi devono avere coppie chiave - valore in cui chiave = numero di pagina e valore = offset.

Quindi sarà la traduzione avvenire così:



Così facendo però gli indirizzi relativi sembrano non funzionare più, bisogna ripensarli.

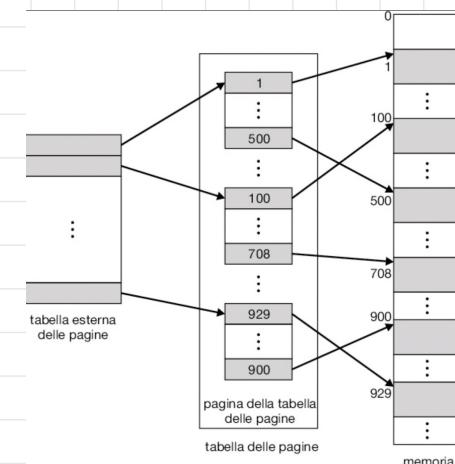


• NUMERO DI BIT SU CUI VA SCRITTO UN INDIRIZZO LOGICO → n bit → m bit per scrivere l'OFFSET in una pagina → 2^n byte
 • LA DIMENSIONE DI UN FRATELLO (E DI UNA PAGINA)

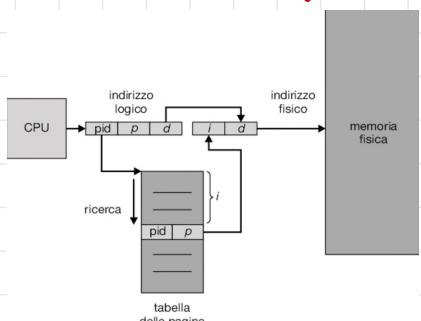
I VANTAGGI DELLA PAGINAZIONE

- viene implementata automaticamente una forma di PROTEZIONE DELLA MEMORIA;
- evita la FRAGMENTAZIONE ESTERNA;
- è una forma di RICLOCCHIAMENTO DINAMICO

PAGINAZIONE A PIÙ UVEZZI (paginare la PT)



INVERTED DAB TABS

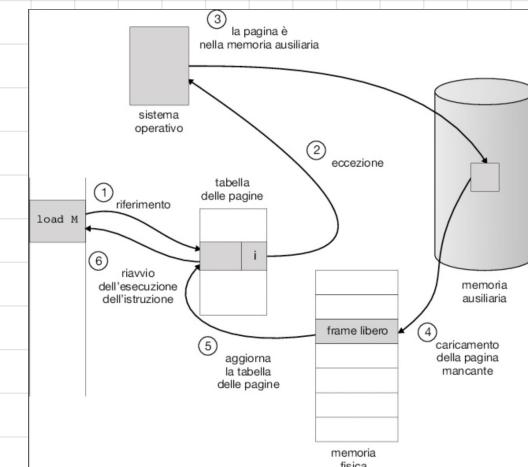


- 4.2 MEMORIA VIRTUALE

È UN INSIEME DI TECNICHE CHE PERMETTE L'ESECUZIONE DI PROCESSI IN CUI CODICI E/O DATI NON SONO STATI CARICATI COMPLETAMENTE IN MEMORIA PRINCIPALE.

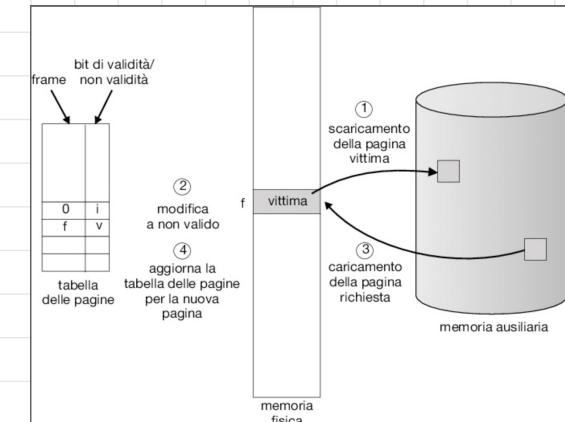
L'idea è quella di portare in RP una pagina solo al momento del primo utilizzo. Aumento di reale localizzazione di memoria delle pagine stesse.
Se la CPU esegue un'istruzione di una pagina non presente in RP, il processo genera un **PAGE FAULT**.

Ogni pagina ha un **BIT DI VALIDITÀ** che è 0 o 1 se non è in RP.



PURE NESSUNO PAGINA → i processi vengono fatti partire con 0 pagine in RP.

La memoria virtuale prende richiede MMU specifico.



Questo è il caso in cui si sceglie una pagina vittima da sostituire e lo si fa con l'aiuto dell'area di swap.

Se la pagina vittima era stata modificata (si capisce grazie al **DIRTY BIT**) no prima salvato in RS.

PV = pagina virtuale

ALGORITMI DI SOSTITUZIONE DELLE PAGINE

OPT

- segue come PV quello scendente da più tempo in RP.
- soffre di BELAY

SOST ETIMICO DELL'OPT (or)

- non soffre di BELAY
- PV è quello che sarà usato in lo stesso tempo
- non soffre di BELAY e si avvicina ad OPT
- non implementabile

OLRU

- opposta all'OPT guardando indietro: PV è quello che non è stato usato da più tempo
- non soffre di BELAY e si avvicina ad OPT
- difficile da implementare

(con approssimazione)

SECONDA CYANC

- REFERENZE BIT, col 1 punto la pagina viene individuata
- se RB è 0 o 0 quello è lo PV
- se RB è al 1, dovrà essere aggiornato la prossima pagina
- nel caso peggiore è FIFO

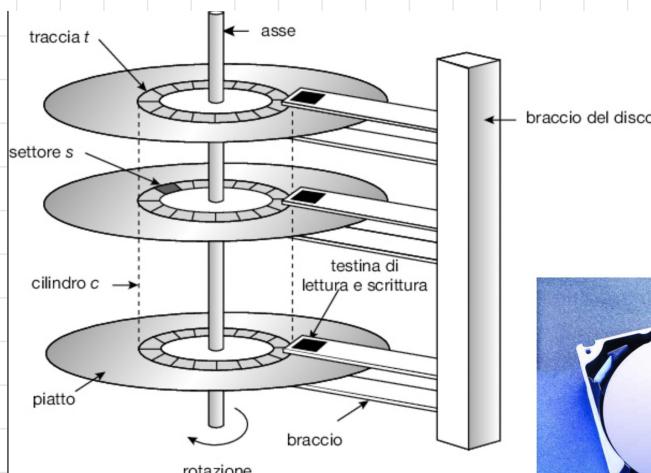
SECONDA CHIAVE MISURATO

- se RB è 0, si manda a rimpicciolire
- se RB è 1, si manda a rimpicciolire
- se RB è 0 e si è già mossa la prossima pagina
- se RB è 1 e si è già mossa la prossima pagina

RB = bit di recente ed modificato
0 = non è recente né modificato
1 = non è recente ma modificato
2 = è recente ma non modificato
3 = è sia recente sia modificato

PARTÉ 5: GESTIONE DELLA MEMORIA DI MASSA

STRUTTURA DI UN DISCO RIGIDO



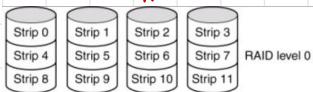
ALGORITMI DI SCHEDUOLUNG

- FCFS
- C-SCAN (coricolar scan), la testina si muove da un'estremità all'altra servendo le richieste, quando arriva all'estremità torna all'inizio senza servire altre richieste.

I SISTEMI RAID

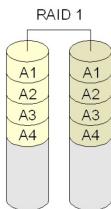
RAID 0

- suddivide i blocchi logici del disco virtuale in stripes dei K blocchi consecutivi
- gli stripes vengono suddivisi fra i blocchi (STRIPPING)
- aumentano le prestazioni e la capacità non aumenta l'affidabilità



RAID 1

- per ogni disco virtuale mola un secondo disco di riserva
- aumenta l'affidabilità



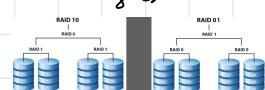
RAID 0+1

- STRIPPING + MIRRORING
- se un disco si rompe si controlla quello di mirroring
- tra le più affidabili, prestazioni proporzionali al numero di dischi coinvolti



RAID 10

- mosaico dello 01 ma con striping e mirroring vengono fatti in ordine inverso
- prestazioni simili a 0+1 con alcuni vantaggi in meno di questi



RAID 4

- risparmia dischi rispetto allo 01 e garantisce il mantenimento dei dati ma è meno efficiente

RAID 5

- fornisce la migliore combinazione di partizioni, affidabilità e capacità di memorizzazione

RAID 6

- risolve anche il guasto di 2 dischi contemporaneamente ma è un evento poco probabile perché è un sistema poco usato.

PARTIE 6: INTEGRACCIA DEI FILE SYSTEM

ASSOCIARE AI FILE I DATI ATTRIBUSSI

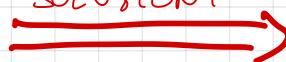
- ① inserire a fianco del nome del file i suoi attributi.
- ② inserire a fianco del nome del file un puntatore ad una struttura interna che contiene gli attributi.
- ③ NTFS: directory organizzate in un **albero di ricerca bilanciato** in cui le foglie sono i file contenuti nella directory.
Ogni foglia contiene il file ed un puntatore ad una struttura interna che contiene i suoi attributi.

2) Dallocazione concatenata

- Catena di blocchi in cui ogni blocco contiene un puntatore al blocco successivo.
- per accedere ai blocchi che contengono il file bisogna memorizzare il numero del blocco iniziale.
- nell'ultimo blocco viene scritto un numero segnale oppure 0.

- VANTAGGI**
- non sono necessari blocchi contigui (non si ricompatta, non c'è flusso esterno)
 - qualsiasi blocco libero può essere usato per memorizzare una porzione di file
- SVANTAGGI**
- l'ultimo blocco è un blocco "speciale"
 - occorre scrivere direttamente sull'intero blocco
 - dallocazione poco affidabile: se si perde un blocco si perde anche tutta quella successiva.

SOLUZIONI



LISTE DOPPIAMENTO CONCATENATO:

se si danneggia un blocco si possono recuperare gli altri scrivendo la catena adindietro (bisogna memorizzare il numero dell'ultimo blocco)

MIGRARE IN UNI BLOCCO IL NOME DEL FILE E LA POSIZIONE DEL BLOCCO ALL'INTERNO DELLA CATENA:
scrivendo tutti i blocchi del disco si possono recuperare quegli dati

CONSIDERARE TUTTO L'HD FORNITO DA CLUSTER DI BLOCCHI ADIACENTI

FAT

Arrey in cui ogni indice corrisponde ad un blocco; se contiene 0 il blocco è vuoto.
Quando bisogna memorizzare il numero del primo blocco che contiene i dati del file.

VANTAGGI

- deve essere tenuta in RAM, così l'economia di spazio migliora sensibilmente.
- in caso di perdita di un blocco il sistema è più sicuro.
- la gestione dei blocchi liberi è automatica.

SVANTAGGI

- essendo in RAM, taglia spazio ai processi.
- occupa spazio in RAM.
- diretta problematica con i file grandi.
- se viene perso non c'è modo di accedere ai file.

3) Dallocazione indirizzata

Si tiene traccia di tutti i blocchi in cui è contenuto un file scrivendo il loro numero su un altro blocco (**blocco indice**)

Per recuperare i dati, bisogna memorizzare gli attributi del file e il numero del suo blocco indice

- VANTAGGI**
- non sono necessari blocchi contigui
 - l'accesso diretto è efficiente

- SVANTAGGI**
- si usa un intero blocco per salvare gli indici, se non nasce a contenervi

SCHEMA CONVENTIONAL
l'ultimo entry del blocco indice punta ad un altro blocco indice

SCHEMA A PIÙ LIVELLI
il blocco indice contiene solo puntatori ad altri blocchi indice

Associazione dati file

① **Dallocazione contigua**: ogni file è allocato in blocchi contigui dell'HD. Per recuperarlo bisogna memorizzare gli attributi del file il numero del primo blocco del file e quanti blocchi occupa

VANTAGGI

- accesso ai file semplice e veloce.
- serve memorizzare poche informazioni per sapere dove si trova al file sull'HD.

SVANTAGGI

- servono blocchi adiacenti
- frammentazione esterna
- necessaria ricompattazione
- se la dimensione di un file aumenta bisogna rialloca o sovraccaricamento (fan out)

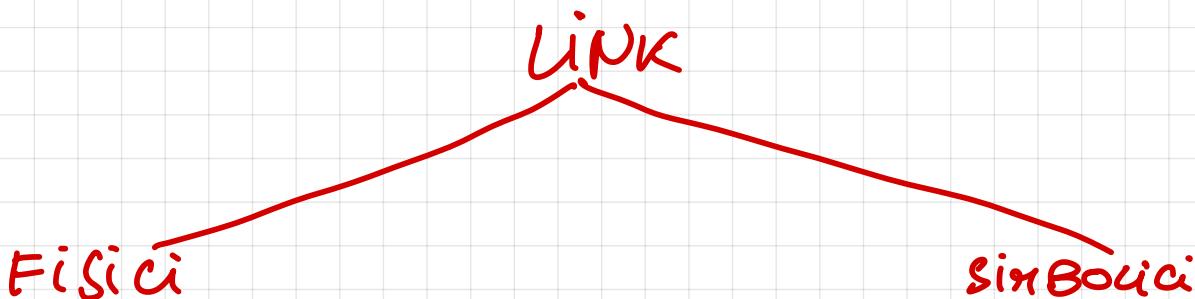
VARIANTO → i-node UNIX

Od ogni file è associato una struttura intorno girata direttamente dal so che contiene i suoi attributi.
E così formato:

- 10 puntatori **diretti** a blocchi di dati del file.
- un puntatore **single indirect** che punterà ad un blocco indice che conterrà puntatori a blocchi di dati file
- un puntatore **double indirect** che punterà ad un blocco indice che conterrà puntatori a blocchi indice ognuno dei quali conterrà puntatori a blocchi di dati del file
- Un puntatore **triple indirect** che punterà ad un blocco indice che conterrà puntatori a blocchi indice ognuno dei quali conterrà puntatori a blocchi indice, ognuno dei quali conterrà puntatori a blocchi di dati del file

NTFS

Ogni file è descritto da un **ENTRY** numerati consecutivamente.
Tutti gli elementi sono contenuti in una **MASTER FILE TABLE (MFT)**.
Il numero dell'elemento è associato al nome del file ed è detto **FILE REFERENCE**.



- È la stringa di caratteri associata all'i-node;
- Un file regolare può avere più link fisici (ci possono essere più **ENTRY** diversi stessa cartella o in cartelle diverse a cui è associato lo stesso i-node).
- UNA STESSA CARTELLA NON PUÒ CONTENERE 2 ENTRY CON LO STESSO NOME E LO STESSO I-NODE
- NON SI POSSONO CREARE LINK FISICI TRA CARTELLE

- i link simbolici sono forniti da cartelle
- viene creato un nuovo i-node e qui dentro viene annotato il file che si è associato ed un link simbolico
- non invocalemente l'i-node del file sorgente