APPUNTI SISTEMI OPERATIVI

CAPITOLO 1

- Cos'è un sistema operativo?
  Un sistema operativo è un insieme di programmi (software) che gestisce gli elementi fisici di un calcolatore (hardware): fornisce una piattaforma ai programmi applicativi e agiscce da intermediario fra l'utente e la struttura fisica del calcolatore.

  L'**hardware**, composto dalla CPU, dalla memoria e dai dispositivi I/O, fornisce al sistema le risorse elaborative fondamentali.
  I **programmi applicativi** definiscono il modo in cui si usano queste risorse per la risoluzione dei problemi computazionali degli utenti.
  Il **sistema operativo** controlla l'hardware e ne coordina l'utilizzo da parte dei programmi applicativi per gli utenti. Gestisce e assegna le risorse, controlla l'esecuzione dei programmi utenti e le operazioni dei dispositivi I/O.
  Il **nucleo (kernel)** è il solo programma che funziona sempre nel calcolatore (tutti gli altri sono programmi d'applicazione).
  (Poi ci sono i vari sistemi operativi che potrebbero essere inutili, quindi al massimo leggi dalle slide)

CAPITOLO 2

Un moderno calcolatore è composto da una o più CPU e da un certo numero di controller di dispositivi connessi attraverso un **bus** che permette l'accesso alla memoria condivisa del sistema.

Ciascuno di questi controller si occupa di un particolare tipo di dispositivo fisico. CPU e controller possono operare in modo **concorrente**, cioè contendendosi i cicli di accesso in memoria.

La sincronizzazione degli accessi in memoria è garantita dalla presenza di un controller di memoria. Ciascun controller ha un **buffer locale**.

La CPU sposta i dati da/verso la memoria principale da/verso i buffer locali.

Il controller informa la CPU di aver terminato un'operazione tramite **interrupt**. Questo deve causare il trasferimento del controllo all'appropriata procedura di servizio dell'evento a esso associato. Si deve anche salvare l'indirizzo dell'istruzione interrotta

Un **segnale d'eccezione (trap)** può essere causato da un programma in esecuzione a seguito di un evento eccezionale oppure da una richiesta specifica effettuata da un programma utente.

Un moderno SO è detto **interrupt driven**.

L'SO memorizza l'indirizzo di ritorno nello stack di sistema.

Determina quale tipo di interruzione si è verificato : **polling** oppure **vectored interrupt system**.

Il polling è un'interrogazione ciclica dei dispositivi per sapere se bisogna effettuare un'operazione input o output di dati. E' semplice ma spreca tanta risorsa. Bisogna usarlo solo se abbiamo un'alta situazione di traffico.

Nel vectored interrupt si effettua l'interruzione solo quando se ne ha bisogni.

Una volta iniziata l'operazione di I/O, si restituisce il controllo al processo utente solo dopo il completamento dell'operazione I/O.

L'istruzione **wait** sospende la CPU fino al successivo interrupt.

La **system call** è la richiesta al SO di consentire al programma utente di attendere il completamento dell'operazione.

STRUTTURA DELLA MEMORIA

- Memoria centrale: dispositivo di memoria direttamente accessibile dalla CPU
- Memoria secondaria: estensione della memoria centrale capace di conservare permanentemente grandi quantità di informazioni

Cache: copia temporanea di informazioni in un'unità più veloce. Si registrano i dati ai quali si è avuto accesso recentemente (e che si prevede che serviranno presto ancora)

Gerarchia memoria:Registri>cache>memoria centrale>disco Ram>dischi magnetici>dischi ottici>nastri magnetici

Uno stesso dato può esistere in vari punti nella memoria.

ARCHITETTURE DI PROTEZIONE

- **Dual mode:** La condivisione delle risorse di sistema rende necessario che l'SO garantisca che un programma malfunzionante non causi una scorretta esecuzione di altri programmi. Quindi con questa architettura viene consentito di gestire almeno due modi di funzionamento
    1. **User mode:** l'istruzione corrente si esegue per conto di un'utente.
    2. **Monitore mode:** l'istruzione corrente si esegue per conto dell'SO

  Si entra in una di queste modalità tramite il **bit di modo**: 0 (monitor) 1(utente).

- **Protezione I/O**: Tutte le istruzioni I/O sono istruzioni privilegiate. Per farlo è necessario evitare che l'utente possa ottenere il controllo del calcolatore quando questo si trova nel modo di sistema.

- **Protezione della memoria**: E' necessario fornire protezione della memoria almeno per il vettore degli interrupt e le relative procedure di servizio.
  Questo tipo di protezione si realizza impiegando due registri che contengono l'intervallo degli indirizzi validi cui un programma può accedere:
  1. **Registro di base:** contiene il più basso indirizzo della memoria fisica al quale il programma dovrebbe accedere
  2. **Registro di limite:** contiene la dimensione dell'intervallo

  Le aree di memoria al di fuori dell'intervallo stabilito sono protette.

- **Protezione CPU:** Un **timer** invia un interrupt alla CPU a intervalli di tempo specificati per assicurare che l'SO mantenga il controllo dell'elaborazione.

CAPITOLO 3

GESTIONE DEI PROCESSI

Un processo è un programma in esecuzione. Un processo necessità di alcune risorse tra cui tempo di CPU, memoria, file e dispositivi di I/O.

L'SO è responsabile delle seguenti attività di gestione dei processi:

- Creazione e cancellazione dei processi utenti e di sistema
- Sospensione e ripristino dei processi
- Fornitura di meccanismi per: sincronizzazzione e comunicazione dei processi.

GESTIONE DELLA MEMORIA CENTRALE

La memoria centrale è un vasto vettore di dimensioni che varianto tra le centinaia di migliaia e i miliardi di parole, ciascuna delle quali è dotata del proprio **indirizzo**.

La memoria centrale è **volatile** e perde le informazioni in caso di guasto del sistema.

Il SO è responsabile di: tenere traccia di quali parti della memoria sono attualmente usate e da che cosa, decidere quali processi si debbano caricare, assegnare e revocare lo spazio di memoria secondo le necessità.

GESTIONE DELLA MEMORIA SECONDARIA

Il calcolatore deve disporre di una memoria secondaria a sostegno della centrale. La maggior parte dei calcolatori impiega dischi come mezzo di memorizzazione secondaria sia per i programmi sia per i dati.

Il SO è responsabile di gestire lo spazio libero, assegnarlo, scheduling del disco.

INTERPRETE DI COMANDI

Molti comandi si impartiscono al SO attraverso **istruzioni di controllo** che riguardano: creazione e gestione dei processi, I/O, gestione della memoria secondaria, della centrale, accesso al file system, protezione e reti. Il programma che legge e interpreta le istruzioni di controllo ha diversi nomi, tra cui **shell**. La sua funzione consiste nel prelevare ed eseguire la successiva istruzione di comando.

CHIAMATE DEL SISTEMA

Le chiamate del sistema costituiscono l'interfaccia tra un processo e il sistema operativo. Generalmente disponibili in assembly. Per passare parametri al SO si usano: registri, memorizzare i parametri in un blocco di memoria e passare l'indirizzo di questo in forma di parametro in un registro, pushare i parametri in una pila da cui possono essere anche poppati dall'SO.

Tipi di chiamate del sistema:

- Controllo dei processi;
- Gestione dei file;
- Gestione dei dispositivi;
- Gestione delle informazioni;
- Comunicazione

PROGRAMMI DI SISTEMA

Offrono un ambiente conveniente per lo sviluppo e l'esecuzione dei programmi; in generale si possono classificare in: gestione dei file, informazioni di stato, modifica dei file ecc. Per la maggior parte degli utenti, l'interfaccia col SO è definita dai programmi di sistema piuttosto che dalle effettive chiamate di sistema.

METODO STRATIFICATO

Con questo metodo si suddivide il sistema operativo in un certo numero di strati. Lo strato più basso è lo strato fisico, quello più alto è l'interfaccia d'utente. Ogni strato si realizza impiegando unicamente le operazioni messe a disposizione dagli strati inferiori.

MICROKERNEL

Si progetta il SO rimuovendo dal nucleo tutti i componenti non essenziali, realizzandoli come programmi del livello d'utente e di sistema. La comunicazione si realizza secondo il modello a scambio di messaggi.

MACCHINE VIRTUALI

Il calcolatore fisico condivide le risorse in modo da creare macchine virtuali. Il SO crea l'illusione che un processo disponga della propria CPU con la propria memoria (virtuale)

CAPITOLO 4 PROCESSI

Un SO esegue differenti programmi: Un **batch system** esegue **lavori**, un **time-shared system** esegue **programmi utenti** o **task**.

Si ricorda che il **processo è un programma in esecuzione**. Comprende un program counter, una stack e una data section. Mentre un processo è in esecuzione è soggetto a **cambiamenti di stato**:

- **NUOVO:** si crea il processo
- **ESECUZIONE**: le istruzioni vengono eseguite
- **ATTESA**: il processo attende che si verifichi un qualche evento
- **PRONTO**: il processo attende di essere assegnato a un'unità di elaborazione
- **TERMINATO**: il processo ha terminato l'esecuzione

Il **Blocco di controllo di un processo (PCB)** contiene tutte le informazioni di un processo: stato, program counter, registri di cpu, informazioni sullo scheduling, informazioni sulla gestione della memoria, informazioni su risorse, informazioni su stato I/O

CODE DI SCHEDULING DEI PROCESSI

- **Coda di processi (job queue)**: insieme di tutti i processi del sistema
- **Coda di processi pronti (ready queue)**: processi presenti nella memoria centrale, pronti e in attesa di essere eseguiti
- **Coda del dispositivo (device queue)**: elenco dei processi che attendono la disponibilità di un particolare dispositivo di I/O

SCHEDULER

Lo **scheduler a lungo termine** seleziona i processi che devono essere caricati nella coda dei processi pronti. Lo **scheduler di CPU** fa la selezione tra i lavori pronti per essere eseguiti e assegna la CPU a uno di essi.

La **latenza di dispatch** è il tempo impiegato dal dispatcher per sospendere un processo e avviare l'esecuzione di uno nuovo.

Chiaramente lo scheduler di CPU viene invocato frequentemente quindi deve essere veloce, mentre lo scheduler a lungo termine viene invocato meno frequentemente quindi può anche essere "lento".

I processi possono essere caratterizzati come:

- **Processo con prevalenza di I/O:** impiega la maggior parte del proprio tempo nell'esecuzione di operazioni di I/O
- **Processo con prevalenza d'elaborazione**: richiede poche operazioni di I/O e impiega la maggior parte del proprio tempo nelle elaborazioni.

CAMBIO DI CONTESTO

Passaggio della CPU a un nuovo processo, con conseguente registrazione dello stato del processo vecchio e il caricamento dello stato precedentemente registrato del nuovo processo. Il tempo necessario al cambio di contesto è puro sovraccarico: il sistema non compie alcun lavoro utile durante la commutazione.

CREAZIONE DI UN PROCESSO

Un processo genitore può creare numerosi processi figli che a loro volta possono creare altri processi creando così un **albero di processi**.

TERMINAZIONE DI UN PROCESSO

Un processo termina quando finisce l'esecuzione della sua ultima istruzione e chiede al sistema operativo di essere cancellato usando la chiamata del sistema **exit**. Il processo figlio può riportare alcuni dati al processo genitore attraverso la chiamata del sistema **wait**. Tutte le risorse del processo sono liberate dal SO.

Un processo genitore può porre termine all'esecuzione di uno dei suoi processi figli (**abort**) perché: il processo figlio ha ecceduto all'uso di alcune risorse, il compito assegnato al processo figlio non è più richiesto, il processo genitore termina.

Il SO non consente a un processo figlio di continuare l'esecuzione in tale circostanza. Terminazione a cascata.

PROCESSI COOPERANTI

Un processo è **indipendente** se non può influire su altri processi nel sistema o subirne l'influsso.

Un processo è **cooperante** se influenza o può essere influenzato da altri processi in esecuzione nel sistema.

COMUNICAZIONE TRA PROCESSI (IPC)

Attraverso le funzioni di IPC i processi possono comunicare tra loro e sincronizzare le proprie azioni. Un sistema di scambio di messaggi permette ai processi di comunicare tra loro senza ricorrere a dati condivisi. Un sistema di IPC fornisce le due operazioni: **send(messaggio)** e **receive(messaggio)**.

Se i processi P e Q vogliono comunicare devono: stabilire un canale di comunicazione tra loro, scambiare messaggi mediante send/receive.

COMUNICAZIONE DIRETTA

I processi devono nominarsi reciprocamente in modo esplicito. I canali di comunicazione vengono stabiliti automaticamente, un canale è associato esattamente a una coppia di processi, esiste esattamente un canale tra ciascuna coppia di processi, il canale può essere unidirezionale ma è in genere bidirezionale.

COMUNICAZIONE INDIRETTA

I messaggi vengono inviati a delle **porte** che li ricevono, ciascuna è identificata in modo univoco, due processo possono comunicare solo se condividono una porta. Tra una coppia di processi si stabilisce un canale solo ese entrambi i processi condividono una stessa porta, un canale può essere associato a più di due processi, ciascuna coppia di processi può condividere più canali di comunicazione, il canale può essere unidirezionale o bidirezionale.

SINCRONIZZAZIONE

Lo scambio di messaggi può essere sincrono (bloccante) o asincrono (non bloccante).

CODE DI MESSAGGI (BUFFERING)

I messaggi scambiati tra processi comunicanti risiedono in code temporanee. Fondamentalmente esistono tre modi per realizzare queste code:

1. **Capacità zero:** 0 messaggi, il trasmittente deve fermarsi finchè il ricevente non prende in consegna il messaggio;
2. **Capacità limitata**: la coda ha lunghezza finita n, il trasmittente deve attendere se il canale è pieno
3. **Capacità illimitata**: la coda ha lunghezza potenzialmente infinita, il trasmittente non si ferma mai

SOCKETS

Una socket è definita come l'estremità di un canale di comunicazione. Ogni socket è identificata da un indirizzo IP concatenato a un numero di porta. 161.24.19.8:1627 si riferisce alla porta 1627 sul calzolatore 161.24.19.8. La comunicazione avviene attraverso una coppia di socket.

CHIAMATE DI PROCEDURE REMOTE(RPC)

La RPC è stata progettata per astrarre il meccanismo della chiamata di procedura affinchè si possa usare tra sistemi collegati tramite una rete.

Stub è un segmento di codice, questo individua la porta del server e struttura i parametri.

INVOCAZIONE DI METODI REMOTI

Una RMI consente a un thread di invocare un metodo di un oggetto remoto.

Un **thread** è un singolo flusso sequenziale di controllo all'interno di un processo. Un processo può contenere più thread. Tutti i thread di un processo condividono lo stesso spazio di indirizzamento. E' detto anche **processo leggero**.

SISTEMA PARALLELO

Architettura in cui sono presenti più CPU sulle quali sono in esecuzione processi e thread. In ogni istante di tempo posso avere più di un processo o più di un thread fisicamente in esecuzione.

SISTEMA MONOPROCESSORE TIMESLICED

Solo una CPU. Il parallelismo di processi e thread viene simulato (concorrenza) allocando a ciascuno una frazione di tempo di CPU. Allo scadere di ogni tempo il SO opera un cambio di contesto.

SISTEMA NON PREEMPTIVE

Il cambio di contesto avviene quando il processo o il thread interrompe la propria esecuzione o volontariamente o perché in attesa di un evento

SISTEMA PREEMPTIVE

Allo scadere del time slice il processo o il thread viene forzatamente interrotto e viene operato il cambio di contesto.

E' spesso necessario dividere il programma in **sottocompiti indipendenti**. Un programma può avere diverse funzioni concorrenti: operazioni ripetute nel tempo ad intervalli regolari, esecuzione di compiti laboriosi, attesa di messaggi da un altro programma ecc. L'alternativa al multithreading è il polling.

Un thread è un singolo flusso di controllo sequenziale all'interno del programma, ha un inizio, una fine, una sequenza ed ad ogni istante un solo punto di esecuzione.

Vantaggi del thread è il tempo di risposta, essendo molto leggero. Anche la condivisione di risorse, sempre essendo leggero ha bisogno di comunicare poche risorse.Maggiore economia per lo stesso motivo di prima. Infine i thread possono essere eseguiti in parallelo.

Thread possono essere definiti a **livello utente** e **livello nucleo**. Quelli a livello utente sono realizzati tramite una **libreria di funzioni** (che saranno sottoposti a scheduling) senza alcun intervento del nucleo. I thread a livello nucleo sono appunto gestiti direttamente dal kernel. La differenza tra i due è che quelli a livello utente sono più semplici perché appunto gestiti dall'utente, mentre quelli a livello nucleo sono più complesse.

Nella programmazione multithread bisogna tenere conto a vari modelli:

- **Modello da molti a uno**: Fa corrispondere molti thread al livello d'utente a un singolo thread al livello del nucleo.
- **Modello da uno a uno**: Mette in corrispondenza ciascun thread del livello d'utente con un thread del livello nucleo
- **Modello da molti a molti**: Mette in corrispondenza più thread del livello d'utente con un numero minore o uguale di thread del livello del nucleo. Consente ai programmatori di creare liberamente i thread che ritengono necessari

PTHREADS: Non si tratta di un diverso tipo di thread, ma si definisce un comportamento dei thread secondo lo standard UNIX.

CHIAMATA DI SISTEMA FORK ED EXEC

In un programma multithread la semantica di queste due chiamate di sistema cambia se un thread in un programma invoca la chiamata di sistema fork(), il nuovo processo potrebbe, in generale, contenere un duplicato di tutti i thread oppure del solo thread invocante.

CANCELLAZIONE

Permette di terminare un thread prima che completi il suo compito, per es la terminazione di un web browser comporta la terminazione dei caricamenti di pagina.

GESTIONE DEI SEGNALI

Riguarda la gestione e coordinazione delle occorrenze di eventi che avvengono nel sistema

GRUPPI DI THREAD

Un numero illimitato di thread potrebbe esaurire le risorse del sistema come la CPU o la memoria. Si cerca allora di creare dei gruppi di thread il cui scopo è quello di snellirne la gestione

CAPITOLO 6 SCHEDULING CPU

L'obiettivo della multiprogrammazione è quello di avere sempre processi in esecuzione al fine di massimizzare l'utilizzo della CPU. L'esecuzione del processo consiste in un **ciclo di elaborazione (svolta dalla CPU) e d'attesa del completamento delle operazioni di I/O**. I processi si **alternano tra questi due stati**.

L'esecuzione del processo comincia con una sequenza di operazioni d'elaborazione svolte dalla CPU (**CPU burst**) seguita da una sequenza di operazioni di I/O (**I/O burst)** e così via alternandosi.

Ogniqualvolta la CPU passa nello stato d'inattività, il SO sceglie per l'esecuzione uno dei processi presenti nella coda dei processi pronti.

Le decisioni riguardanti lo scheduling della CPU si possono prendere nelle seguenti circostanze:

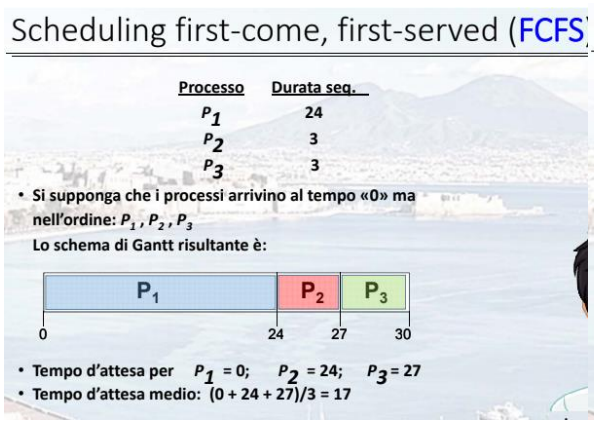
1. Un processo passa dallo stato di esecuzione a quello di attesa
2. Un processo passa dallo stato di esecuzione allo stato pronto
3. Un processo passa dallo stato di attesa allo stato pronto
4. Un processo termina

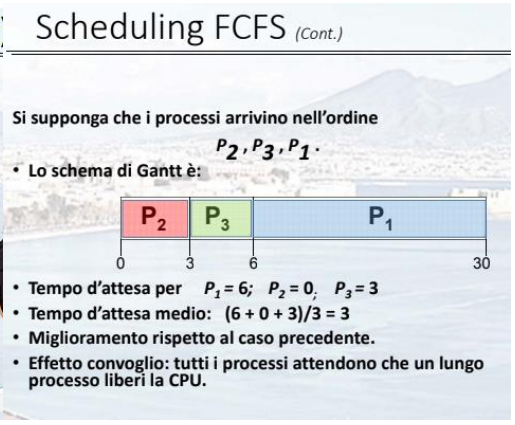
Nei casi 1 e 4 lo schema di scheduling è senza diritto di prelazione, gli altri casi è con diritto di prelazione.

Il **dispatcher** è il modulo che passa effettivamente il controllo della CPU ai processi scelti dallo scheduler a breve termine. Questa funzione riguarda: il cambio di contesto, il passaggio al modo d'utente, il salto alla giusta posizione del programma d'utente per riavviarne l'esecuzione.

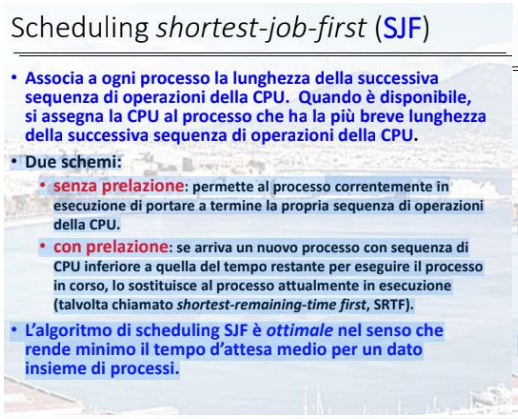
Latenza di dispatch: tempo richiesto dal dispatcher per fermare un processo e avviare l'esecuzione di un altro

First come, first served





Shortest job first



Scheduling per priorità

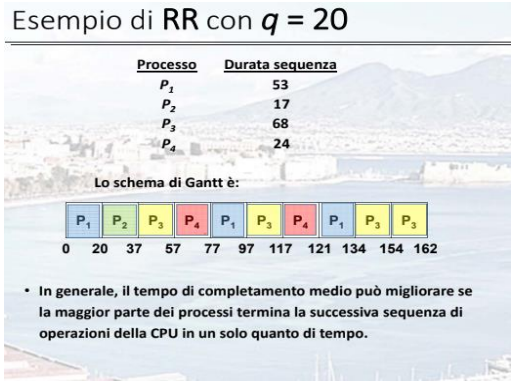
Si assegna a ciascun processo un numero di priorità, quindi si assegna la CPU al processo con priorità più elevata (numero più piccolo==priorità più elevata).

L'algoritmo SJF è un caso particolare del più generale algoritmo di scheduling per priorità, dove la la priorità è l'inverso della lunghezza della successiva sequenza di operazioni della CPU.

- Problema= **attesa indefinita (starvation)**: processi con bassa priorità possono rimanere nell'attesa indefinita della CPU
- Soluzione=**invecchiamento (aging)**: aumento graduale della priorità dei processi che attendono nel sistema da parecchio tempo

Scheduling circolare (round robin)

Ciascun processo riceve una piccola quantità fissata del tempo della CPU, chiamata **quanto di tempo**, che varia generalmente da 10 a 100 ms, e la coda dei processi pronti è trattata come una coda circolare.



Scheduling a code multiple

Una distinzione diffusa è quella che si fa tra i processi che si eseguono: In primo piano (foreground), o interattivi in sottofondo (background), o a lotti (batch)

Questi due tipi di processi possono avere diverse necessità di scheduling: in primo piano :RR, in sottofondo FCFS;

E' necessario avere uno scheduling tra le code:

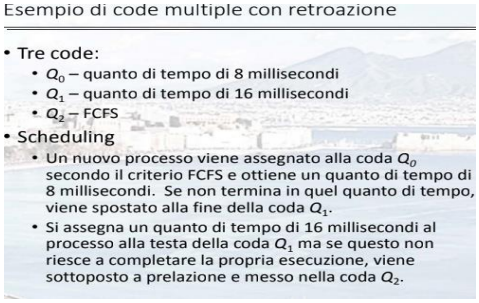
- Scheduling con priorità fissa e prelazione. Possibilità di starvation
- Possibilità di impostare i quanti di tempo.

Scheduling a code multiple con retroazione

Permette ai processi di spostarsi fra le code. In questo modo si attua una forma di aging che impedisce il verificarsi di un'attesa indefinita.

E' caratterizzato dai seguenti parametri:

- Numero di code
- Algoritmo di scheduling per ciascuna coda
- Metodo usato per determinare quando spostare un processo in una coda con priorità maggiore
- Metodo usato per determinare quando spostare un processo in una coda con priorità minore
- Metodo usato per determinare in quale coda si deve mettere un processo quando richiede un servizio



Scheduling per sistemi con più unità di elaborazione

Lo scheduling della CPU diventa più complesso nei sistemi con più unità di elaborazione

Sistemi omogenei: Sistemi nei quali le unità d'elaborazione sono, in relazione alle loro funzioni, identiche.

Condivisione del carico (load sharing)

Multielaborazione asimmetrica: tutte le attività del sistema sono gestite da un'unica CPU che quindi assume il ruolo di unità centrale, le altre unità eseguono soltanto il codice utente.

Sistemi in tempo reale stretto: sistemi capaci di garantire il completamento delle funzioni critiche entro un tempo definito

Sistemi in tempo reale debole: sistemi nei quali i processi critici hanno una priorità maggiore dei processi ordinari.

CAPITOLO 7 SINCRONIZZAZIONE

Negli SO multiprogrammati diversi processi o thread sono in **esecuzione asincrona** e possono condividere dati. I processi cooperanti possono: condividere uno spazio logico di indirizzi, oppure solo dati.

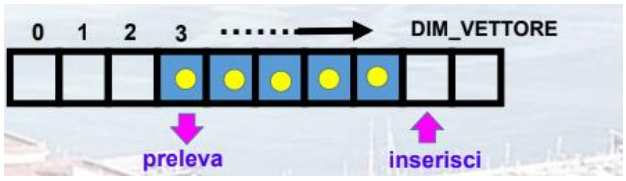
L'accesso concorrente a dati condivisi, se non si adottano politiche di sincronizzazione, può causare incoerenza degli stessi dati. Mantenere la coerenza dei dati richiede meccanismi atti ad assicurare un'ordinata esecuzione dei processi cooperanti.

Ipotizziamo di usare una zona di memoria condivisa ed usiamo un vettore circolare di grandezza DIM_VETTORE e che fa uso di 2 indici per accedere al vettore. Con l'indice **inserisci** puntiamo alla prima casella disponibile del vettore. Con l'indice **preleva** puntiamo alla prima posizione occupata del vettore. Al momento iniziale poniamo a zero sia inserisci che preleva. Da tali def si ricava che :

se inserisci==preleva ALLORA il vettore è vuoto

se ((inserisci +1) % DIM_VETTORE)==preleva ALLORA vettore pieno

Max elementi nel vettore = DIM_VETTORE=1



Per come abbiamo settato questo vettore, non è possibile riempirlo al massimo della capienza, perché rimarrà una locazione vuota. Al massimo noi possiamo utilizzare n-1 elementi.

Supponiamo di voler impiegare tutto il vettore, allora dobbiamo modificare il codice mettendo una variabile **counter**, inizializzata a 0, man mano che vengono prodotti gli elementi, il contatore viene aumentato, mentre man mano che vengono prelevati gli elementi, il contatore viene decrementato.

(NON SO SE SERVA QUINDI AL MASSIMO SLIDE 200)

Race condition: situazione in cui più processi accedono e modificano gli stessi dati in modo concorrente e i risultati dipendono dall'ordine degli accessi. Per prevenire questa situazione, i processi concorrenti devono essere sincronizzati.

Problema della sezione critica

Si considerino **n** processi concorrenti. Ciascun processo ha un segmento di codice chiamato **sezione critica** nel quale il processo può modificare variabili comuni.

Problema: Quando un processo è in esecuzione nella propria sezione critica, non si deve consentire a nessun altro processo di essere in esecuzione nella propria sezione critica.

Nella gestione della sezione critica, le garanzie principali da fornire sono tre: mutua esclusione, progresso, attesa limitata.

Mutua esclusione: Se il processo Pi è in esecuzione nella sua sezione critica, nessun altro processo può essere in esecuzione nella propria sezione critica

Progresso: se nessun processo è in esecuzione nella sua sezione critica, solo i processi che desiderano entrare nella propria sezione critica possono partecipare alla decisione su chi sarà il prossimo ad entrarci (tal decisione non può essere ritardata indefinitivamente).

Attesa Limitata: se un processo ha già chiesto l'ingresso nella sua sezione critica esiste un limite al numero di volte che si consente ad altri processi di entrare nelle rispettive sezioni critiche prima che si accordi la richiesta del primo processo: Si suppone che ogni processo sia eseguito a una velocità diverso da zero, Non si può fare alcuna ipotesi sulla velocità relativa degli n processi;

ARCHITETTURE DI SINCRONIZZAZIONE

Controlla e modifica il contenuto di una parola di memoria in modo atomico.

```
Boolean TestAndSet(boolean &obiettivo){
        Boolean valore=obiettivo;
        obiettivo=true;
        return valore;
}
```

L'istruzione swap agisce sul contenuto di due parole di memoria; come per TestAndSet è anch'essa eseguita in modo atomico

SEMAFORI

Strumento di sincronizzazione che non richiede attesa attiva. Un semaforo S è una variabile intera. A questa si può accedere solo tramite 2 operazioni atomiche predefinite:

Wait(S) ->while S<=0 do no-op; S--

Signal(S)-> S++

Un semaforo si definisce come una struttura in c. Due semplici operazioni: block sospende il processo che lo invoca, wakeup(P) riprende l'esecuzione di un processo P bloccato.

Uno stallo (deadlock) è una situazione in cui 2 o più processi attendono indefinitivamente un processo che può essere causato solo da uno dei processi in attesa.

Siano quindi S e Q 2 semafori inizializzati a 1 -> starvation->situazione d'attesa indefinita nella coda di un semaforo.

Starvation: situazione d'attesa indefinita nella coda di un semaforo.

Esistono:

Semaforo contatore: il suo valore interno può variare in un dominio logicamente non limitato.

Semaforo binario: il suo valore interno può essere soltanto 0 o 1; può essere più semplice da realizzare

PROBLEMI TIPICI DEI SINCRONIZZATORI

Problema dei produttori e dei consumatori con memoria limitata;

Problema dei lettori e degli scrittori;

Problema dei 5 filosofi;

Produttori e consumatori: Ci sono due processi, uno produttore e uno consumatore che condividono lo stesso buffer di dimensione fissata. Compito del produttore è produrre e depositare nel buffer continuamente, mentre il consumatore utilizzerà le cose prodotte, rimuovendoli dal buffer. Il problema è assicurare che il produttore non elabori nuovi dati se il buffer è pieno, e il consumatore non usi dati se il buffer è vuoto. La soluzione per il produttore è sospendere la propria esecuzione se il buffer è pieno; non appena il consumatore avrà prelevato un elemento dal buffer esso sveglierà il produttore. Per il consumatore intuibile…

Problema cinque filosofi: Cinque filosofi siedono ad una tavola rotonda con un piatto di riso al centro, cinque sedie cinque piatti, ogni filosofo ha una bacchetta a sinistra e una a destra. Il problema è che per cinque filosofi ci sono cinque bacchette. Chiaramente le risorse sono condivise, e ogni filosofo ha bisogno di due bacchette. Le possibile soluzioni:

Solo 4 filosofi possono stare contemporaneamente a tavola

Un filosofo può prendere le sue bacchette solo se sono entrambe disponibili (sezione critica)

Si adotta una soluzione asimmetrica: un filosofo dispari prende prima la bacchetta di sinistra e poi quella di destra; invece un filosofo pari prende prima la bacchetta di destra e poi quella di sinistra.

REGIONI CRITICHE

Costrutto di sincronizzazione ad alto livello, una variabile condivisa v di tipo T è dichiarata come

v: shared T

A questa variabile si può accedere solo dall'interno di un'istruzione:

region v when B do S

dove B è un espressione booleana.

Quando un processo vuole accedere alla variabile condivisa v nella regione critica, si valuta B, se true allora si esegue S, altrimenti il processo rilascia la mutua esclusione ed è ritardato fino a che B diventa vera e nessun altro processo si trova nella regione associata a v

CAPITOLO 8 STALLO DEI PROCESSI

Un insieme di processi bloccati, in cui ciascun processo detiene una risorsa e attende di accederne ad un'altra (questa in possesso ad un altro processo).

Questa problematica può essere risolta con il metodo del **rollback** (prelazione di risorse e ristabilimento di uno stato sicuro).

Si può avere una situazione di stallo solo se si verificano contemporaneamente le seguenti quattro condizioni: **mutua esclusione** cioè solo un processo alla volta può utilizzare una risorsa, **possesso e attesa** cioè un processo in possesso di almeno una risorsa attende di acquisire risorse già in possesso di altri processi, **impossibilità di prelazione** cioè una risorsa può essere rilasciata dal processo che la possiede solo volontariamente, dopo aver terminato il proprio compito, **attesa circolare** cioè deve esistere un insieme di processi tale che il primo attende una risorsa posseduta dal secondo, il secondo attende una risorsa posseduta dal terzo ecc.

Grafo di assegnazione delle risorse

Un insieme di vertici V e di archi E

Tipi di vertici

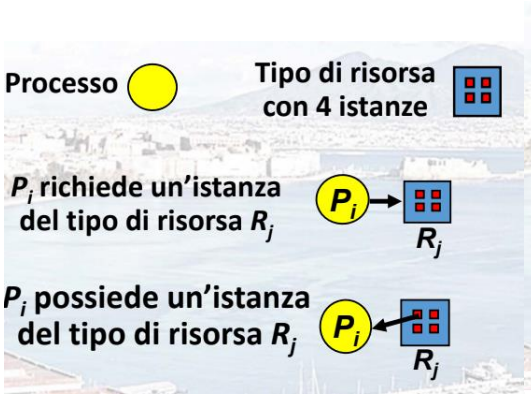
P={P1, P2,…, Pn}= tutti i processo

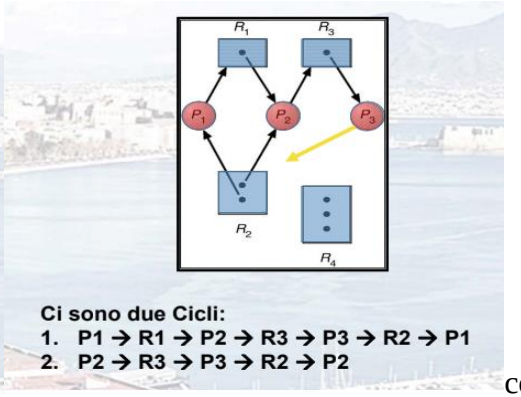
R={R1, R2,.., Rm}= tutti i tipi di risorsa

Tipi di archi

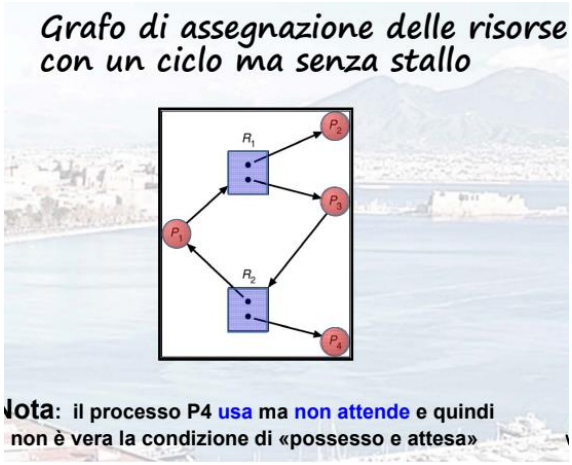
Arco di richiesta: arco orientato P1->Rj

Arco di assegnazione: arco orientato Rj->Pi





con stallo



Osservazioni se il grafo NON contiene cicli: non si verificano situazioni di stallo. Se il grafo contiene un ciclo: se c'è solo un'istanza per tipo di risorsa, allora si verifica una situazione di stallo, se vi sono più istanze per tipo di risorsa, allora c'è la possibilità di stallo.

METODI PER LA GESTIONE DELLE SITUAZIONI DI STALLO

- Assicurare che il sistema non entri mai in stallo
- Consentire al sistema di entrare in stallo, individuarlo e quindi eseguire il ripristino
- Ignorare il problema

Un metodo alternativo per evitarle sarebbe richiedere ulteriori informazioni sui modi delle richieste delle risorse. Il modello più utile e semplice richiede che ciascun processo dichiari il

numero massimo delle risorse di ciascun tipo di cui necessita. L'algoritmo per evitare lo stallo esamina dinamicamente lo stato di assegnazione delle risorse per garantire che non possa esistere una condizione di attesa circolare. Lo stato di assegnazione delle risorse è definito dal numero di risorse disponibili e assegnare, e dalle richieste massime dei processi.

STATO SICURO

Uno stato si dice sicuro se il sistema è in grado di assegnare risorse a ciascun processo (fino al suo massimo) in un certo ordine e impedire il verificarsi di uno stallo.

Un sistema si trova in uno stato sicuro solo se esiste una sequenza sicura. (appunti ignorano)

Se un sistema è in uno stato sicuro non si verificano situazioni di stallo, altrimenti possibilità di stallo.

Evitare lo stallo-> assicurare che il sistema non si trovi mai in uno stato non sicuro

(appunti ignorano appunti ignorano appunti ignorano)

CAPITOLO 9 GESTIONE DELLA MEMORIA

Per essere eseguito un programma deve essere caricato nella memoria e inserito all'interno di un processo.

Coda d'ingresso: insieme dei processi presenti nei dischi che attendono d'essere trasferiti nella memoria per essere eseguiti.

Generalmente, l'associazione di istruzioni e dati a indirizzi di memoria si può compiere in qualsiasi fase del seguente percorso:

- **Compilazione**: se in qeusta fase si sa dove il processo risiederà nella memoria, si può generare un **absolute code**; se, in un momento successivo, la locazione iniziale cambiasse, sarebbe necessario ricompilare il codice
- **Caricamento:** Se nella fase di compilazione non è possibile sapere in che punto della memoria risiederà il processo, il compilatore deve generare **codice rilocabile**
- **Esecuzione:** Se durante l'esecuzione il processo può essere spostato da un segmento di memoria a un altro, si deve ritardare l'associazione degli inidirizzi fino alla fase di esecuzione.

Indirizzo LOGICO: indirizzo generato dalla CPU, detto anche indirizzo virtuale

Indirizzo FISICO: indirizzo visto dall'unità di memoria

I metodi di associazione degli indirizzi nelle fasi di compilazione e di caricamento producono indirizzi logici e fisici identici. Con i metodi di associazione nella fase d'esecuzione invece, gli indirizzi logici non coincidono con gli indirizzi fisici

UNITA' DI GESTIONE DELLA MEMORIA MMU: E' un dispositivo per l'associazione, nella fase di esecuzione, degli indirizzi virtuali agli indirizzi fisici. Nello schema MMU, quando un processo utente genera un indirizzo, prima dell'invio all'unità di memoria, **si somma a tale indirizzo il valore contenuto nel registro di locazione.** Il programma utente tratta con gli indirizzi logici, non con i fisici.

Caricamento dinamico: Una procedura non viene caricata fino a quando non è richiamata

Collegamento dinamico: Il collegamento viene differito fino al momento dell'esecuzione. Una piccola porzione di codice di riferimento indica come localizzare la giusta procedura di libreria residente nella memoria. Il codice (stub) sostituisce se stesso con l'indirizzo della procedura, che viene poi eseguita.

Sovrapposizione di sezioni(overlay): Mantiene nella memoria soltanto le istruzioni e i dati che si usano con maggior frequenza. Necessario per consentire a un processo di essere più grande della memoria che gli si assegna.

SWAPPING

Un processo può essere trasferito temporaneamente nella **memoria ausiliaria** e poi riportato nella memoria centrale al momento di riprenderne l'esecuzione.

Memoria ausiliaria: disco veloce sufficientemente capiente da contenere le copie di tutte le immagini di memoria di tutti i processi; deve per mettere un accesso diretto a queste immagini di memoria.

Roll OUT, Roll IN: variante impiegata per algoritmi basati sulle priorità: il processo con priorità inferiore viene scaricato dalla memoria centrale per fare spazio all'esecuzione del processo con priorità maggiore.

ASSEGNAZIONE CONTIGUA DELLA MEMORIA

La memoria centrale si divide di solito in due partizioni: una per l'SO l'altra per i processi utenti.

Lo schema a registro di rilocazione viene usato per proteggere il SO dai processi utenti e i processi utenti dagli altri processi utenti.

Il **registro di Rilocazione** contiene il valore dell'indirizzo fisico minore.

Il registro di Limite contiene l'intervallo di indirizzi logici

Assegnazione su più partizioni:

hole: blocco di memoria disponibile, varie dimensioni, sparsi

quando si carica un processo che necessità di memoria occorre cercare un buco sufficientemente grande da contenerlo

Il SO tiene traccia di: partizioni assegnate e partizioni libere (buchi)

- **First fit:** si assegna il primo buco abbastanza grande
- **Best fit:** si assegna il buco più piccolo in grado di contenere il processo, occorre compiere la ricerca su tutta la lista, sempre che non sia ordinata per dimensione.
- **Worst fit:** si assegna il buco più grande: anche in questo occorre esaminare tuta la lista

Page frame: area fisica (intervallo di locazioni contigue) di memoria RAM di dimensione fissata

FRAMMENTAZIONE

Frammentazione esterna: Con i blocchi di dimensioni variabili, dopo un certo numero di allocazione e deallocazione di processi, si formeranno un certo numero di porzioni di memoria libera (buchi) di dimensioni insufficienti a contenere un processo.

Frammentazione interna: Con i blocchi di dimensioni fisse, all'interno di ogni blocco ci sarà uno spazio non utilizzato.

Una soluzione al problema della partizione è data dalla **compattazione:** riordina il contenuto della memoria per riunire la memoria libera in un unico grosso blocco. E' possibile solo se la rilocazione è dinamica e si compie nella fase di esecuzione.

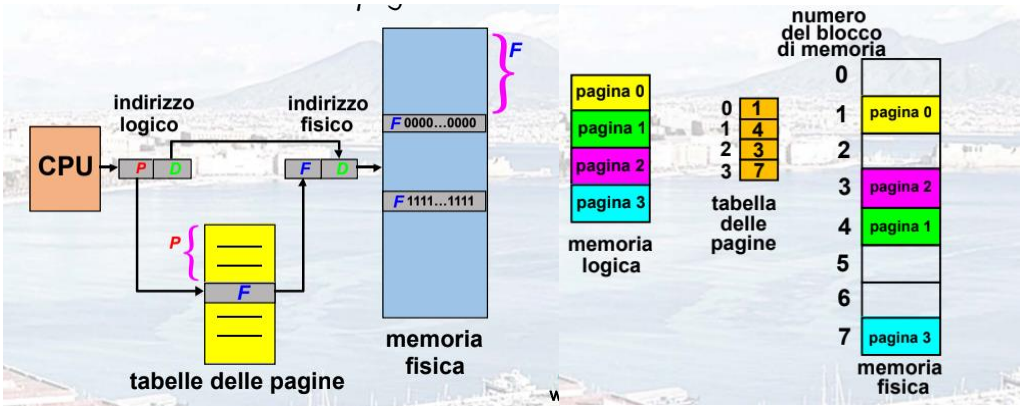
PAGINAZIONE

Metodo di gestione della memoria che permette che lo spazio degli indirizzi fisici di un processo non sia contiguo. Suddivide la memoria fisica in blocchi di memoria di dimensioni fisse (frame). Dimensione tipica pagina compresa tra 512 byte e 16 MB. Suddivide la memoria logica in blocchi di uguale dimensione detti (pages). Tiene traccia di tutti i blocchi liberi. Utilizza una tabella delle pagine per tradurre gli indirizzi logici in indirizzi fisici. **Può evitare la frammentazione esterna**.

Schema di traduzione degli indirizzi

Ogni indirizzo generato dalla CPU è diviso in:

- **Numero di pagina(p):** serve come indice per la tabella delle pagine, che contiene l'indirizzo di base nella memoria fisica di ogni pagina
- **Scostamento di pagina (d):** combinato con l'indirizzo di base definisce l'indirizzo della memoria fisica, che si invia all'unità di memoria



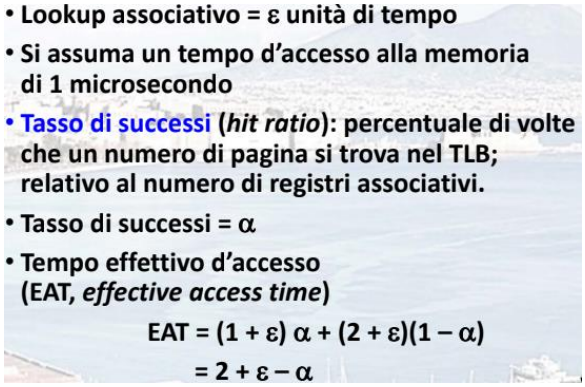
Memoria associativa TLB: Cache HW che velocizza la traduzione di indirizzi lineari in fisici.

La prima volta che un indirizzo lineare è tradotto, l'indirizzo fisico corrispondente è registrato in un elemento della TLB. Per usi successivi dello stesso indirizzo lineare si userà l'indirizzo fisico presente in TLB.

Ogni CPU ha la propria TLB-> sincronizzazione non necessaria.

Processi che eseguono su diverse CPU possono usare stessi indirizzi logici corrispondenti a indirizzi fisici diversi. TLB funziona da cache per gli elementi della Page table. Se cambia l'indirizzo fisico di una frame pagina (swapping), l'elemento della TLB corrispondente è invalidato.

Tempo effettivo d'accesso



Protezione della memoria

In un ambiente paginato, la protezione della memoria è assicurata dai bit di protezione associati a ogni blocco di memoria. Di solito si associa un **bit di validità** a ciascun elemento della tabella delle pagine: bit valido indica che la pagina corrispondente è nello spazio d'indirizzi logici del processo, quindi la pagina è valida, bit non valido il contrario.

Struttura della tabella delle pagine

- **Paginazione gerarchica:** suddivide la tabella delle pagine in parti più piccole.
- **Tabella delle pagine di tipo hash:** L'argomento della funzione di hash è numero della pagina virtuale. Ogni elemento nella tabella hash contiene una lista concatenata di elementi che la funzione hash fa corrispondere alla stessa locazione. I numeri di pagina virtuali vengono confrontati e, se coincidono, si usa l'indirizzo del relativo blocco di memoria per generare l'indirizzo fisico desiderato.
- **Tabella delle pagine invertita:** Un elemento per ogni pagina reale. Ciascun elemento è costituito dall'indirizzo virtuale della pagina memorizzata in quella reale locazione di memoria, con informazioni sul processo che possiede tale pagina. Riduce la quantità di memoria necessaria per memorizzare ogni tabella delle pagine, ma aumenta il tempo di ricerca nella tabella quando si fa rifermimento a una pagina.

SEGMENTAZIONE:schema di gestone della memoria che implementa una visione del programma per blocchi funzionali.

Un aspetto importante della gestione della memoria è quello della separazione tra la visione della memoria dell'utente e l'effettiva memoria fisica. Un programma è un insieme di segmenti. Un segmento è un'unità logico come: programma principale, procedure, funzioni, metodi ecc.

Un indirizzo logico è una coppia <numero di segmento, scostamento>

La **segment table** traduce gli indirizzi bidimensionale definiti dall'utente negli indirizzi fisici unidimensionali, ogni elemento è composto da: base che contiene l'indirizzo fisico iniziale della memoria al quale il segmento risiede e il limite che contiene la lunghezza del segmento.

Segmentazione v Paginazione

Entrambi sono ridondanti, entrambi partizionano/assegnano la memoria fisica tra processi.

La segmentazione assegna intervalli di indirizzi lineari a diversi processi. La paginazione mappa un intervallo di indirizzi lineari su intervalli di indirizzi fisici diversi.

CAPITOLO 10 MEMORIA VIRTUALE

Sistema SW/HW in grado di simulare uno spazio di memoria centrale maggiore di quello fisicamente presente. Rappresenta in modo astratto e logico la memoria centrale di un calcolatore.

La memoria virtuale rappresenta la separazione della memoria logica, vista dall'utente, dalla memoria fisica. Solo parte del programma deve essere in memoria per l'esecuzione. Lo spazio degli indirizzi logici può essere maggiore dello spazio degli indirizzi fisici.

La memoria virtuale generalmente si realizza nella forma di: demand paging o demand segmentation.

Demand paging

Non si carica mai nella memoria una pagina che non sia necessaria.

Bit di validità: a ciascun elemento della tabella delle pagine è associato un bit di validità (1-> in memoria, 0-> non in memoria). Inizialmente impostato a 0.

Eccezione di pagina mancante: Si verifica se il processo tenta di accedere a una pagina che non era stata caricata nella memoria. L'architettura di paginazione, traducendo l'indirizzo, nota che il bit non è valido e invia il segnale d'eccezione al SO. Quindi il SO deve cercare un blocco di memoria libero (se non c'è deve crearlo tipo politica cache), si trasferisce la pagina desiderata in questo blocco, si aggiornano le tabelle bit di validità =1, si riavvia l'istruzione che era stata interrotta.

Copiatura su scrittura COW: consente la condivisione iniziale delle pagine da parte dei processi genitori e dei processi figli. Se un processo scrive su una pagina condivisa, il sistema deve creare una copia di tale pagina. Maggiore efficienza perché solo le pagine modificate vengono copiate.

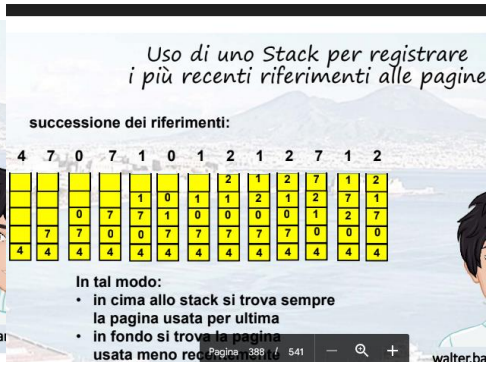
Associazione dei file alla memoria: Consiste nel permettere che una parte dello spazio degli indirizzi virtuali sia associata logicamente a un file.

Schema di base : Si individua la locazione nel disco della pagina richiesta, si cerca un blocco di memoria libero (se esiste lo si usa, altrimenti si impiega un algoritmo di sostituzione della pagine per scegliere un blocco di memoria vittima), si scrive la pagina richiesta nel blocco di memoria libero, si modificano le tabelle pagine e dei blocchi di memoria, si riavvia il processo utente.

ALGORITMI DI SOSTITUZIONE PAGINE

Algoritmo FIFO

solo uso didattico



CAPITOLO 11 INTERFACCIA DEL FILE SYSTEM

Un file è un insieme di informazioni, correlate e registrate nella memoria secondaria, cui è stato assegnato un nome.

Tipi:

Dati:numerici, alfabetici, alfanumerici, binari

Programmi

Struttura dei file

Nessuna->sequenza di parole, byte

Strutture semplici->Righe, Lunghezza fissa, lunghezza variabile

Strutture complesse->Documento formattato, Relocatable load file

Attributi dei file

Nome	Unica informazione human-readable
Tipo	Infomrazione necessaria ai sistemi che gestiscono tipi di file diversi
Locazione	Puntatore al dispositivo e alla locazione del file in tale dispositivo
Dimensione	Dimensione corrente del file
Protezione	Permessi sul file
Ora, data e identificazione dell' utente	Dati utili ai fini della protezione e del controllo di utilizzo

Le informazioni sui file sono conservate nella struttura di directory, che risiede a sua volta nella memoria secondaria.

Operazione sui file

- Creazione;
- Scrittura;
- Lettura;
- Riposizionamento;
- Cancellazione;
- Troncamento;
- Open(Fi): ricerca nella directory del disco elemento Fi e ne sposta il contenuto in memoria
- Close(Fi): rimuove il contenuto dell'elemento Fi dalla memoria alla directory del disco

Metodi di accesso

Accesso sequenziale

```
read next
write next
reset
no read aftr last write (rewrite)
```

Accesso diretto

```
read n
write n
position to n
      read next
      write next
rewrite n
```

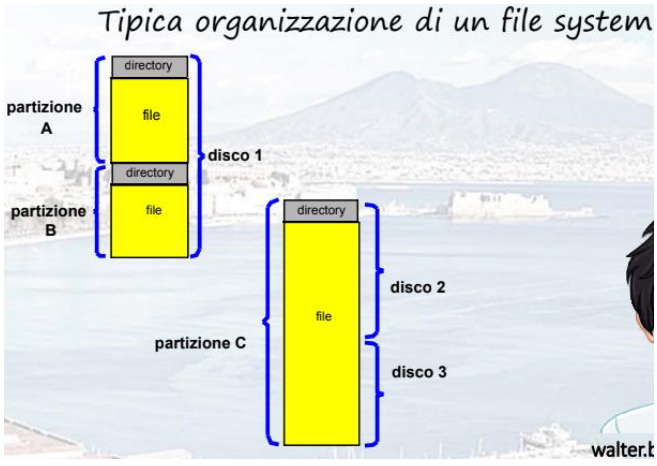
Dove $n \rightarrow$ numero relativo del blocco

21.6 Simulazione dell'accesso sequenziale a un file ad accesso diretto

Accesso sequenziale	Realizzazione nel caso di accesso diretto
reset	cp = 0;
read next	read cp; cp = cp+1;
write next	write cp; cp = cp+1;

Struttura di directory

Un insieme di nodi contenenti informazioni su tutti i file. Sia la directory sia i file risiedono sul disco. Le copie di backup di queste 2 strutture vengono in genere archiviate su nastro.



Informazioni sui file nella directory

- Nome;
- Tipo;
- Indirizzo;
- Dimensione corrente;
- Dimensione massima;
- Data dell’ultimo accesso;
- Data dell’ultimo aggiornamento;
- ID del proprietario;
- Informazioni di protezione

Operazioni eseguite su una directory

- Ricerca di un file;
- Creazione di un file;
- Cancellazione di un file;
- Elencazione di una directory;
- Ridenominazione di un file;
- Navigazione del file system

Organizzare logicamente una directory

Efficienza-> individuare rapidamente un file.

Scelta del nome

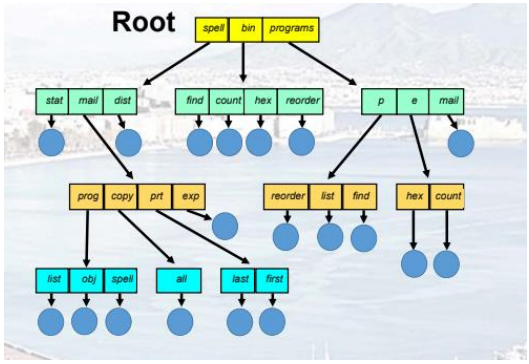
Raggruppamento-> in base alle caratteristiche del file

Directory a singolo o doppio livello

Directory a singolo livello-> una singola directory per tutti gli utenti

Directory a doppio livello-> directory separate per ciascun utente

Struttura di directory ad albero

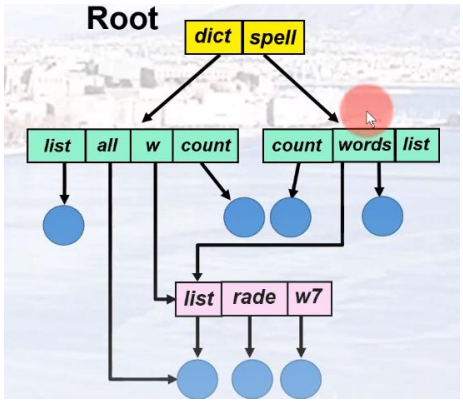


Ricerca efficiente

Possibilità di raggruppamento

Directory corrente

Struttura di directory a grafo aciclico



Consente alla directory di avere sottodirectory e file condivisi.

Nomi diversi che possono riferirsi allo stesso file

Se a ogni operazione di cancellazione segue l'immediata rimozione del file, potrebbero restare puntatori a un file che non esiste più. Per risolvere: cercare tutti questi collegamenti e rimuoverli, conservazione del file fino a che non siano stati cancellati tutti i riferimenti a esso.

Montaggio di un file system: Per essere reso accessibile un file system deve essere montato. Si fornisce al SO il nome del dispositivo e la sua locazione nella struttura di file e directory alla quale agganciare il file system.

Condivisione di file: Caratteristica desiderabile nei file di sistemi multiutente. La condivisione può avvenire attraverso uno schema di protezione. Nei sistemi distribuiti i file sono condivisi attraverso la rete. Il file system di rete NFS è uno dei metodi più comuni per la condivisione.

Protezione: Il proprietario/creatore di un file deve essere in grado di controllare che cosa può essere fatto e da chi. Tipi di accesso: lettura, scrittura, esecuzione, aggiunta, cancellazione, elencazione.

CAPITOLO 12 REALIZZAZIONE DEL FILE SYSTEM

Struttura del file:

- Unità di memorizzazione logica
- Raccolta di informazioni correlate

Il file system risiede permanentemente nella memoria secondaria. Ha una struttura stratificata.

Blocco di controllo dei file: struttura di memorizzazione che contiene informazioni sui file come la proprietà, i permessi e la posizione del contenuto.

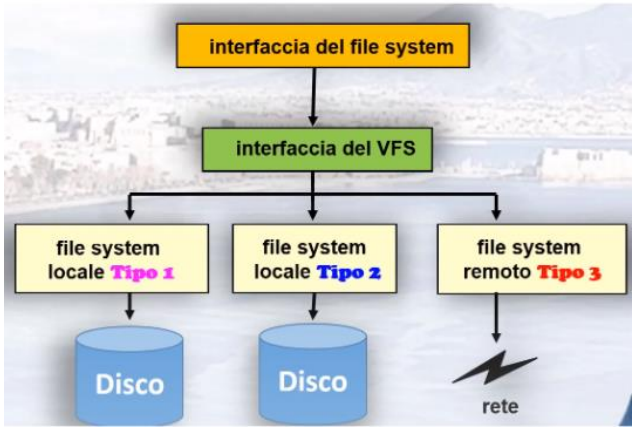
File system stratificato

Programmi d'applicazione->file system logico->modulo di organizzazione dei file-> file system di base-> controllo dell'i/o-> dispositivi

File system virtuali

Fornisce una tecnica object-oriented per la realizzazione del file system. Separa le operazioni generiche del file system dalla loro realizzazione definendo un'interfaccia VFS uniforme.

Il VFS è basato su una struttura di rappresentazione dei file detta vnode che contiene un indicatore numerico per tutta la rete di ciascun file



Realizzazione delle directory

Lista lineare: contenente i nomi dei file con puntatori ai blocchi di dati. Metodo facile di programmazione ma onerosa in terminii di tempo

Tabella di hash: lista lineare con struttura di dati hash. Diminuisce notevolmente il tempo di ricerca nella directory. Collisioni: situazioni in cui da due nomi di file si ottiene un riferimento alla stessa locazione. Dimensione fissa.

Metodi di assegnazione

Un metodo di assegnazione indica il modo in cui i blocchi di disco vengono assegnati ai file:

- Assegnazione contigua
- Assegnazione concatenata
- Assegnazione indicizzata

Assegnazione contigua

Ciascun file deve occupare un insieme di blocchi contigui nel disco. Semplice perché è richiesto solo l'indirizzo del primo blocco e la lunghezza. Impiego di spazio

Associazione concatenata

Ciascun file è composto da una lista concatenata di blocchi del disco i quali possono essere sparsi in qualsiasi punto del disco stesso. Semplice perché necessità solo dell'indirizzo di partenza- Non c'è spreco di spazio. Non vi è accesso casuale. Una variante di questa associazione consiste nell'uso della FAT.

Associazione indicizzata

Raggruppa tutti i puntatori in una sola locazione:il blocco indice. Necessità di tabella indice, accesso casuale, accesso dinamico senza frammentazione esterna. Ogni file deve avere un blocco indice, quindi è auspicabile che questo sia quanto più piccolo possibile, ma se il blocco indice è troppo piccolo non può contenere un numero di puntatori sufficiente per un file di grandi dimensioni, quindi è necessario disporre di un meccanismo per la gestione di questa situazione;

- Schema concatenato
- Indice a più livelli
- Schema combinato

Gestione dello spazio libero

Vettore di bit (n blocchi)

$$\text{bit[i]} = 1 \rightarrow \text{blocco[i] libero}$$

$$\text{bit[i]} = 0 \rightarrow \text{blocco[i] assegnato}$$

Il numero del primo blocco libero è dato dalla seguente espressione:

$$(\text{numero di bit per parola}) \times (\text{numero di parole di valore 0}) + \text{scostamento del primo bit 1}$$

Efficienza e prestazioni

L'efficienza dipende dagli algoritmi usati per l'assegnazione del disco e dalla gestione delle directory.

Prestazioni:

- Cache del disco;
- Rilascio indietro e lettura anticipata-> tecniche di ottimizzazione degli accessi sequenziali
- Per migliorare le prestazioni nei PC si riserva e si gestisce una sezione della memoria con un disco virtuale o disco ram

I/O senza una buffer cache unificata

Tecniche di memoria virtuale per la gestione dei dati dei file come pagine anziché come blocchi di file system.

Ripristino

Verifica della coerenza dei dati (si cerca di correggere ogni incoerenza).

Backup di dati come metodologia per ovviare a incoerenze di dati. In caso contrario c'è il restore.

NFS

Sistema per l'accesso a file remoti attraverso Lan. Si considera un insieme di stazioni di lavoro interconnesse come un insieme di calcolatori indipendenti con file system indipendenti. Lo scopo è quello di permettere un certo grado di condivisione tra questi file system su richiesta esplicita in modo trasparente. Uno degli scopi nella progettazione dell'NFS era quello di operare in un ambiente eterogeneo di calcolatori, sistemi operativi e architetture di rete.

Protocollo di montaggio

Stabilisce una connessione logica iniziare tra server e client. Comprende il nome della directory remota da montare e il nome del calcolatore server in cui tale directory è memorizzata. Quando il server riceve una richeista di montaggio conforme alla propria lista di esportazione, riporta al client un file handle da usare come chiave per ulteriori accessi al file. Questa file handle contiene tutte le informazioni di cui ha bisogno il server per gestire un proprio file.

Protocollo NFS

Offre un insieme di RPC per operazioni su file remoti che svolgono le seguenti operazioni:

- Ricerca di un file in una directory;
- Lettura di un insieme di elementi di una directory;
- Manipolazione di collegamenti e di directory;
- Accesso ad attributi di file;
- Lettura e scrittura di file

Assenza dell'informazione di stato-> ogni richiesta deve fornire un insieme completo di argomenti.

I dati modificati si devono riscrivere nei dischi del server prima che i risultati siano riportati al client.

Il protocollo NFS non fornisce meccanismi per il controllo della concorrenza.

Architettura NFS

Interfaccia del file system UNIX

File system virtuale-> identifica i file locali da quelli remoti e invoca l'appropriata operazione del file system.

Vantaggio-> client e server sono identici

Traduzione dei nomi di percorso

Si compie suddividendo il percorso stesso in nomi componenti ed eseguendo una chiamata lookup nell'NFS separata per ogni coppia formata da un nome componente e un vnode di directory.

Una cache per la ricerca dei nomi delle directory, nel sito del client, conserva i vnode per i nomi delle directory remote; in questo modo si accellerano i riferimenti ai file con lo stesso nome di percorso iniziale.

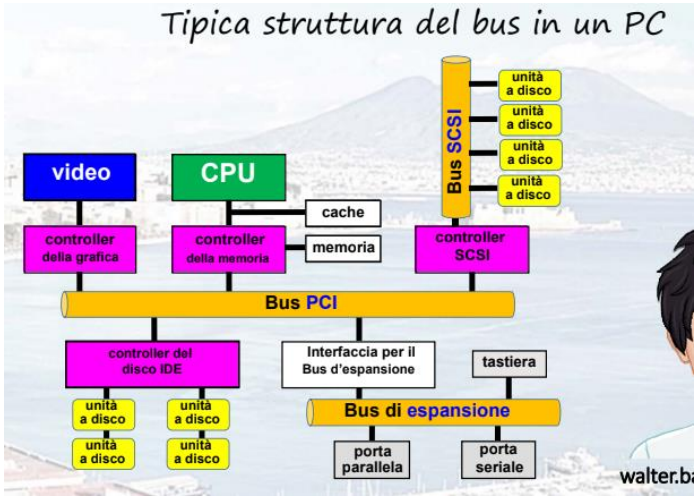
CAPITOLO 13 SISTEMI DI (D)I/O

Grande numero di tipi di dispositivi;

Concetti comuni: Porta, Bus (collegamento a margherita o accesso diretto condiviso) e controller.

Dispositivi di controllo delle istruzioni I/O.

I dispositivi hanno indirizzi usati da: speciali istruzioni di I/O, I/O associato alla memoria



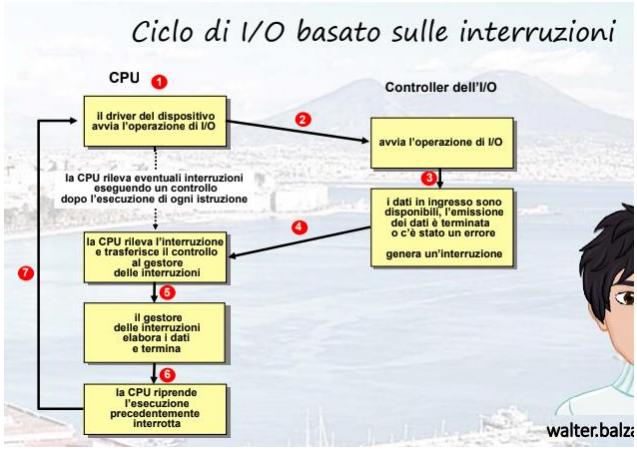
Polling

Determina lo stato del servizio: command-ready, busy, error.

Il polling è efficiente se c'è un'alta situazione di traffico, stupido usarlo se trova raramente un dispositivo pronto mentre ci sono altre elaborazioni che attendono la CPU.

Interrupt

Usati diffusamente dai SO moderni per gestire eventi asincroni e per eseguire nel modo supervisore le procedure del nucleo. I controller dei dispositivi, gli errori e le chiamate del sistema generano segnali d'interruzione al fine di innescare l'esecuzione di procedure del nucleo. Poiché le interruzioni sono usate in modo massiccio per affrontare situazioni in cui il tempo è un fattore critico, è necessario avere un'efficiente gestione delle interruzioni per ottenere buone prestazioni del sistema.



Accesso diretto alla memoria (DMA)

Usato per evitare l'I/O programmato per trasferimento di grandi quantità di dati. Richiede un controller DMA. Questo agisce direttamente sul bus della memoria, presentando al bus gli indirizzi di memoria necessari per eseguire il trasferimento senza aiuto da parte della CPU.

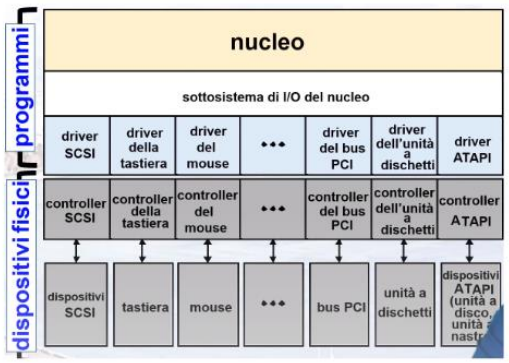
L'interfaccia di I/O per le applicazioni

Le chiamate del sistema di I/O incapsulano il comportamento dei dispositivi in alcune classi generiche. Lo scopo dello strato dei driver dei dispositivi è di nascondere al sottosistema di I/O del nucleo le differenze fra i controller dei dispositivi.

I dispositivi possono ovviamente differire in molti aspetti:

- Trasferimento a flusso di caratteri o a blocchi;
- Accesso sequenziale o diretto;
- Sincroni o asincroni;
- Condivisibili o riservati;
- Velocità di funzionamento;
- Lettura e scrittura, solo lettura o solo scrittura;

Struttura relativa all' I/O di un nucleo



Dispositivi con trasferimento a blocchi o a caratteri

Includono le unità a disco:

- Le istruzioni comprendono read, write e seek;
- I/O a basso livello (raw I/O) o accesso tramite file system;
- Possibile accesso al file associato alla memoria;

I dispositivi con trasferimento a caratteri includono tastiere, mouse, porte seriali: i comandi comprendono get, put. E' possibile costruire servizi aggiuntivi quali l'accesso riga per riga.

Dispositivi di rete

Interfaccia di rete socket-> separa il protocollo di rete dalle operazioni di rete (include la funzione select)

Gli approcci variano ampliamente.

Orologi e temporizzatori

Segnano l'ora corrente, segnalano il tempo trascorso, regolano il temporizzatore. Il dispositivo che misura la durata di un lasso di tempo e che può avviare un'operazione si chiama temporizzatore programmabile.

Ioctl tratta gli aspetti dell'I/O quali orologi e temporizzatori

I/O bloccante e non bloccante

Bloccante-> si sospende l'esecuzione dell'applicazione.

Non bloccante->sovrappone elaborazione e I/O

Chiamate del sistema asincrone -> restituiscono immediatamente il controllo al chiamante senza attendere che l'I/O sia stato completato.

Sottosistema per l'I/O del nucleo

Scheduling -> fare lo scheduling di una serie di richieste dell'I/O significa stabilirne un'ordine d'esecuzione efficace.

Memoria transitoria -> un buffer è un'area di memoria che contiene dati mentre vengono trasferiti tra 2 dispositivi o tra un'applicazione e un dispositivo:

- Necessita di gestire la differenza di velocità fra il produttore e il consumatore di un flusso di dati;
- Gestione dei dispositivi che trasferiscono dati in blocchi di dimensioni diverse;
- Realizzazione della semantica delle coppie

Cache -> regione di memoria veloce per copie di dati.

Code (spooling) -> memoria di transito contentente dati per un dispositivo che non può accettare flussi di dati intercalati.

Gestione degli errori

Un So che usi la protezione della memoria può proteggersi da molti tipi di errori dovuti ai dispostivi o alle applicazioni.

Di norma una chiamata del sistema per l'I/O riporta un bit d'informazione sullo stato d'esecuzione della chiamata.

Molti dispositivi SCSI mantengono alcune pagine di informazioni su errori avvenuti; queste pagine possono essere richieste dalla macchina (raramente).

Strutture dati del nucleo

Il nucleo ha bisogno di mantenere informazioni sullo stato dei componenti coinvolti nelle operazioni di I/O.

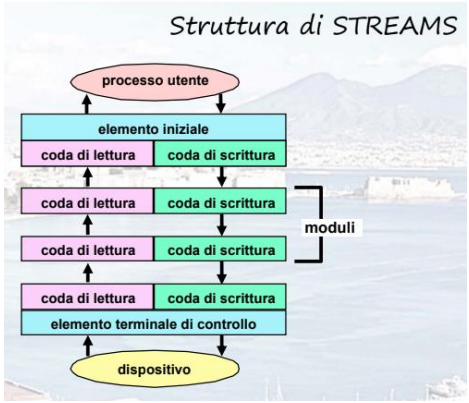
Trasformazione delle richieste di I/O in operazioni dei dispositivi

Si consideri la lettura di un file da un'unità a disco:

- Si determina il dispositivo che detiene il file
- Si traduce il nome nella rappresentazione del dispositivo
- Si trasferiscono i dati dal disco al buffer
- Si rendono disponibili i dati al processo
- Si restituisce il controllo al processo

Streams

Connessione full-duplex tra un driver dispositivo e un processo utente. Consiste di un elemento iniziale per il processo utente, un elemento terminale che controlla il dispositivo, un certo numero di moduli intermedi fra questi due estremi. Tutti questi elementi possiedono una coppia di code, una di lettura e una di scrittura. Per il trasferimento dei dati tra le due code, si usa uno schema a scambio di messaggi.



Prestazioni

L'I/O è uno tra i principali fattori che influiscono sulle prestazioni di un sistema:

- Richiede un notevole impiego della CPU per l'esecuzione del codice del driver e per uno scheduling equo ed efficiente
- I risultanti context-switch sfruttano fino in fondo la CPU e le sue memorie cache
- Copia dei dati
- Traffico di rete

Per migliorare le prestazioni si possono applicare diversi principi:

- Ridurre il numero di context switch
- Ridurre la frequenza degli interrupt
- Uso di controllori DMA intelligenti
- Equilibrare le prestazioni della CPU, del sottosistema per la gestione della memoria, del bus e dell'I/O

CAPITOLO 14 MEMORIA SECONDARIA E TERZIARIA

Struttura dei dischi

I dischi sono considerati un grande vettore monodimensionale di blocchi logici, dove un blocco logico è la minima unità di trasferimento. Il vettore monodimensionale di blocchi logici corrisponde ai settori del disco:

- Il settore 0 è il primo settore della prima traccia sul cilindro più esterno.
- La corrispondenza prosegue ordinatamente lungo la prima traccia, quindi lungo le rimanenti tracce del primo cilindro e cosi via dall'esterno verso l'interno.

HDD vs SSD

	Costo	Capacità	Delicatezza	Velocità
HDD	Basso	Alta	Alta	Bassa
SSD	Alto	Bassa	Bassa	Alta

Caratteristiche hard disk

Dispositivo meccanico, più lento dei dispositivi elettronici, di tipo CAV(fino a 10k e 15 k RPM). I tempi di accesso alle informazioni su un Hard disk dipendono fortemente da:

- Lo spostamento della testina in senso radiale fino a raggiungere la traccia desiderata (seek time)
- L'attesa che il settore desiderato si trovi a passare sotto la testina, tale tempo dipende dalla velocità di rotazione del disco (latency time)
- Il tempo di lettura vero e proprio dell'informazione.

Un hard disk è uno dei tipi di dispositivi di memoria più utilizzati. E' un dispositivo magnetico che utilizza uno o più dischi magnetizzati. L'insieme delle tracce dei dischi che compongono lo stesso HDD e che hanno lo stesso raggio formano quello che viene detto **cilindro**.

Floppy disk, CD ROM, Dischi rimovibili, Dischi WORM, Nastri

CD-ROM -> disco su cui i dati sono scritti su una lunga spirale che si propaga dall'interno verso l'esterno. Nel funzionamento il disco ruota ad una velocità lineare costante CLV

Floppy disk -> costituito da una sostanza elettromagnetica distribuita in modo uniforme su tutta la superficie del disco. Nel funzionamento il disco ruota ad una velocità angolare costante CAV. Le tracce sono aree circolari numerate da 0(traccia esterna) ad N (traccia interna)

Dischi rimovibili -> registrano i dati su un disco rigido ricoperto da materiale magnetico, non sfruttano il magnetismo ma materiali speciali che la luce laser può alterare in modo da creare punti relativamente chiari o scuri.

Dischi WORM (Write once, Read many) -> dischi che possono essere scritti una volta sola e per questo durevoli e affidabili

Nastro -> meno costoso di un disco ma presenta un'accesso più lento. Vengono utilizzati nel caso di grosse quantità di dati che non richiedano rapidi accessi diretti.

Hard dish z-cav

Considerando la forma dei cilindri si è iniziato a progettare una nuova strategia nota come Zone Bit Recording ZBR il cui obiettivo è di memorizzare più settori nelle tracce esterne.

Poiché la tipologia usuale degli Hard Disk è di tipo CAV la strategia ZBR dovrà opportunamente gestire la velocità variabile della lettura e della scrittura (la velocità aumenta ovviamente nelle tracce esterne).

Partizionamento di un hard disk

Tra i vantaggi del partizionamento, con lo scopo di una migliore gestione del disco abbiamo:

- Necessità di separare al massimo aree di dati logicamente molto differenti tra loro;
- Evitare Seek-Time troppo elevati

Scheduling dell'hard disk (tradizionale)

Per accedere in diversi punti del disco il braccetto impiega un tempo proporzionale allo spostamento compiuto; la spezzata che congiunge tutti i punti determina la lunghezza totale.

Gli obiettivi principali della gestione del disco è quella di minimizzare i tempi di accesso al dispositivo stesso. Se occorre quindi accedere a dati che sono memorizzati sull'hard disk in diverse posizioni, allora occorre stabilire una metodologia ottimale che determini in quale ordine sia preferibile accedere a tali dati. L'ordine di accesso ai dati contenuti sull'hard disk ne determina il tempo totale.

Il tempo d'accesso ha due componenti principali:

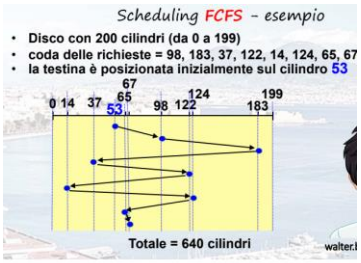
il **tempo di ricerca**(seek time) è il tempo necessario affinchè il braccio dell'unità a disco sposti le testine fino al cilindro contente il settore desiderato.

La **latenza di rotazione** (rotational latency) è il tempo aggiuntivo necessario perché il disco ruoti finchè il settore desiderato si trovi sotto la testina.

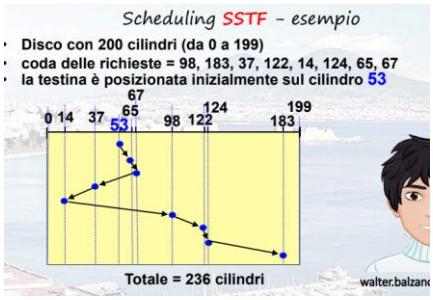
L'ampiezza di banda del disco è il numero totale di byte trasferiti diviso il tempo totale intercorso fra la prima richiesta e il completamento dell'ultimo trasferimento.

Metodologie di scheduling

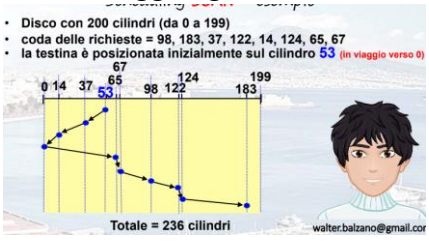
- FCFS le richieste vengono soddisfatte nell'ordine in cui esse sono pervenute (semplice da realizzare ma non ottimizza la distanza da percorrere)



- SSTF (Shortest Seek Time First) si seleziona la richiesta più vicina rispetto all'attuale posizione della testina (si può incorrere in situazioni di starvation).



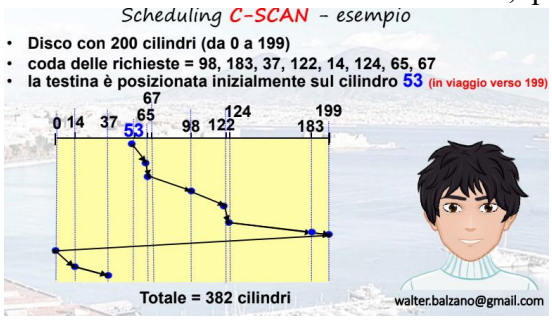
- SCAN il braccetto parte da un'estremo e si sposta nella sola direzione possibile, servendo le richieste mentre attraversa i cilindri, fino a raggiungere l'altro estremo, a questo punto inverte la marcia e la procedura continua.



Scheduling per scansione circolare C-SCAN

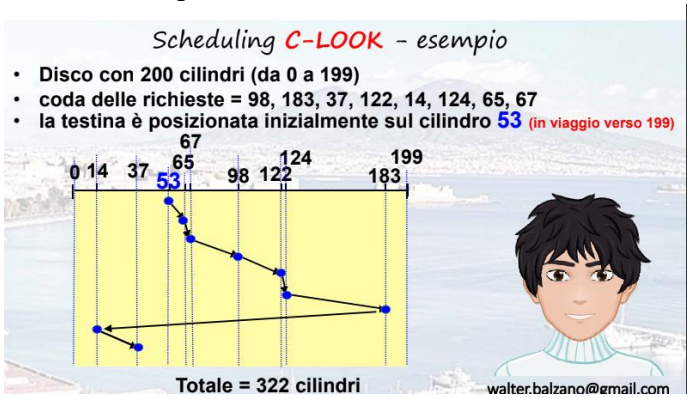
Variante dello SCAN concepita con lo scopo di garantiere un tempo di attesa meno variabile. A differenza dello SCAN classico, una volta raggiunto l'estremo, il braccetto ritorna immediatamente all'inizio del disco stesso, senza servire richieste durante di viaggio di ritorno.

L'algoritmo tratta il disco come una lista circolare, quindi come se il primo e l'ultimo cilindro fossero adiacenti.



Scheduling C-LOCK

Versione di C-SCAN in cui il braccio si sposta solo finchè ci sono altre richieste da servire in ciascuna direzione, dopo di che cambia immediatamente direzione, senza giungere all'estremo del disco



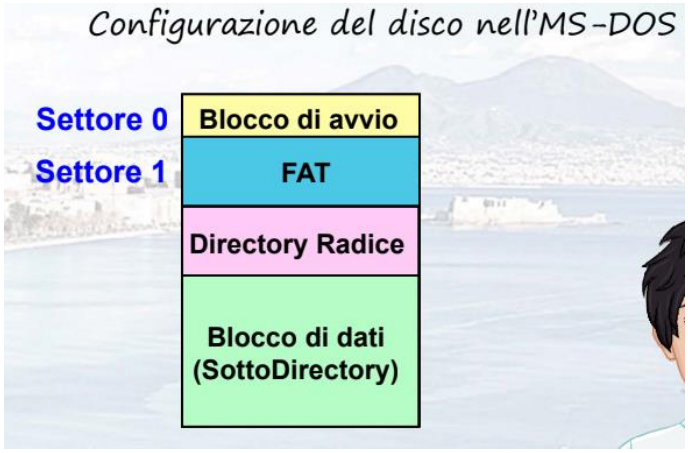
Gestione dell'unità a disco

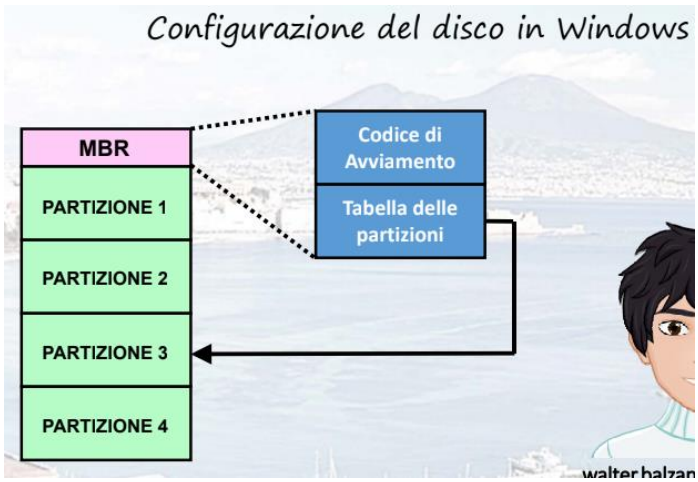
Prima che possa memorizzare dati, il disco deve essere suddiviso in settori che possano essere letti/scritti dal controllore. Il SO deve registrare quindi le proprie strutture dati all'interno del disco. Ciò avviene in 2 passi:

1. Partizionare il disco
2. Creare il file system(formattazione logica)

Il blocco di avviamento inizializza il sistema. Il bootstrap loader è memorizzato nella ROM.

La formattazione fisica mette anche da parte dei settori di riserva non visibili al SO: si può istruire il controller affinchè sostituisca da un punto di vista logico un settore difettoso con uno dei settori di riserva non utilizzati. Questa strategia è nota come accantonamento di settori (sector sparing).





Gestione dell'area di avvicendamento (swap space)

La memoria virtuale usa lo spazio dei dischi come estensione della memoria centrale. L'area d'avvicendamento può essere ricavata all'interno del file system o può trovarsi in una partizione separata del disco. In base alle performance del disco e del pc si sceglie tra: swap partition, swap file e zram.

Configurazioni RAID

Combinazione fra più hard disk con lo scopo di migliorare l'affidabilità del sistema oppure le prestazioni o entrambi gli aspetti. Le tecniche RAID possono essere realizzate si HW che SW.

Gran parte della strategia RAID è basata sulla capacità di poter leggere/scrivere in parallelo un insieme di dischi collegati al nostro sistema.

E' possibile realizzare differenti tipologie:

- RAID level 0: permette la distribuzione automatica dei dati su più dischi ma visto che i dati non sono replicati non garantisce la tolleranza ai guasti in caso di rottura di un disco
- RAID level 1: I dati vengono replicati e nel caso di rottura di un disco si passa automaticamente ad un altro disco senza perdita dei dati
- RAID level 3: simile al RAID 0, ma dedica un hard disk al recupero automatico mediante il controllo di parità
- RAID level 5; usa data striping complete e un disco per la correzione d'errore

Connessione dei dischi

I calcolatori accedono alla memoria secondaria in due modi: tramite le porte di I/O o per mezzo di un file system distribuito.

Compiti SO

Gestione dei dispositivi fisici di cui il SO realizza 2 astrazioni:

- Dispositivo a basso livello: un semplice vettore di blocchi di dati
- File system: il SO accoda e organizza le richieste provenienti da diverse applicazioni.