Capitolo 1

SISTEMA OPERATIVO: Programma che agisce come intermediario tra l'utente e gli elementi fisici del calcolatore. E' un insieme di programmi (software) che gestisce gli elementi fisici del calcolatore (hardware). E' il solo programma che funziona sempre nel calcolatore (nucleo).

SCOPI:

- Fornire un ambiente nel quale un utente possa eseguire programmi in modo conveniente ed efficiente.

COSA FA:

- Assicura il corretto funzionamento del calcolatore.
- Controlla e coordina l'uso dei dispositivi da parte dei programmi d'applicazione per gli utenti
- Assegna le risorse (fisiche e programmi).

SISTEMA DI CALCOLO:

1) Dispositivi fisici:
    - CPU
    - Memoria
    - Dispositivi di I/O
2) Sistema operativo
3) Programmi d'applicazione
4) Utenti

MAINFRAME

- A LOTTI : L'utente non interagisce direttamente col sistema di calcolo, ma prepara un lavoro d'elaborazione (job) e lo affida all'operatore del sistema e attende i risultati. In presenza di errori, i risultati sono accompagnati da un'immagine del contenuto finale della memoria e dei registri (dump) al fine di individuare e correggere gli errori (debugging). **Compito del sistema operativo**: trasferire il controllo da un lavoro ad un altro.
  Svantaggi: CPU spesso inattiva

- MULTIPROGRAMMATI: La multiprogrammazione consente di aumentare l'utilizzo della CPU organizzando i lavori in modo tale da tenerla in continua attività. Il sistema operativo tiene contemporaneamente nella memoria centrale diversi lavori, di modo che, al verificarsi di un'operazione di I/O (per esempio), il sistema inizi l'elaborazione di un altro lavoro, sfruttando quello che altrimenti sarebbe un tempo di inattività della CPU.
  Compiti del sistema operativo: job scheduling, caricamento in memoria per l'esecuzione, gestione della memoria, CPU scheduling, controllo della concorrenza.
  Svantaggi: nessuna interazione con l'utente.

- A PARTIZIONE DI TEMPO: La partizione del tempo d'elaborazione (time sharing o multitasking) è un'estensione logica della multiprogrammazione: la CPU esegue più lavori commutando le loro esecuzioni con una frequenza tale da permettere l'interazione dell'utente col proprio programma durante l'esecuzione. Lo 'scambio' tra un processo e un altro non è più dato dal solo verificarsi di eventi, ma anche dallo scadere un un prefissato intervallo di tempo (quanto di tempo). Un sistema di questo tipo permette a più utenti di condividere contemporaneamente il calcolatore, dando l'illusione ad ogni utente di disporre dell'intero calcolatore.
  Compiti del sistema operativo: gestione e protezione della memoria, virtualizzazione della memoria, file system, CPU scheduling, comunicazione e sincronizzazione dei processi, assicurare il non verificarsi di situazioni di stallo.

Processo: programma caricato nella memoria e predisposto per la fase d'esecuzione.

DESKTOP: In questi sistemi si è mirato alla comodità e prontezza d'uso per l'utente (PC). Questi sistemi hanno beneficiato di alcuni aspetti sviluppati per i mainframe, anche se non di tutti. Inoltre sorgono altre esigenze, come la protezione dei file a estranei.

SISTEMI CON PIU' CPU (sistemi paralleli, strettamente connessi): Le cpu sono in stretta comunicazione, condividono i canali di comunicazione (bus) e i clock e a volte anche la memoria e i dispositivi di I/O.
Vantaggi:

- Maggiore produttività
- Economia di scala

- Incremento dell'affidabilità

In un sistema costituito da due cpu identiche, collegate tra loro tramite un bus, una è primaria, mentre la seconda è di riserva (backup). Affinché il funzionamento del sistema continui anche alla presenza di guasti, durante l'esecuzione, in corrispondenza di determinati punti di verifica (checkpoint) si copia l'informazione di stato di ogni processo dalla cpu primaria a quella di riserva. Al verificarsi del guasto, si riavvia la copia di riserva a partire dal più prossimo checkpoint.

- ✓ Multielaborazione simmetrica (SMP): ciascuna cpu esegue unì'identica copia del sistema operativo comunicando tra di loro solo quando necessario.
- ✓ Multielaborazione asimmetrica (AMP): a ogni cpu si assegna un compito specifico, mentre una cpu principale controlla il sistema e assegna il lavoro alle cpu secondarie.

SISTEMI DISTRIBUITI: Si basano sulle reti per realizzare le proprie funzioni, sfruttando la capacità di comunicazione.

- Sistemi client-server: I sistemi centralizzati fungono da sistemi server pensati per soddisfare le richieste generate dai sistemi client. I sistemi server possono essere:
  i) Di calcolo: un client invia una richiesta d'esecuzione che il server svolge riportando il risultato al client
  ii) Di file: fornisce un'interfaccia al file system dando la possibilità al client di interagire coi suoi file.
- Sistemi paritetici (sistemi debolmente connessi): le reti di calcolatori sono costituite da un insieme di cpu che non condividono né memoria né clock; ciascuna ha la propria memoria localee comunicano tra di loro per mezzo di linee di comunicazione.

Rete: canale di comunicazione tra due o più sistemi
Sistema operativo di rete: offre funzioni come la condivisione di file tra i calcolatori della rete e lo schema di comunicazione utilizzato per lo scambio dei messaggi tra i processi dei diversi calcolatori della rete.

BATTERIE DI SISTEMI (cluster system): Sono basate sull'uso congiunto di più cpu; differiscono dai sistemi paralleli per il fatto che sono composte da due o più calcolatori completi collegati tra loro. Calcolatori che condividono la memoria di massa e sono connessi per mezzo di una rete locale. Quando si presenta un malfunzionamento, il calcolatore che svolge il controllo può appropriarsi dei mezzi di memorizzazione del calcolatore malfunzionante e riavviare le applicazioni che erano in esecuzione.

- Batterie asimmetriche: un calcolatore rimane nello stato di attesa attiva mentre l'altro esegue le operazioni. Il primo controlla il secondo, entrando in funzioni solo quando l'ultimo presenta un guasto.
- Batterie simmetriche: due o più calcolatori eseguono le operazioni e si controllano reciprocamente.

Non potendo offrire funzioni di file system condiviso, questi sistemi non permettono l'accesso condiviso ai dati presenti nei dischi; usualmente tramite la gestione distribuita degli accessi.

Reti di memorizzazione d'area: permettono l'associazione di più unità di memorizzazione a più calcolatori.

SISTEMI D'ELABORAZIONE IN TEMPO REALE
Presenta vincoli di tempo fissati e ben definiti entro i quali si deve effettuare l'elaborazione

- Sistemi d'elaborazione in tempo reale stretto: i dati si memorizzano in memoria a breve termine. Questi sistemi mancano di gran parte delle caratteristiche comuni agli altri sistemi operativi più progrediti, come per esempio la virtualizzazione della memoria. Sono in conflitto con il modo di funzionamento dei sistemi a partizione di tempo.
- Sistemi d'elaborazione in tempo reale debole: hanno caratteristiche meno restrittive: i processi d'elaborazione in tempo reale critici hanno priorità sugli altri processi e la mantengono fino al completamento dell'esecuzione.

SISTEMI PALMARI
Svantaggi:

- Memoria limitata
- CPU lenta
- Schermo piccolo

A mano a mano che i pc progrediscono sia nell'architettura che nel sistema operativo, la linea che separa i sistemi mainframe dai microcalcolatori diventa sempre più sfumata.

AMBIENTI D'ELABORAZIONE

- Tradizionale
- Basata sul Web
- Integrati

Capitolo 2

Al fine di evitare che i programmi utenti interferiscano con le operazioni proprie del sistema, l'architettura del calcolatore deve fornire meccanismi appropriati per assicurarne il corretto funzionamento.

FUNZIONAMENTO DI UN CALCOLATORE

- CPU
- CONTROLLORI DI DISPOSITIVI che, tramite il bus, sono connessi alla MEMORIA. Ognuno di essi si occupa di un particolare tipo di dispositivo fisico

La cpu e i controllori possono operare concorrentemente, il CONTROLLORE DI MEMORIA garantisce la sincronizzazione degli accessi

PROGRAMMA D'AVVIAMENTO: specifico programma che inizializza i diversi componenti di sistema, carica in memoria il nucleo del sistema operativo e ne avvia l'esecuzione. Si attiva all'avviamento del sistema ed è in genere contenuto in una ROM.

EVENTO: segnalato da un'interruzione dell'attuale sequenza d'esecuzione della cpu, può essere causato da:
- Dispositivo fisico → INTERRUPT
- Programma → ECCEZIONE (trap), che può essere a sua volta:
  - Evento eccezionale riconosciuto dall'architettura della cpu (es. divisione per 0)
  - Chiamata di sistema :richiesta specifica effettuata da un programma utente per ottenere l'esecuzione di un servizio del sistema operativo.

Un sistema operativo si dice **guidato dalle interruzioni** in quanto resta inattivo in assenza di operazioni da eseguire, in attesa che si verifichi un evento.

Un interrupt causa il trasferimento del controllo all'appropriata procedura di servizio dell'evento associato, allocata in memoria. Questa locazione contiene l'indirizzo iniziale della procedura di servizio di quel dato segnale di interruzione. Per gestire gli interrupt in maniera rapida si ricorre ad un vettore di puntatori alle specifiche procedure, sfruttando anche il fatto che il numero di segnali di interruzione è predefinito → attivazione delle procedure di servizio in modo indiretto attraverso il vettore delle interruzioni senza procedure intermediarie.
Se la procedura di gestione dell'interruzione richiede la modifica dello stato della cpu, deve salvare esplicitamente lo stato corrente per poterlo ripristinare prima di restituire il controllo.Una volta completata l'esecuzione dell'operazione richiesta, la cpu riprende l'elaborazione precedentemente interrotta, caricando nel program counter l'indirizzo alla prossima istruzione da eseguire.

STRUTTURA DI I/O

Un controllore di dispositivo dispone di una propria memoria interna, detta memoria di transito (buffer), e di un insieme di registri. Il controllore è responsabile del trasferimento dei dati tra i dispositivi periferici a esso connessi e la propria memoria di transito.

INTERRUZIONI DI I/O

I/O sincrono: per iniziare un'operazione di I/O, la cpu carica i registri appropriati del controllore del dispositivo con il quale intende comunicare. Il controllore, a sua volta, esamina il contenuto di questi registri allo scopo di determinare l'azione da compiere, restituendo il controllo al processo utente, tramite un segnale di interruzione, solamente dopo il completamento dell'operazione di I/O.
→ l'attesa del completamento dell'operazione si può realizzare tramite una particolare istruzione d'attesa, *wait*, che sospende la cpu fino al successivo segnale d'interruzione.
Questo metodo esclude la possibilità di operazioni di I/O concorrenti su più dispositivi, così come esclude la possibilità di sovrapporre utili elaborazioni con l'I/O.
I/O asincrono: restituzione immediata del controllo al processo utente, senza attendere il completamento dell'operazione di I/O. Vantaggio:l'I/O può proseguire mentre il sistema esegue altre operazioni.
Sono necessari:

- Una chiamata del sistema che consenta al programma d'utente di attendere, se richiesto, il completamento dell'operazione;
- Una tabella di stato dei dispositivi [Dispositivo,indirizzo,stato], per poter tener traccia delle varie richieste di I/O eventualmente attive nello stesso istante. Se un dispositivo è impegnato a soddisfare una precedente richiesta, si memorizza il tipo di richiesta insieme con altri parametri nell'elemento della tabella corrispondente a quel dispositivo. Poiché altri processi possono sottoporre uno richiesta allo stesso dispositivo, il sistema operativo tiene una coda d'attesa per ciascun dispositivo di I/O.

1. un controllore di dispositivo invia un segnale di interruzione al sistema per richiedere un servizio;
2. il sistema operativo individua il dispositivo che ha effettuato il segnale
3. accede alla tabella dei dispositivi, risale al dispositivo interessato e ne modifica l'elemento indicante l'occorrenza dell'interruzione;
4. il dispositivo segnala il completamento dell'operazione tramite un'interruzione
5. se ci sono altre richieste nella coda d'attesa del dispositivo, il sistema operativo ne avvia l'elaborazione;
6. si restituisce il controllo tramite un'interruzione.

- Per rendere più agevole l'accesso ai dispositivi di I/O molti calcolatori forniscono l'I/O associato alla memoria (memory mapped I/O)→ si fa corrispondere ai registri dei dispositivi a intervalli di spazio d'indirizzi della cpu, in questo modo ogni operazione di lettura o scrittura a tali indirizzi, comporta il trasferimento diretto di dati con i registri del dispositivo (benché tali indirizzi non siano veramente indirizzi di locazione della memoria centrale)
- I/O programmato→ la cpu impiega l'interrogazione ciclica (pooling) del bit di controllo del dispositivo, per verificare se è pronto
- I/O guidato dalle interruzioni→ attesa di un'interruzione che segnali che il dispositivo è pronto.

ACCESSO DIRETTO ALLA MEMORIA

Il trasferimenti di dati bit a bit risulta molto oneroso, per questo si utilizza la tecnica dell'accesso diretto alla memoria (direct memory access –DMA). Una volta impostata l'area della memoria, i puntatori e i contatori per il dispositivo I/O, il controllore trasferisce un intero blocco di dati dalla propria memoria di transito direttamente nella memoria centrale, o viceversa, senza alcun intervento da parte della cpu. In questo modo il trasferimento richiede una sola interruzione per ogni blocco di dati trasferito. Il principio è lo stesso per quanto riguarda le operazioni di base della cpu. Un programma utente, o il sistema operativo, richiede il trasferimento di un blocco di dati. Il sistema individua l'area della memoria interessata, quindi il driver di dispositivo (un elemento del sistema operativo) imposta gli appositi registri del controllore DMA affinché questo impieghi gli appropriati indirizzi di sorgente e destinazione e la lunghezza del blocco da trasferire. Mentre il controllore DMA esegue il trasferimento dei dati, la cpu è libera di eseguire altri compiti, anche se più lentamente in quanto il DMA le sottrae cicli d'accesso alla memoria. Il controllore DMA notifica il completamento dell'operazione inviando un'interruzione alla cpu.

STRUTTURA DELLA MEMORIA

Memoria d'accesso diretto (RAM)→ è la sola area di memoria di grandi dimensioni direttamente accessibile dalla cpu, per questo un programma per essere eseguito deve risiedervi. E' strutturata come un vettore di parole di memoria, ognuna delle quali possiede un proprio indirizzo. L'interazione avviene per mezzo di una sequenza di istruzioni *load* e *store* opportunamente indirizzate.
La tipica sequenza d'esecuzione di un'istruzione, in un sistema con architettura di von Neumann, comincia con il prelievo (*fetch*) di un'istruzione dalla memoria centrale e il suo trasferimento nel registro d'istruzione. Quindi si *decodifica* l'istruzione, la si *esegue* e il risultato può essere scritto in memoria.
Svantaggi:

- La memoria centrale non è sufficientemente grande per contenere in modo permanente tutti i programmi e i dati richiesti;
- È un dispositivo di memorizzazione volatile

Perciò si ricorre all'ausilio della **memoria secondaria**

Memoria secondaria:
Dischi magnetici: L'efficienza di un disco è caratterizzata da:

- Velocità di trasferimento
- Tempo di posizionamento (trovare il cilindro desiderato), detto anche tempo di ricerca
- Latenza di rotazione (trovare il settore desiderato)

Può essere *rimovibile*(es. floppy)

Un'unità disco è connessa a un calcolatore attraverso il bus di I/O e il trasferimento dei dati è eseguito da controllori: adattatori (posti all'estremità relativa al controllore del bus); controllori dei dischi (incorporati in ciascuna unità a disco.

1. il calcolatore inserisce un comando nell'adattatore
2. l'adattatore invia il comando al controllore del disco
3. il controllore del disco agisce sugli elementi elettromeccanici dell'unità a disco e porta a termine il comando.

I controllori dei dischi di solito hanno una cache incorporata, di modo che il trasferimento dei dati avviene tramite la cache che ha una velocità elevata.

Nastri magnetici: Il tempo d'accesso dei nastri magnetici è migliaia di volte maggiore a quello dei dischi, poiché è sequenziale, anche se, una volta raggiunta la posizione desiderata, l'unità a nastro può leggere o scrivere informazioni a una velocità paragonabile a quella di un'unità a disco.. Il loro uso principale è quello di creare copie di backup.

GERARCHIA DELLE MEMORIE

Le memorie si possono ordinare gerarchicamente a secondo della velocità, del costo, e del fatto che siano o no volatili. Un sistema di memorizzazione deve bilanciare il rapporto tra questi fattori.In una struttura gerarchica può accadere che gli stessi dati siano mantenuti in diversi livelli del sistema di memorizzazione. Questo non costituisce un problema in un ambiente di calcolo che ammette l'esecuzione di un solo processo alla volta, ma nei sistemi a partizione di tempo, in ambienti con più unità di elaborazione e in quelli distribuiti bisogna garantire che il cambiamento di un dato si rifletta su tutti i livelli di memorizzazione (coerenza della cache). Questo si risolve a livello dell'architettura del sistema e quindi a un livello più basso di quello del sistema operativo.

Cache: si tratta di una memoria ad alta velocità tra la cpu e la memoria centrale. Quando si deve accedere a una particolare informazione, si controlla se è già presente nella cache; in tal caso si adopera direttamente quella copia, altrimenti la si preleva dalla memoria centrale e la si copia nella cache, poiché si presuppone che servirà ancora successivamente.
La gestione della cache è un importante problema di progettazione.
I registri programmabili presenti all'interno della cpu rappresentano una cache per la memoria centrale, così come la memoria centrale rappresenta una cache per la memoria secondaria.

ARCHITETTURE DI PROTEZIONE

I primi sistemi operativi si chiamavano monitor residenti, e con essi il sistema operativo cominciò a eseguire molte funzioni che prima erano di competenza del programmatore. In seguito si rese possibile la condivisione simultanea delle risorse di sistema tra più programmi, incrementando l'utilizzo del sistema ma trovandosi nella situazione in cui un errore di un programma poteva alterare il comportamento di molti processi. Molti errori di programmazione sono direttamente riconosciuti dall'architettura del sistema e gestiti dal sistema operativo→ il sistema operativo deve terminare l'esecuzione del programma, proprio come quando un utente richiede una terminazione anormale. In questi casi di solito si scrive l'immagine della memoria (dump) per trovare e correggere l'errore.

DUPLICE MODO DI FUNZIONAMENTO

Per proteggere sia il sistema operativo, sia gli altri programmi da qualsiasi programma non funzionante, esistono due modi di funzionamento,indicati dal *bit di modo*:

- Modo d'utente (1)
- Modo di sistema (0)

All'avviamento del sistema, il bit è posto nel modo di sistema. Si carica il sistema operativo che provvede all'esecuzione dei processi utenti nel modo d'utente. Ogni volta che si verifica un'interruzione o un'eccezione si passa dal modo d'utente a quello di sistema, ponendo a 0 il bit di modo. Perciò quando il sistema operativo riprende il controllo si trova nel modo di sistema; prima di passare il controlla al programma utente ripristina il modo utente riportando a 1 il valore del bit.
Si definiscono *istruzioni privilegiate* tutte quelle istruzioni macchina che possono causare danni allo stato del sistema, e vengono quindi effettuate in monitor mode. Il sistema operativo, in modo sistema, ha accesso indiscriminatamente a tutta la memoria.

PROTEZIONE

I/O

Allo scopo di impedire l'esecuzione di operazioni illegali di I/O da parte dell'utente, si definiscono privilegiate tutte le operazioni di I/O, facendole eseguire in modo di sistema. Il sistema operativo controlla che la richiesta sia valida e , in questo caso, porta a termine l'operazione di I/O e restituisce il controllo al programma utente

MEMORIA

E' necessario proteggere il vettore delle interruzioni e le procedure di servizio ad esse relative da ogni possibile

alterazione da parte dei programmi utenti. Bisogna inoltre proteggere il sistema operativo dai programmi utenti, e i

programmi utenti tra di loro. Per far ciò, si separa lo spazio in memoria di ogni singolo programma, determinandone

l'intervallo di indirizzi necessari e proteggendo la memoria al di là di questi. Si ricorre a due registri:

- Registro di base: contiene il più basso indirizzo della memoria fisica al quale il programma dovrebbe accedere
- Registro di limite: contiene la dimensione dell'intervallo.

La cpu confronta ciascun indirizzo generato nel modo d'utente con i valori contenuti nei due registri; nel caso in cui si tenti di accedere ad una zona riservata, si genera un segnale di eccezione al sistema operativo che gestisce l'evento come un errore. Questi due registri possono essere modificati solo in modo sistema.

CPU

Occorre assicurare che il sistema operativo mantenga il controllo dell'elaborazione, per impedire che un programma

utente entri in un ciclo infinito o non restituisca più il controllo alla cpu.

A tale scopo si può usare un temporizzatore programmabile affinché invii un segnale di interruzione alla cpu a intervalli di tempo specificati, fissi o variabili.
Un temporizzatore variabile si realizza assegnando un valore a un contatore e decrementandolo ad ogni impulso finché non raggiunge il valore 0 e genera un segnale d'interruzione. Naturalmente anche le istruzioni per modificare il funzionamento del temporizzatore si possono eseguire solo nel modo sistema. I temporizzatori vengono usati per realizzare la partizione del tempo d'elaborazione, dove il tempo concesso a ciascun utente è chiamato *quanto di tempo*.

1. modo utente
2. scade il quanto
3. il controllo torna al sistema che esegue le necessarie operazioni di amministrazione del sistema, salva i valori dei registri e delle proprie variabili interne e contenuto delle proprie strutture dati, aggiorna altri parametri allo scopo di preparare l'esecuzione del programma successivo (cambio di contesto-context switch)
4. il programma che riceve il controllo della cpu riprende l'esecuzione dal punto esatto in cui era stata interrotta.

Le **reti** che collegano più calcolatori possono essere:

- LAN
- WAN

Capitolo 3

COMPONENTI DEL SISTEMA

GESTIONE DEI PROCESSI

Processo d'elaborazione: un programma in esecuzione, l'unità di lavoro di un sistema. Può essere del sistema operativo o utente.
Un programma è un'entità passiva, mentre un processo è un'entità attiva, con un program counter che indica la successiva istruzione da eseguire. L'esecuzione di un processo deve essere sequenziale: la cpu esegue le istruzioni una dopo l'altra finchè il processo termina; a ogni istante si esegue al più una istruzione del processo→ due processi dello stesso programma si considerano cmq come due sequenze d'esecuzione separate.
Un processo, per svolgere i propr compiti, necessita di risorse, come il tempo della cpu, memoria e dispositivi I/O.
In quest'ambito il sistema operativo è responsabile di:

- Creazione e cancellazione dei processi;
- Sospensione e ripristino dei processi;
- Fornitura di meccanismi per la sincronizzazione dei processi;
- Fornitura di meccanismi per la comunicazione tra processi;
- Fornitura di meccanismi per la gestione delle situazioni di stallo.

GESTIONE DELLA MEMORIA CENTRALE

Per eseguire un programma è necessario che questo sia associato a indirizzi assoluti e sia caricato nella memoria. Durante l'esecuzione del programma, la cpu accede alle proprie istruzioni e ai dati provenienti dalla memoria, generando i suddetti indirizzi assoluti. Quando il programma termina, si dichiara disponibile il suo spazio in memoria. Per migliorare l'utilizzo della cpu e la rapidità d'esecuzione, occorre tenere molti programmi in memoria, gestendola opportunatamente.
In quest'ambito il sistema operativo è responsabile di:

- Tener traccia delle parti della memoria che sono attualmente usate e da chi
- Decidere quali processi si debbano caricare in memoria quando c'è spazio disponibile
- Assegnare e revocare lo spazio di memoria a seconda della necessità.

GESTIONE DEI FILE

File: raccolta di informazioni correlate definite dal loro creatore, formati da una sequenza di bit o record. Rappresentano programmi, sia sorgente sia oggetto, e dati. Sono generalmente organizzati in directory e l'accesso, in caso della presenza di più utenti, può essere controllato in lettura, scrittura o modifica.
In quest'ambito il sistema operativo è responsabile di:

- Creazione e cancellazione di file/directory
- Fornitura delle funzioni fondamentali per la gestione di file e directory
- Associazione dei file ai dispositivi di memoria secondaria
- Creazione di copie di backup dei file su dispositivi di memorizzazione non volatile.

GESTIONE DEL SISTEMA DI I/O

Uno degli scopi di un sistema operativo è nascondere all'utente le caratteristiche degli specifici dispositivi.Il sottosistema di I/O è composto da:

- Componente di gestione della memoria→ gestione del buffer, della cache, delle operazioni I/O asincrone e dell'esecuzione di più processi (spooling)
- Interfaccia generale per i driver dei dispositivi
- Driver specifici dei dispositivi→soltanto questi conoscono le caratteristiche dello specifico dispositivo su cui sono assegnati.
-

GESTIONE DELLA MEMORIA SECONDARIA

I dischi contengono la maggior parte dei programmi, che vi rimangono finchè non vengono caricati in memoria principale per essere eseguiti.
In quest'ambito il sistema operativo è responsabile di:

- Gestione dello spazio libero
- Assegnazione dello spazio
- Scheduling del disco

RETI

Sistema distribuito: Insieme, anche eterogeneo, di sistemi fisicamente separati, organizzato in modo da costituire un unico ambiente coerente che offre all'utente l'accesso alle proprie risorse. Ciascuna unità di elaborazione ha una propria memoria locale e un suo clock; le unità comunicano tra di loro tramite una rete di comunicazione, che può essere totalmente o parzialmente connessa. La condivisione delle risorse aumenta la funzionalità, la disponibilità dei dati e l'affidabilità.

SISTEMA DI PROTEZIONE

La protezione è definita da ogni meccanismo che controlla l'accesso da parte di programmi, processi o utenti delle risorse del calcolatore.
L'architettura di indirizzamento della memoria assicura che un processo possa svolgersi solo all'interno del proprio spazio d'indirizzi; il temporizzatore assicura che nessun processo occupi la cpu in un ciclo infinito; agli utenti non è concesso l'accesso ai registri di controllo dei dispositivi.

INTERPRETE DEI COMANDI

E' l'interfaccia tra l'utente e il sistema operativo. Alcuni sistemi operativi lo contengono nel nucleo, altri lo trattano come un programma speciale che si esegue quando si avvia un lavoro d'elaborazione oppure quando un utente inizia una sessione di lavoro in un sistema a partizione di tempo.
Interprete di riga di comando (shell): preleva ed esegue la successiva istruzione di comando.
Le istruzioni di comando riguardano la creazione e la gestione dei processi, la gestione dell'I/O, della memoria secondaria e centrale, l'accesso al fle system, la protezione e la comunicazione tramite la rete.

SERVIZI DI UN SISTEMA OPERATIVO

- **Esecuzione di un programma**
- **Operazioni di I/O**
- **Gestione del file system**
- **Comunicazioni:** possono avvenire tra processi sullo stessi calcolatore o tra processi su calcolatori uniti da una rete. Può avvenire tramite:
  - Memoria condivisa
  - Scambio di messaggi
- **Rilevamento d'errori:** per ciascun tipo d'errore il sistema operativo deve saper intraprendere l'azione giusta per assicurare un'elaborazione corretta e coerente
- **Assegnazione delle risorse**
- **Contabilizzazione dell'uso delle risorse**
- **Protezione**

CHIAMATE DI SISTEMA

Sono l'interfaccia tra un processo e il sistema operativo, disponibili sotto forma di istruzioni in linguaggio assemblativo o di alto livello. Anche programmi molto semplici possono fare un uso intensivo del sistema operativo, anche se il compilatore e il suo insieme di funzioni d'ausilio alla fase d'esecuzione nascondono al programmatore la maggior parte dei dettagli d'interfaccia al istema operativo.

Per passare parametri al sistema operativo:

- Si passano in registri
- Se ci sono più parametri che registri, si memorizzano in un blocco di memoria e si passa l'indirizzo del blocco stesso nel registro

- Il programma colloca (push) i parametri in una pila; essi vengono poi prelevati (pop) dal sistema operativo.

Le chiamate di sistema possono essere classificate in:

1. **CONTROLLO DEI PROCESSI**

- Termine normale e anormale →(*end* o *abort*); in caso di terminazione anormale di solito si registra l'immagine del contenuto della memoria (dump). Il debugger si occupa della ricerca e correzione degli errori
- Caricamento,esecuzione→ (*load* e *execute*); un processo può richiedere il caricamento e l'esecuzione di un altro programma.
- Creazione ed arresto di un processo→ (*create process* e *end process*) Se un programma chiama un altro programma, questi possono continuare l'esecuzione in modo concorrente.
- Esame e impostazione degli attributi di un processo→ (*get process attributes* e *set process attributes*); quando si crea un nuovo processo si deve mantenerne il controllo, perciò si deve poter determinare e reimpostare i suoi attributi
- Attesa per il tempo indicato→ (*wait* tempo); può essere necessario attendere che i processi terminino la loro esecuzione, per un certo periodo di tempo
- Attesa e segnalazione di un evento→ (*wait* evento e *signal* evento); l'attesa può anche riguardare il verificarsi di un dato evento, che deve a sua volta segnalare l'esecuzione avvenuta.
- Assegnazione e rilascio di memoria.

2. **GESTIONE DEI FILE**

- Creazione e cancellazione di file→ (*create* e *delete*)
- Apertura, chiusura→ (*open* e *close*)
- Lettura, scrittura e posizionamento→ (*read, write* e *reposition*)
- Esame e impostazione degli attributi di un file → (*get file attribute* e *set file attribute*)

3. **GESTIONE DEI DISPOSITIVI**

- Richiesta e rilascio di un dispositivo → (*request* e *release*); poiché un sistema può avere più utenti, bisogna richiedere prima il dispositivo per averne l'esclusiva e rilasciarlo terminato l'uso.
- Lettura, scrittura e posizionamento → (*read, write* e *reposition*)
- Esame e impostazione degli attributi di un dispositivo
- Inserimento logico ed esclusione logica di un dispositivo

La somiglianza tra dispositivi di I/O e file è tale che alcuni sistemi operativi li fondono in un0unica struttura

file/dispositivo, identificando i dispositivi con nomi di file speciali.

4. **GESTIONE DELLE INFORMAZIONI**

- Esame e impostazione dell'ora e della data→ (*time* e *date*)
- Esame e impostazione dei dati del sistema
- Esame e impostazione degli attributi dei processi, file e dispositivi → (*get process attributes* e *set process attributes*)

5. COMUNICAZIONE

- **Scambio di messagi:** le informazioni si scambiano per mezzo di una funzione di comunicazione tra processi fornita dal sistema operativo
  - Creazione e chiusura di una connessione→ (*open connection* e *close connection*); si apre un collegamento tra due processi. Il nome dell'altro comunicante e del processo devono essere noti e trasformati in identificatori (*get hostid* e *get processid*)
  - Invio e ricezione di messaggi→ (*read message* e *write message*)
  - Informazioni sullo stato di un trasferimento
  - Inserimento ed esclusione di dispositivi remoti

E' utile quando ci sono pochi dati da trasferire e anche quando il trasferimento avviene tra calcolatori distinti

Memoria condivisa: per accedere alle aree di memoria possedute da altri processi, i processi utilizzano chiamate del sistema *map memory***.** Normalmente il sistema operativo impedisce a un processo l'accesso alla memoria di un altro processo, per cui, nel modello di comunicazione a memoria condivisa, i processi devono concordare e superare questo limite. In questo modo si possono scambiare informazioni leggendo e scrivendo i dati nelle aree di memoria condivisa, avendo la responsabilità di non scrivere contemporaneamente nella stessa posizione.

Vantaggi: velocità e convenienza; svantaggi: problemi di protezione e sincronizzazione.

PROGRAMMI DI SISTEMA

Offrono un ambiente più conveniente per lo sviluppo e l'esecuzione dei programmi.

- Gestione dei file: compiono operazioni sui file e le directory
- Informazioni di stato
- Modifica dei file: editor
- Ambienti d'ausilio alla programmazione: compilatori, assemblatori, interpreti dei linguaggi di programmazione comuni.
- Caricamento ed esecuzione dei programmi:caricatori assoluti, rilocabili, linkage editor, caricatori di sezioni sovrapponibili di programmi (overlay), debugger
- Comunicazioni: offrono i meccanismi per creare collegamenti virtuali tra processi, utenti e calcolatori.

Programma d'applicazione: Programma che risolve problemi comuni o esegue operazioni comuni.

Il programma di sistema più importante è l'**interprete dei comandi.** I comandi si possono realizzare:

- L'interprete dei comandi contiene il codice per l'esecuzione del comando. Ossia, arriva il comando che provoca un salto dell'interprete dei comandi a una sezione del suo stesso codice che imposta i parametri e invoca le idonee chiamate di sistema → la dimensione dell'interprete è fissata dal numero di comandi in esso contenuti
- L'interprete dei comandi non 'capisce' il comando, ma ne impiega semplicemente il nome per identificare un file da caricare nella memoria per essere eseguito→ la dimensione dell'interprete è abbastanza piccola e non necessita alcuna modifica quando si inseriscono nuovi comandi.

STRUTTURA DEL SISTEMA

SEMPLICE (v. fig 3.7 p 78)

STRATIFICATA

Si suddivide il sistema operativo in un certo numero di strati, ciascuno costruito sopra gli strati inferiori, a partire dal livello fisico (0) all'interfaccia utente (n). Il vantaggio principale offerto da questo metodo è la modularità: gli strati sono composti in modo che ciascuno di essi usi solo funzioni e servizi che appartengono agli strati inferiori. In questo modo, se durante la messa a punto di un particolare strato si riscontra un errore, questo deve esser in quello strato, poiché gli strati inferiori sono già stati corretti→ la stratificazione semplifica la progettazione e la realizzazione di un sistema.

Ogni strato nasconde a quelli superiori l'esistenza di determinate strutture di dati, operazioni ed elementi fisici.

Problemi:

- Poiché uno strato può utilizzare solo strati che si trovano ad un livello inferiore, è difficile definire gli strati
- Meno efficienza, poiché i parametri, passando di strato in strato, vengono modificati; oppure i dati vengono trasferiti di strato in strato

Possibile soluzione→ meno strati con più funzioni.

MICRONUCLEO

Si progetta il sistema operativo rimovendo dal nucleo tutti i componenti non essenziali, realizzandoli come programmi del livello d'utente e di sistema→ nucleo di dimensioni inferiori che generalmente offre servizi di gestione dei processi, della memoria e di comunicazione(scambio di messaggi).

Vantaggi:

- Facilità di estensione del sistema operativo
- Se il nucleo deve essere modificato, i cambiamenti da fare sono circoscritti
- Il sistema operativo è semplice da adattare alle diverse architetture
- Incremento della sicurezza e dell'affidabilità

MACCHINE VIRTUALI

Essendo il sistema di calcolo costituito da vari strati, i programmi di sistema che si trovano sopra al nucleo, e quindi anche sopra lo strato fisico, possono usare indifferentemente chiamate di sistema o istruzioni di macchina. Alcuni sistema permettono, analogamente, ai programmi d'applicazione di chiamare programmi di sistema. Il metodo basato sulle macchine virtuali offre un'interfaccia identica all'architettura sottostante, di modo che ogni processo dispone di

una copia (virtuale) del calcolatore sottostante. Servendosi della partizione dell'uso della cpu e delle tecniche di memoria virtuale, un sistema operativo può creare l'illusione che un processo disponga della propria cpu e della propria memoria.→ l'utente dispone della propria macchina virtuale su cui eseguire qualsiasi sistema operativo o programma progettato per la macchina fisica sottostante.

Risulta però difficile ottenere un esatto duplicato della macchina sottostante. Come la macchina fisica, anche quella virtuale deve disporre di due modi d'uso (i programmi che servono per le macchine virtuali si eseguono in modo di sistema, mentre ciascuna macchina virtuale può funzionare nel modo d'utente)

Vantaggi:

- Proteggendo completamente le risorse di sistema, fornisce un'efficace livello di sicurezza
- Permette lo sviluppo del sistema senza turbarne l'ordinario funzionamento

Svantaggi:

Non c'è condivisione diretta della risorse:

- Possibilità di condividere un minidisco
- Definizione di una rete di macchine virtuali, ciascuna delle quali può inviare informazioni tramite la rete di comunicazione virtuale.

JAVA

I programmi scritti in linguaggio Java sono eseguiti tramite una macchina virtuale; ciò consente l'esecuzione di un programma scritto in JAVA su ogni calcolatore che dispone di una macchina virtuale per tale linguaggio, costituita da:

- Caricatore delle classi
- Verificatore delle classi
- Interprete del linguaggio
    - Può interpretare gli elementi del bytecode uno alla volta
    - JIT (just-in-time)- traduce il bytecode nel linguaggio specifico del calcolatore

Gli oggetti si specificano con il costrutto *class* e un programma consiste in una o più classi. Per ogni classe, il compilatore produce un file *(*.class*) contenente il bytecode, ossia codice nel linguaggio di macchina della JVM indipendente dall'architettura sottostante.

1. Il caricatore carica i file *.*class*
2. Il verificatore controlla la correttezza sintattica del bytecode
3. se il controllo ha esito positivo, la classe viene eseguita dall'interprete.

La JVM gestisce la memoria in modo automatico, ripulendola dagli oggetti non più in uso (garbage collection).

PROGETTAZIONE E REALIZZAZIONE DI UN SISTEMA

Scopi:

- D'utente
  - Utile, facile, affidabile, sicuro, efficiente
- Di sistema
  - Di facile progettazione, realizzazione, manutenzione
  - Flessibile, affidabile, senza errori, efficiente

Meccanismo: come eseguire qualcosa

Criterio: cosa eseguire

Sarebbe preferibile disporre di meccanismi generali: un cambiamento di criterio implicherebbe solo la ridefinizione di alcun parametri del sistema.

Realizzazione

Tradizionalmente i sistemi operativi si scrivevano in linguaggio assemblativo, attualmente si scrivono in liguaggio di alto livello.

Vantaggi:

- Si scrive più rapidamente il codice
- Codice più compatto, più facile da capire e mettere a punto
- Adattabilità

Svantaggi:

- Minore velocità d'esecuzione
- Maggior occupazione di spazio in memoria

Soluzione: una volta scritto il sistema operativo e verificato il suo corretto funzionamento, è possibile identificare le procedure che possono costituire ' strozzature' e sostituirle con procedure equivalenti scritte in linguaggio assemblativo.

GENERAZIONI DI SISTEMI

Solitamente si progettano sistemi operativi da impiegare in macchine di una stessa classe con configurazioni diverse, quindi il sistema si deve configurare o generare per ciascuna situazione specifica →**generazione di sistemi** (SYSGEN)

Sono necessarie informazioni su:

- CPU e opzioni installate
- Quantità di memoria disponibile
- Dispositivi disponibili (indirizzo, numero di dispositivo, numero del segnale d'interruzione, tipo, modello….)
- Opzioni del sistema operativo richieste o i valori dei parametri che è necessario usare.

Dopo che un sistema operativo è stato generato, deve essere eseguito dal calcolatore.

Avviamento (booting): procedura di caricamento del nucleo. Nella maggior parte dei sistemi c'è un piccolo segmento di codice (programma d'avviamento) in una ROM, che individua il nucleo e ne avvia l'esecuzione.

Capitolo 4

Sistema a lotti → lavori(job)
Sistema a partizione di tempo → programmi utenti (task)
Attività della cpu → processi

Processo(sezione di testo): programma in esecuzione.
Comprende:

- Attività corrente
  - Valore del program counter
  - Contenuto registri della cpu
- Pila del processo (stack)
  - Dati temporanei
    - Paramentri di un metodo
    - Indirizzi di rientro
    - Variabili locali
  - Sezione dati
    - Variabili globali

Stato del processo:

In ciascuna cpu può essere in esecuzione solo un processo per volta, mentre molti processi possono essere pronti o in attesa.

Descrittore di processo (blocco di controllo di un processo –PCB)

- Stato del processo
- Program counter
- Registri di cpu
  - Accumulatori
  - Registri d'indice
  - Stack pointer
  - Registri d'uso generale
  - Registri contenenti informazioni relative ai codici di condizione
- Informazioni sulla scheduling della cpu
  - Priorità del processo
  - Puntatori alle code di scheduling
  - Altri parametri di scheduling
- Informazioni sulla gestione della memoria
  - Registri di base e di limite
  - Tabelle delle pagine o dei segmenti
- Informazioni di contabilizzazione delle risorse
  - Tempo d'uso della cpu
  - Tempo reale d'utilizzo della cpu
  - Limiti di tempo

    - Numero processi
- Informazioni sullo stato dell'I/O
    - Lista dei dispositivi assegnati al processo
    - Elenco dei file aperti
    -

Thread: percorso d'esecuzione all'interno di un processo

SCHEDULING DEI PROCESSI

L'obiettivo della partizione del tempo d'esecuzione della cpu è di commutarne l'uso tra i vari processi così frequentemente che gli utenti possano interegire con ciascun programma mentre esso è in escecuzione.

CODE DI SCHEDULING

Coda dei processi pronti: è una coda dei PCB dei processi pronti in memoria centrale. Quando si 'libera', la cpu prende un processo da questa coda.
Coda di dispositivo: è una coda dei processi che attendono la disponibilità di un particolare dispositivo di I/O.

Un nuovo processo si colloca nella coda dei processi pronti; poi è assegnato alla cpu ed è in esecuzione, finchè:

- Emette una richiesta di I/O, viene inserito nella coda del dispositivo→ si mette in attesa finchè non termina l'I/O, ritorna nello stato di pronto e viene reinserito nell'apposita coda;
- Crea un nuovo processo figlio di cui attende l'esecuzione→ si mette in attesa finchè non termina il figlio, ritorna nello stato di pronto e viene reinserito nell'apposita coda;
- Viene rimosso forzatamente a causa di un'interruzione → viene reinserito nella coda dei processi pronti.

SCHEDULER

E' il mezzo che compie la selezione di un processo da una coda.

Processo con prevalenza di I/O(I/O bound)→la maggior parte del suo tempo è dedicata all'esecuzione di operazioni di I/O
Processo con prevalenza d'elaborazione (cpu bound)→ la maggior parte del suo tempo è dedicata all'elaborazione

Scheduler a breve termine (scheduler di cpu): sceglie i lavori tra quelli pronti per essere eseguiti e assegna la cpu a uno di essi.→ si esegue frequentemente
Scheduler a lungo termine (job scheduler): sceglie i lavori tra quelli memorizzati in memoria secondaria e li carica in quella centrale affinché vengano eseguiti.→si esegue con poca frequenza, a seconda del grado di programmazione, ossia il numero di processi in memoria. Questo scheduler deve scegliere una buona combinazione di processi con prevalenza di I/O e d'elaborazione.
Scheduler a medio termine: presente in alcuni sistemi operativi, si occupa dell'avvicendamento (swapping) dei processi tra l'elaborazione e la memoria, riducendo il grado di multiprogrammazione.

Quando passa da un processo a un altro, si verifica un cambio di contesto (context switch), in cui il nucleo registra il contesto del vecchio processo nel suo PCB e carica il contesto precedentemente registrato del nuovo processo scelto per l'esecuzione.

OPERAZIONI SUI PROCESSI

CREAZIONE

Durante la propria esecuzione, un processo(genitore) può creare nuovi processi(figli). Ciascuno di essi può fare altrettanto, formando un albero di processi. I figli possono:

- Ottenere tutte le risorse tramite il sistema operativo, oppure
- Poter accedere solo a un sottoinsieme delle risorse del processo genitore

Quando un processo ne crea uno nuovo, per quanto riguarda l'esecuzione

- Il genitore continua l'esecuzione concorrentemente ai figli
- Il genitore attende che alcuni o tutti i figli abbiano terminato l'esecuzione

..e per ciò che riguarda lo spazio d'indirizzi

- Il figlio è un duplicato del genitore
- Nel processo figlio si carica un programma (vedi per esempio le chiamate *fork* ed *exec* in unix)

TERMINAZIONE

Un processo termina quando

- Finisce l'esecuzione della sua ultima istruzione e usa la chiamata di sistema *exit*
- Viene terminato da un altro processo, generalmente il genitore, tramite una chiamata di sistema *abort* (il genitore conosce l'identità dei propri figli). Ciò può avvenire perché
    - Il figlio ha usato più risorse di quante gli erano assegnate
    - Il compito del figlio non è più richiesto

- Il processo genitore termina l'esecuzione e il sistema operativo non consente a un processo figlio di continuare l'esecuzione in questa circostanza (terminazione a cascata).

PROCESSI COOPERANTI

Processo indipendente: non può influire su altri processi del sistema o subirne l'influsso→ non condivide dati.
Processo cooperante: influenza altri processi del sistema e ne è influenzato→ condivide dati.

Vantaggi

- Condivisione di informazioni
- Accelerazione del calcolo
- Modularità
- Convenienza

Paradigma per processi cooperanti: un processo **produttore** produce informazioni che sono consumate da un processo **consumatore**. Per permettere l'esecuzione concorrente dei processi produttore e consumatore, occorre disporre di un vettore di elementi inseriti dal produttore e prelevati dal consumatore. In caso di memoria illimitata, un produttore può sempre produrre, mentre in caso di memoria limitata, deve attendere se il vettore è pieno (il consumatore deve sempre attendere nel caso in cui il vettore sia vuoto). Per risolvere questo problema si può:

- Usare una funzione di comunicazione tra processi (interprocess comunication-IPC)
- Codificare esplicitamente utilizzando memoria condivisa.

COMUNICAZIONE TRA PROCESSI

Attraverso le funzioni IPC i processi possono comunicare tra loro senza condividere lo stesso spazio d'indirizzi→ particolarmente utile in un ambiente distribuito

SCAMBIO DI MESSAGGI

I servizi sono forniti sotto forma di ordinari processi utenti→operano al di fuori del nucleo
Operazioni:

- *Send* (`messaggio`)
- *Receive* (`messaggio`)

Dimensione messaggio:

- Fissa→facile realizzazione, difficile programmazione
- Variabile → difficile realizzazione, facile programmazione

P e Q, per comunicare, necessitano di un canale di comunicazione

Comunicazione diretta

- *Send* (P, `messaggio`) → invia `messaggio` a P
- *Receive* (Q, `messaggio`) → riceve, in `messaggio,` un messaggio da Q

Il canale di comunicazione ha le seguenti caratteristiche:

- Tra ogni coppia di processi che intendono comunicare si stabilisce automaticamente un canale→i processi devono conoscere la reciproca identità.
- Un canale è associato esattamente a due processi
- Esiste esattamente un canale tra ciascuna coppi di processi

Simmetria d'indirizzamento: il trasmittente e ricevente devono nominarsi a vicenda

- *Send* (P, `messaggio`)
- *Receive* (id, `messaggio`) → riceve, in `messaggio,` un messaggio da qualsiasi processo

[id identifica il processo con cui è avvenuta la comunicazione].
Asimmetria d'indirizzamento: soltanto il trasmittente deve nominare il ricevente, non viceversa.

Comunicazione indiretta

I messaggi s'inviano tramite delle porte (cassette postali - mailbox), identificate in modo univoco.

- *Send* (A, `messaggio`) → invia `messaggio` alla porta A
- *Receive* (A, `messaggio`) → riceve, in `messaggio,` un messaggio dalla porta A.

Il canale di comunicazione ha le seguenti caratteristiche:

- Si stabilisce un canale solo se condivide la stessa porta
- Un canale può essere associato a più processi
- A ogni coppia possono essere associati più canali corrispondenti alle rispettive porte

Poiché una porta può essere condivisa tra più di due processi, sorgono problemi su chi di essi riceve il messaggio. Soluzioni:

- Associare un canale solamente a 2 processi
- Consentire a un solo processo per volta di eseguire una *receive*
- Consentire di decidere arbitrariamente chi riceverà il messaggio

Il sistema operativo consente ad un processo di:

- Creare e rimuovere una porta
- Inviare e ricevere messaggi tramite essa

Questa porta appartiene al processo:

- Il processo è il proprietario e quindi unico ricevente--> non ci sono problemi su chi riceve il messaggio. D'altronde il diritto di proprietà e di ricezione possono passare anche ad altri processi.
- L'utente può solo inviare messaggi alla porta
- La porta fa parte dello spazio d'indirizzi del processo
- Al termine dell'esecuzione del processo, la porta scompare.

Esistono anche porte appartenenti al sistema, che sono indipendenti dai processi.

SINCRONIZZAZIONE

Invio:

- Sincrono (bloccante): chi invia il messaggio deve attendere che il ricevente o la porta lo riceva.
- Asincrono (non bloccante): il processo invia il messaggio e riprende la propria esecuzione.

Ricezione:

- Sincrona: il ricevente è bloccate durante l'attesa dell'arrivo di un messaggio
- Asincrona: il ricevente continua la sua esecuzione finchè non riceve un messaggio valido oppure un valore nullo.

Ricezione + invio bloccanti : rendez-vous

I messaggi scambiati tra processi comunicanti possono risiedere in code temporanee di messaggi:

- Capacità zero: il canale non può avere messaggi in attesa → sistema a scambio di messaggi senza memorizzazione transitoria (no buffering)
- Capacità limitata: la coda può contenere fino a n messaggi, quando è piena il trasmittente deve attendere che si liberi dello spazio → memorizzazione transitoria automatica (automatic buffering)
- Capacità illimitata: la coda ha lunghezza potenzialmente infinita; il trasmittente non si ferma mai →automatic buffering

COMUNICAZIONE NEI SISTEMI CLIENT –SERVER

Socket: estremità di un canale di comunicazione.

Due processi che comunicano tramite una rete hanno ognuno un socket identificato da <indirizzo ip, porta>

Server→ attende le richieste dei client stando in ascolto su una porta specifica. Al client viene affidato un numero di porta superiore a 1024. Ogni connessione deve essere unica, perciò ogni coppia di socket è diversa dalle altre.

La comunicazione tramite viene considerata di basso livello, mentre, per ciò che riguarda la comunicazione ad alto livello nei sistemi distribuiti si ricorre alle chiamate di procedure remote (RPC) o all'invocazione di metodi remoti (RMI)

CHIAMATE DI PROCEDURE REMOTE

E' un meccanismo simile a quello IPC, solo che si inviano messaggi strutturati e non semplici pacchetti dati. Questi messaggi si inviano alla macchina remota a un demone RPC e contengono un identificatore della funzione da eseguire coi relativi parametri. Una volta svolta la funzione, il sistema remoto restituisce i risultati al richiedente. La porta è un numero inserito all'inizio dei pacchetti dei messaggi → all'interno di un indirizzo un sistema può avere molte porte, ognuna delle quali contraddistingue un diverso servizio di rete.
La semantica delle RPC permette a un client di invocare una procedura presente in un sistema remoto allo stesso modo in cui invocherebbe una procedura locale.

1. Stub: segmento di codice di riferimento nel client, uno per ogni procedura remota.
2. Marshalling: strutturazione dei parametri → assemblaggio dei parametri in una forma trasmissibile per rete.

3. Scheletro: segmento di codice di riferimento nel server, invoca la procedura e restituisce i risultati

Per evitare che una procedura remota venga chiamata più volte, spesso gli si affida un timestamp.

INVOCAZIONE DI METODI REMOTI

E' una funzione del linguaggio Java che permette a un thread di invocare un metodo di un oggetto remoto (ossia residente in un'altra macchina virtuale).

Differenze con le RPC:

- Le RMI invocano metodi su oggetti remoti, mentre le RPC possono chiamare solo procedure o funzioni remote.
- Le RMI permettono di passare oggetti come parametri, mentre le RPC solo normali strutture dati.

La comunicazione avviene in maniera analoga alle RPC → il client invoca un metodo remoto, chiamando in realtà lo stub, si procede col marshalling, si inserisce tutto in un pacchetto (parcel) . Lo scheletro del server lo riceve, procede con l'unmarshalling e invoca il metodo. Infine, lo scheletro struttura il valore di ritorno in un pacchetto e lo invia al client.

Se i parametri strutturati sono oggetti locali sono passati per copiatura, secondo una tecnica chiamata serializzazione dell'oggetto. Se i parametri invece sono oggetti remoti sono passati per riferimento.

Capitolo 5

Thread (processo leggero, lightweight process –LWP): percorso di controllo all'interno di un processo. E' l'unità di base d'uso della CPU. E' composto da:

- Identificatore
- Program counter
- Insieme di registri
- Stack

Condivide coi thread dello stesso processo dati, codice e altre risorse del sistema.
Un processo con un solo thread è detto processo pesante, altrimenti è un processo multithread.

Vantaggi:

- Tempo di risposta: permette a un programma di continuare la sua esecuzione anche se una parte di esso è bloccata o sta eseguendo un'operazione particolarmente lunga.
- Condivisione delle risorse: i thread condividono la memoria e le risorse del processo cui appartengono → un processo può avere molti thread di attività diverse nello stesso spazio d'indirizzi.
- Economia: la creazione di un processo è più onerosa della creazione di un thread.
- Uso di più unità d'elaborazione: nei sistemi con più cpu, i thread si possono eseguire in parallelo; in quelli con una sola cpu li si può avvicendare velocemente creando l'illusione di un'esecuzione parallela.
- Permettono la comunicazione tra processi → particolarmente utile nelle RPC.

La gestione dei thread può avvenire su due livelli

- **Thread al livello d'utente:** sono gestiti come uno strato separato sopra il nucleo del sistema operativo, e sono realizzati tramite una libreria di funzioni per la creazione, lo scheduling e la gestione senza alcun intervento diretto del nucleo→ la creazione e lo scheduling avvengono nello spazio d'utente senza richiedere l'intervento del nucleo, perciò sono operazioni veloci. Svantaggi: se il nucleo è a singolo thread, ogni thread d'utente che esegue una chiamata di sistema bloccante causa il blocco dell'intero sistema, anche se altri thread sono disponibili per l'esecuzione.
- **Thread al livello del nucleo:** sono gestiti direttamente dal sistema operativo, ossia il nucleo si occupa della creazione, dello scheduling e della gestione dello spazio d'indirizzi→ sono più lenti, ma se si verifica una chiamata bloccante il sistema operativo può attivare l'esecuzione di un altro thread dell'applicazione. In un sistema con più cpu, il nucleo può far eseguire più thread parallelamente.

MODELLI

DA UNO A MOLTI: si fanno corrispondere molti thread a livello d'utente a un thread a livello del nucleo. Vantaggi: gestione a livello utente → agevole; svantaggi: se un thread effettua una chiamata bloccante, blocca l'intero processo; non si possono eseguire thread parallelamente su calcolatori con più cpu.
DA UNO A UNO: si fa corrispondere un thread a livello d'utente a un thread a livello del nucleo. Vantaggi: se un thread effettua una chiamata bloccante, un altro thread del processo può proseguire, si possono svolgere più thread parallelamente in sistemi con più cpu; svantaggi: la creazione di un thread d'utente comporta la creazione di un thread a livello di nucleo, che ha un carico notevole.
DA MOLTI A MOLTI: si fa corrispondere più thread a livello d'utente ad altrettanti o meno thread a livello del nucleo. Vantaggi: i programmatori possono creare liberamente i thread che ritengono necessari, e i corrispondenti thread del livello del nucleo si possono eseguire in parallelo nelle architetture con più cpu. Se un thread effettua una chiamata di sistema bloccante, il nucleo può fare in modo che si esegua un altro thread.

PROGRAMMAZIONE MULTITHREAD

FORK ED EXEC

Se un thread invoca la chiamata di sistema `fork`, il nuovo processo potrebbe:

- Contenere un duplicato di tutti i thread
- Contenere solo il thread invocante

Se un thread invoca la chiamata di sistema `exec`, il programma specificato come parametro della `exec` sostituisce

l'intero processo, inclusi tutti i thread.

- Se si invoca subito la `exec` dopo la `fork`, la duplicazione dei thread non è necessaria in quanto il programma invocato sostituirà il processo → si duplica solo il chiamante
- Se la `exec` non si invoca immediatamente dopo la `fork` potrebbe essere utile la duplicazione di tutti i thread del processo genitore.

CANCELLAZIONE

La cancellazione di un thread è l'operazione che permette di terminarlo prima che completi il suo compito, ed è chiamato thread bersaglio.

- Cancellazione asincrona → un thread fa immediatamente terminare il thread bersaglio

In questo caso il sistema operativo si riappropria delle risorse assegnate al thread, ma questo tipo di cancellazione potrebbe non liberare una risorsa necessaria a tutto il sistema.

- Cancellazione differita→ il thread bersaglio controlla periodicamente se deve terminare

Questo metodo consente di programmare la verifica in un punto dell'esecuzione in cui il thread può essere cancellato senza problemi (punto di cancellazione).

GESTIONE DEI SEGNALI

Nei sistemi UNIX si usano segnali per comunicare ai processi il verificarsi di determinati eventi.

I segnali possono essere:

- Sincroni → accesso illegale alla memoria, divisione per 0; il segnale si invia al processo che ha causato l'errore
- Asincroni→ evento esterno al processo; il segnale si invia a un altro processo.

1. all'occorrenza di un particolare evento si genera un segnale
2. s'invia il segnale a un processo
3. una volta ricevuto, il segnale deve essere gestito
    a. tramite un gestore predefinito dei segnali
    b. tramite un gestore dei segnali gestito dall'utente

Per i processi multithread si pone il problema del thread a cui si deve inviare il segnale.

- Inviare il segnale a cui il thread si riferisce → segnale sincrono
- Inviare il segnarle a ogni thread del processo
- Inviare il segnale a specifici thread del processo → segnale asincrono
- Definire un thread specifico per ricevere i segnali diretti al processo

Le funzione di chiamate di procedure asincrone (APC) permettono a un thread del livello d'utente di specificare la funzione da chiamare quando il thread riceve la comunicazione di un particolare evento

GRUPPI DI THREAD

Problemi relativi alla programmazione multithread

- Tempo di creazione di un thread prima di poter soddisfare la richiesta, considerando anche il fatto che questo thread sarà terminato non appena avrà terminato il suo lavoro
- Un numero illimitato di thread potrebbe esaurire le risorse del sistema

Soluzione: gruppi di thread → si crea un numero di thread alla creazione del processo, organizzandoli in un gruppo nel quale attendono il lavoro che gli sarà richiesto.

1. il server riceve una richiesta
    a. se il gruppo non contiene alcun thread disponibile, il server attende fino al rientro di un thread nel gruppo;
    b. se c'è un thread disponibile nel gruppo, lo attiva passandogli la richiesta
2. Il thread completa il suo lavoro
3. il thread rientra nel gruppo d'attesa.

Vantaggi:

- si elimina l’attesa della creazione di un nuovo thread poiché il thread è già esistente
- si limita il numero di thread esistenti contemporaneamente, così non si esauriscono le risorse del sistema

Dati specifici di thread: copia privata di alcuni dati di un thread, non condivisa con gli altri thread dello stesso processo.

[Poiché in Java i thread sono gestiti dalla JVM e non da una libreria di thread al livello d’utente o del nucleo, i thread del linguaggio Java non si possono considerare né thread a livello d’utente né del livello del nucleo.]

Capitolo 6

Con la multiprogrammazione si tengono in contemporaneamente in memoria più processi, e quando un processo deve attendere un evento, il sistema operativo gli sottrae il controllo della cpu per cederlo ad un altro processo → scheduling della cpu

Ciclo di un processo: l'esecuzione di un 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), quindi da un'altra sequenza di operazioni della cpu, di nuovo una sequenza di operazioni di I/O e così via. L'ultima sequenza di operazioni della cpu si conclude con una richiesta al sistema di terminare l'esecuzione.

Programma con prevalenza di I/O (I/O bound) → molte sequenze di cpu di breve durata.
Programma con prevalenza d'elaborazione (cpu bound) → poche sequenze di cpu, lunghe.

SCHEDULER DELLA CPU

E' lo scheduler a breve termine a scegliere, tra i processi pronti, quello da assegnare alla cpu. I processi nella coda dei processi pronti possono essere 'sistemati' secondo diverse politiche, e generalmente gli elementi della coda sono i descrittori dei processi (PCB).

Si prendono decisioni riguardanti lo scheduling della cpu nel caso in cui:
1. un processo passa dallo stato di esecuzione allo stato di attesa (es. I/O)
2. un processo passa dallo stato di esecuzione allo stato di pronto (es. interrupt)
3. un processo passa dallo stato di attesa allo stato di pronto (es. completamento dell'I/O)
4. un processo termina.

Il 1 e 4 caso non comportano una scelta di scheduling, ma solo il 'prendere' un nuovo processo da eseguire → se lo scheduling interviene solo in questi casi, si dice **senza diritto di prelazione** (un processo rimane in possesso della cpu finchè non termina o passa allo stato di attesa)

Diritto di prelazione: facoltà riconosciuta a un soggetto, di acquisire un bene con precedenza su altri soggetti, a parità di condizioni.

Inconvenienti dello scheduling con diritto di prelazione:

- il nucleo non può esercitare la prelazione su un processo mentre le strutture dati del nucleo si trovano in uno stato incoerente
- poiché le interruzioni si possono verificare in ogni istante e il nucleo non può sempre ignorarle, le sezioni di codice eseguite per effetto delle interruzioni devono essere protette da un uso simultaneo. Per evitare che più processi accedano in modo concorrente a tali sezioni di codice, queste disattivano le interruzioni al loro inizio e le riattivano alla fine → ciò richiede tempo.

DISPATCHER

Modulo che passa effettivamente il controllo della cpu ai processi scelti dallo scheduler a breve termine:
1. cambio di contesto
2. passaggio al modo utente
3. salto alla giusta posizione del programma utente per riavviare l'esecuzione

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

CRITERI DI SCHEDULING

- Utilizzo della cpu: la cpu deve essere il più attiva possibile;

- Produttività: numero di processi completati nell'unità di tempo;
- Tempo di completamento: intervallo di tempo che intercorre tra la sottomissione del processi e completamento dell'esecuzione. (tempi d'attesa per essere caricati in memoria + tempo d'attesa nella coda dei processi pronti + tempo di esecuzione della cpu + tempo delle operazioni di I/O)
- Tempo d'attesa: somma degli intervalli d'attesa passati nella coda dei processi pronti;
- Tempo di risposta: tempo che intercorre tra la sottomissione della richiesta e la prima risposta prodotta (tempo per iniziare la risposta, non il suo tempo di emissione)

Utilizzo e produttività della cpu devono essere al massimo, mentre i tempi devono essere ridotti al minimo. Un sistema il cui tempo di risposta sia ragionevole e prevedibile può essere considerato migliore di un sistema mediamente più rapido, ma molto variabile.

ALGORITMI DI SCHEDULING

IN ORDINE D'ARRIVO (first-come, first-served –FCFS) [*SENZA DIRITTO DI PRELAZIONE]*

La cpu si assegna al processo che la richiede per primo, fondandosi sul criterio di una coda FIFO. Quando un processo entra nella coda dei processi pronti, si collega il suo PCB all'ultimo elemento della coda. Quando è libera, si assegna la cpu al processo che si trova alla testa della coda dei processi pronti, rimovendolo da essa.
Il tempo medio d'attesa non è in genere minimo e può variare al variare della durata delle sequenze di operazioni della cpu dei processi.
Effetto convoglio: tutti i processi attendono che un lungo processo liberi la cpu → riduzione dell'utilizzo della cpu e dei dispositivi.

PER BREVITA' (shortest-job-fisrt –SJF ; oppure **shertest next cpu burst)**

Associa a ogni processo la lunghezza della successiva sequenza di operazioni della cpu, scegliendo, quando la cpu è disponibile, quella con la più breve lunghezza. Se due processi hanno le successive sequenze di operazioni della cpu della stessa lunghezza, si applica lo scheduling FCFS.
Il tempo d'attesa medio disunisce, quindi si dice che questo algoritmo è ottimale. D'altra parte è difficile conoscere la durata della successiva richiesta della cpu→questo algoritmo di scheduling si usa spesso nello scheduling a lungo termine (ossia quello che carica i processi dalla memoria secondaria a quella centrale).

Si può cercare di 'predire' un valore approssimato della lunghezza della successiva sequenza di operazioni della cpu, calcolando una media esponenziale delle effettive lunghezze delle precedenti sequenze di operazioni della cpu.[media esponenziale $\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$].

Questo algoritmo può essere:

- Con prelazione (shortest-remainig-time-first)→ se si presenta nella coda dei processi pronti un nuovo processo con una successiva sequenza di operazioni della cpu più breve del processo attualmente in esecuzione, sostituisce quest'ultimo in favore del nuovo arrivato;
- Senza prelazione→ permette al processo attualmente in esecuzione di terminare la propria sequenza di operazioni della cpu.

[NB. Questo algoritmo non è altro che un algoritmo per priorità in cui la priorità di un processo è data dall'inverso della lunghezza prevista della seguente sequenza di operazioni della cpu]

PER PRIORITA'

Si associa una priorità a ogni processo e si assegna la cpu al processo con priorità più alta; i processi con uguale priorità si ordinano secondo una schema FCFS. Le priorità sono definite:

- Internamente: usano quantità misurabili (es. limiti di tempo, requisiti di memoria…)
- Esternamente: definite secondo criteri esterni al sistema operativo

Questo algoritmo può essere:

- Con prelazione→ se si presenta nella coda dei processi pronti un processo con priorità più alta, lo si sostituisce a quello in esecuzione;
- Senza prelazione → il processo in esecuzione termina le sue operazioni di cpu e l nuovo arrivato viene semplicemente posto in testa alla coda dei processi pronti.

Attesa indefinita(starvation): un processo con priorità bassa può continuare a rimanere costantemente in attesa di utilizzare la cpu.
Invecchiamento (aging): soluzione alla starvation→si aumenta gradualmente la priorità dei processi in attesa, di modo che tutti vengano eseguiti prima o poi.

CIRCOLARE(Round Robin –RR) *[CON PRELAZIONE]*

E' simile al FCFS, ma ha la capacità di prelazione per la commutazione dei processi → progettato per i sistemi a partizione di tempo.

La coda dei processi pronti è circolare e ogni nuovo processo viene inserito alla fine della coda secondo una politica FIFO; lo scheduler della cpu scorre questa coda assegnando la cpu a ciascun processo per un intervallo di tempo della durata massima di un quanto di tempo.
Può succedere che:

- Un processo termina le operazioni di cpu prima dello scadere del quanto → restituisce il controllo della cpu e lo scheduler sceglie un nuovo processo;
- Scade il quanto
  1. un temporizzatore invia un segnale d'interruzione al sistema operativo,
  2. si esegue un cambio di contesto
  3. aggiunge il processo alla fine della coda dei processi pronti
  4. tramite lo scheduler si seleziona un nuovo processo

Bisogna scegliere opportunamente la grandezza del quanto di tempo, in particolare questo deve essere ampio rispetto alla durata del cambio di contesto. Un quanto troppo ampio tenderebbe a un algoritmo FCFS, mentre un quanto troppo breve ricorrerebbe a troppi cambi di contesto, aumentando i tempi.

Se il quanto di tempo è abbastanza breve, questo algoritmo si chiama **condivisione della CPU**, in quanto tutti gli utenti hanno l'illusione che ogni processo disponga della propria cpu.

A CODE MULTIPLE (multilevel queue scheduling algorithm)

Questo algoritmo è adatto a situazioni in cui i processi nella coda dei processi pronti si possono classificare in code diverse (ossia hanno tempi di risposta e necessità differenti), per esempio:

- primo piano (interattivi)
- sottofondo (a lotti)

Generalmente, i processi in primo piano hanno una priorità esterna su quelli di sottofondo.
Si possono avere più code differenti.
I processi si assegnano permanentemente ad una coda, che ha un proprio algoritmo di scheduling; inoltre è necessario avere uno scheduling tra le code.
[*Per esempio, si può impostare un quanto di tempo per le code e gestire la coda dei processi in primo piano con un algoritmo RR e quella dei processi di sottofondo con un algoritmo FCFS.]*

A CODE MULTIPLE CON RETROAZIONE (multilevel feedback queue scheduling)

Permette ai processi di migrare da una coda all'altra; ossia un processo che usa molto tempo di elaborazione della cpu può spostarsi in una coda con priorità più bassa e viceversa; analogamente si può evitare l'attesa indefinita di un processo spostandolo in una coda con priorità più alta.
I parametri di questo algoritmo sono:

- numero di code
- algoritmo di scheduling di ciascuna coda
- metodo usato per determinare quando spostare un processo in una coda con priorità maggiore/minore
- metodo usato per determinare in quale coda si deve mettere un processo quando richiede un servizio.

SCHEDULING PER SISTEMI CON PIU' CPU

Sistemi omogenei: le cpu sono identiche tra loro.

Condivisione del carico: si usa una coda dei processi pronti comune e si assegnano a una qualsiasi cpu disponibile.

a) Ogni cpu esamina la coda dei processi pronti e sceglie un processo da eseguire → bisogna accertarsi che 2 cpu non eseguano lo stesso processo e che i processi non vengano persi dalla coda;
b) Si fissa una cpu per lo scheduling delle altre cpu, creando una struttura gerarchica. Nel caso in cui si assegni ad una unica cpu (master server) lo scheduling, l'elaborazione delle operazioni di I/O e le altre attività di sistema, si parla di multielaborazione asimmetrica

SCHEDULING PER SISTEMI D'ELABORAZIONE IN TEMPO REALE

Prenotazione delle risorse: lo scheduler accetta un processo se e solo se è possibile garantirne il completamento entro i termini di tempo dichiarati (*hard real time*)

Per le elaborazioni in *soft real time* ci sono vincoli meno restrittivi, ossia i processi critici hanno una priorità più alta rispetto a quelli ordinari. Caratteristiche:

- Il sistema deve disporre di un algoritmo di scheduling per priorità, in modo da poter associare le priorità più elevate ai processi d'elaborazione in tempo reale;
- La priorità di questi non deve diminuire con il trascorrere del tempo, mentre ciò può accadere per i processi ordinari
- Latenza di dispatch bassa→ per garantire ciò, anche le chiamate di sistema possono essere sottoposte a prelazione.Come fare?
    - Punti di prelazione: punti all'interno di una chiamata di sistema particolarmente lunga che permettono la prelazione;
    - Tutto il nucleo può essere sottoposto a prelazione→in questo caso tutte le strutture dati del nucleo si devono proteggere tramite la sincronizzazione.

Protocollo di ereditarietà delle priorità: i processi che accedono alle risorse richieste dal processo con priorità più alta ereditano temporaneamente l'alto grado di priorità, fino al rilascio della risorsa contesa, dopodiché i processi riprendono la loro ordinaria priorità → questo risolve il problema dell'**inversione delle priorità**, in cui un processo ad alta priorità richiede di accedere a strutture a cui ha attualmente accesso un processo con priorità inferiore (il processo ad alta priorità si ritroverebbe quindi ad attendere il completamento di quello a bassa).

La fase conflittuale della latenza di dispatch consiste in:

1. prelazione di tutti i processi attualmente in esecuzione all'interno del nucleo
2. rilascio da parte dei processi a bassa priorità delle risorse richieste dal processo ad alta priorità.

VALUTAZIONE DEGLI ALGORITMI

- Valutazione analitica: secondo l'algoritmo dato e il carico di lavoro del sistema, fornisce una formula o un numero che valuta le prestazioni (modello deterministico).
I risultati sono facilmente confrontabili, poiché si tratta di numeri, ciononostante anche i parametri devono essere

numeri esatti, e la cosa non sempre è possibile. Si utilizzano i modelli deterministici soprattutto nella descrizione degli

algoritmi di scheduling e negli esempi.

- Il sistema di calcolo si può descrivere come una rete di unità serventi, ciascuna con una coda d'attesa. La cpu è un'unità servente con la propria coda dei processi pronti e il sistema di I/O ha le sue code dei dispositivi. Se sono noti l'andamento degli arrivi e dei servizi, si possono calcolare l'utilizzo, la lunghezza media delle code, il tempo medio d'attesa, ect…(**analisi delle reti delle code**)

FORMULA DI LITTLE

lunghezza media della coda – processo attuale → $n = \lambda \times W$ ← tempo medio d'attesa della coda

λ: andamento medio d'arrivo

Per poter calcolare la risposta, le reti di code spesso si limitano ad approssimare un sistema reale, rendendo discutibile la precisione dei risultati ottenuti.

- I dati necessari per condurre **una simulazione** si possono ottenere in vari modi, per esempio tramite un generatore di numeri casuali, tuttavia con questo metodo non si riproduce un sistema reale. Si può sottoporre il sistema reale a un controllo continuo, con la registrazione degli eventi effettivi (*trace tape*), usato poi per condurre la simulazione. Svantaggi: una simulazione dettagliata dà risultati precisi, ma richiede molto tempo, spazio in memoria; inoltre la creazione di un tale simulatore è impegnativa.

L'unico modo assolutamente sicuro per valutare un algoritmo di scheduling consiste nel codificarlo, inserirlo nel sistema operativo e osservarne il comportamento nelle reali condizioni di funzionamento del sistema, ma ciò ha un costo elevato.
Inoltre le prestazioni di un algoritmo di scheduling può variare al variare dell'ambiente in cui lo si usa. Gli algoritmi di scheduling più flessibili sono quelli che permettono di essere modificati nella fase di realizzazione del sistema operativo, di avviamento o di esecuzione tramite il cambiamento delle variabili.(a questo fine, è utile la separazione tra meccanismi e criteri) → pochi sistemi operativi hanno questo scheduling 'regolabile'.

Per ciò che riguarda i thread

I thread del livello d'utente sono gestiti da una libreria, mentre il nucleo non ne conosce l'esistenza. Per essere eseguiti dalla cpu, si fa corrispondere un thread a livello del nucleo, tramite un processo leggero (LWP)
Scheduling dei processi locali (process local scheduling): la libreria dei thread compie lo scheduling dei thread del livello d'utente per farli eseguire a un LWP disponibile → avviene a livello d'applicazione, non riguarda il sistema operativo
Scheduling di sistema globale (system global scheduling): il nucleo decide quale thread del libello del nucleo eseguire → è un'attività del sistema operativo

Capitolo 7

Un processo cooperante è un processo che può influenzare un altro processo in esecuzione nel sistema o anche subirne l'influenza. Questi processi possono condividere direttamente uno spazio logico d'indirizzi → thread, oppure condividono soltanto attraverso file.

Per evitare situazioni in cui più processi accedono agli stessi dati in modo concorrente e i risultati dipendono dall'ordine degli accessi, occorre assicurare che un solo processo alla volta possa modificare i dati condivisi → sincronizzazione.

PROBLEMA DELLA SEZIONE CRITICA

Sezione critica: segmento di codice in cui il processo può modificare variabili comuni e altri dati condivisi.
Ogni processo deve richiedere il permesso per entrare nella propria sezione critica. La sezione di codice che realizza questa richiesta è la sezione d'ingresso, mentre la sezione che segue quella critica è detta sezione d'uscita. La parte restante di codice è detta sezione non critica. Bisogna soddisfare i seguenti requisiti:

- Mutua esclusione: se un processo è nella sua sezione critica, nessun altro processo può essere in esecuzione nella propria sezione critica.
- Progresso: se nessun processo è nella sua sezione critica e qualche processo desidera farlo, solo i processi che si trovano nelle rispettive sezioni d'ingresso possono partecipare alla decisione riguardante la scelta del processo che può entrare per primo nella propria sezione critica
- Attesa illimitata: se un processo ha già richiesto l'ingresso nella sua sezione critica, esiste un limite massimo al numero di volte che si consente ad altri processi di entrare nelle proprie sezioni critiche prima che si accordi la richiesta al primo processo.

SOLUZIONE PER 2 PROCESSI

Ho 2 processi, P_0 e P_1

Algoritmo 1

Condividono una variabile `turno`, che può assumere valore 0 o 1. A seconda di questo valore, il processo con indice uguale a `turno` può eseguire la sua sezione critica, dopo di che cambia il valore di `turno` stesso.
Questa soluzione non soddisfa il requisito di progresso poiché richiede una stretta alternanza dei processi (infatti se P_0 vuole entrare nella sua sezione critica e P_1 non lo è, non può in ogni caso se `turno` =1)

Algoritmo 2

Condivido un vettore booleano `pronto` di dimensione 2 (poiché ho 2 processi), inizializzandone i valori a *false*.
Quando una dei processi è pronto a eseguire la propria sezione critica:

1. imposta a *true* il valore corrispondente in `pronto`
2. controlla che l'altro processo non sia anche lui a *true*
    a. in questo caso attende che l'altro processo termini e reimposti il valore a *false*
    b. altrimenti
3. esegue la sua sezione critica
4. reimposta il valore *false* nel vettore `pronto`.

Algoritmo 3

Ho 2 strutture: un vettore booleano `pronto` di dimensione 2 e una variabile condivisa `turno`. Quando un processo vuole entrare bella propria sezione critica:

1. imposta a *true* il bit corrispondente nel vettore `pronto`
2. assegna a `turno` il valore dell'altro processo
3. controlla il valore corrispondente all'altro processo nel vettore `pronto` e il valore di `turno`
    a. se anche l'altro processo è a *true* e il `turno` è suo → rimango in attesa che non valga più almeno una di queste condizioni
    b. altrimenti
4. entro nella sezione critica
5. terminata la sezione critica, imposto a *false* il bit corrispondente al processo corrente nel vettore `pronto`.

SOLUZIONE PER PIU' PROCESSI: algoritmo del fornaio.

1. Quando un processo deve entrare nella sua sezione critica, lo segnala in un vettore booleano;
2. gli viene attribuito un numero progressivo e riporta a false il valore nel vettore.
3. Per decidere quale processo debba entrare per primo nella propria sezione critica, si confrontano i numeri progressivi di quei processi che hanno già acquisito tale numero (infatti gli altri processi o non ce l'hanno

perché non sono interessati a svolgere la propria sezione critica, o stanno acquisendo il numero in quel momento). Entra nella propria sezione critica il processo con numero progressivo minore e, a parità di numero, quello con l'indice inferiore (poiché ogni processo deve avere indici diversi, è garantita la mutua esclusione).

4. Svolta la propria sezione critica, il processo setta il suo numero progressivo a 0.

ARCHITETTURE DI SINCRONIZZAZIONE

In un sistema con una sola cpu si potrebbe risolvere il problema della sezione critica semplicemente se si potessero impedire le interruzioni mentre si modificano le variabili condivise→ questa soluzione non è sempre possibile in sistemi con più cpu poiché comporterebbe sprechi di tempo e una diminuzione di prestazioni dell'intero sistema.

Per questo motivo molte architetture offrono la possibilità di modificare, controllare il contenuto o inviare parole di memoria in modo atomico attraverso speciali istruzioni, come

- ✓ **TestAndSet** si esegue in modo atomico, cioè non è soggetta a interruzioni e permette la mutua esclusione e il progresso, mentre non garantisce l'attesa limitata;
- ✓ **Swap** agisce sul contenuto di due parole di memoria; viene eseguita in modo atomico e garantisce la muta esclusione e il progresso.

SEMAFORI

Semaforo: strumento di sincronizzazione formato da una variabile intera con 3 operazioni associate

- ✓ inizializzazione con un valore non-negativo
- ✓ una operazione `wait` che ne decrementa il valore di una unità ed induce una attesa sul processo che la esegue in caso il valore del semaforo diventa negativo.
- ✓ una operazione `signal` che ne incrementa il valore di una unità; se il valore ottenuto non e' positivo allora uno dei processi in attesa per opera di una operazione di `wait` precedentemente eseguita viene liberato dall'attesa.
- ✓ Le operazioni devono essere eseguite in modo atomico.

Il principale svantaggio delle soluzioni del problema della mutua esclusione è che richiedono una condizione di attesa attiva (*busy waiting*)→ se un processo è nella sezione critica e un altro prova ad entrare nella propria, rimane sempre nel ciclo della sezione d'ingresso, sprecando cicli di cpu. Questi tipi di semafori sono detti spinlock (perché 'girano' finchè non rimangono 'bloccati' nell'attesa) e sono utili nei sistemi con più cpu e dove si prevedono tempi di attesa brevi.

Negli altri casi, in cui compiere un cambio di contesto quando un processo attende per un accesso sarebbe meno oneroso di un'attesa attiva, si possono modificare le definizioni di `wait` e `signal`:

1. quando un processo invoca un'operazione di `wait` e trova che il valore del semaforo non è positivo, invece di rimanere nell'attesa attiva blocca se stesso.
2. viene inserito in una coda d'attesa associata al semaforo e il processo viene messo nello stato di attesa
3. il controllo passa allo scheduler della cpu che sceglie un nuovo processo da eseguire
4. quando un altro processo esegue una `signal` sul semaforo, un processo bloccato si riavvia tramite una `wake up`, che modifica lo stato del processo da 'in attesa' a 'pronto'
5. il processo a questo punto è nella coda dei processi pronti e sarà messo in esecuzione a seconda del tipo di scheduling della cpu.

Le operazioni di bloccaggio e risveglio del processo sono chiamate del sistema di base.

Poiché occorre garantire che due processi non possano eseguire contemporaneamente operazioni di `wait` e `signal` sullo stesso semaforo, l'esecuzione dei semafori deve essere di tipo atomico:

- ✓ Per ambienti con una sola cpu basta inibire le interruzioni mentre si eseguono queste due operazioni;
- ✓ Per ambienti con più cpu si possono usare o istruzioni speciali, o trattare come sezioni critiche le sezioni di codice che costituiscono queste due operazioni.

Può succedere che in una coda d'attesa si verifichi uno stallo, ossia ciascun processo di un insieme in attesa attende che un altro processo, dello stesso insieme, causi un evento.

Semaforo contatore: il valore al suo intero può variare in un dominio logicamente non limitato.
***Semaforo binario**: il valore intero può essere soltanto 0 o 1.*

PROBLEMI TIPICI DI SINCRONIZZAZIONE

PRODUTTORI E CONSUMATORI CON MEMORIA LIMITATA

Abbiamo un vettore di n posizioni che costituisce la nostra memoria condivisa. Un semaforo mutex gestisce la mutua esclusione degli accessi a questo vettore, inoltre si utilizzano altri 2 semafori (`vuote` e `piene`:il primo conta le

posizioni vuote, il secondo quelle occupate)→la cooperazione avviene in base a due situazioni distinte, ciascuna descritta dallo stato di un semaforo:

All'inizio
- vuoto =n
- pieno =0

- ✓ il produttore può produrre se e solo se il buffer è ha posizioni vuote, riempiendole
- ✓ il consumatore può consumare se e solo se il buffer ha posizioni piene, svuotandole

LETTORI E SCRITTORI

Considerando un insieme di dati condivisi tra più processi, è logico supporre che alcuni di essi debbano semplicemente leggerli (lettori), mentre altri debbano apportarvi delle modifiche (scrittori). Se due lettori accedono allo stesso dato, non si ha alcun problema di coerenza, che sorge invece nel momento in cui provino ad accedervi uno scrittore e un lettore o due scrittori. Per risolvere questo problema, bisogna far accedere gli scrittori in modo esclusivo al dato condiviso. Due problemi:

1. *nessun lettore deve attendere, a meno che uno scrittore abbia già ottenuto il permesso di usare l'insieme di dati condivisi → ciò potrebbe portare ad un'attesa indefinita degli scrittori;*
2. *uno scrittore, una volta pronto, deve eseguire la propria scrittura al più presto → ciò potrebbe portare ad un'attesa indefinita dei lettori.*

Soluzione al primo problema: si condividono, tra processi lettori e scrittori, due semafori (`mutex` e `scrittura`) e una variabile (`numlettori`). I semafori vengono inizializzati a 1, la variabile a 0.

- ✓ `mutex` *assicura la mutua esclusione quando si aggiorna il valore di numlettori*
- ✓ `numlettori` *contiene il numero di processi che stanno attualmente leggendo i dati condivisi*
- ✓ `scrittura` *è un semaforo di mutua esclusione per gli scrittori e serve anche al primo o allìulòtimo lettore che entra/ esce dalla sezione critica.*

I CINQUE FILOSOFI

Ci sono cinque filosofi (processi) seduti stabilmente ad un tavolo rotondo e esiste un unico grande piatto al centro della tavola da cui ogni filosofo mangia (risorse condivise). Tra un filosofo ed un altro vi è una solo bacchetta. Ciascun filosofo non fa che tre cose: pensare quando è sazio, mangiare e mettersi in attesa di mangiare quando ha fame. Un filosofo decide di mettersi a pensare per un certo intervallo di tempo (casuale e finito), poi si siede a tavola e decide di mangiare il riso. Prima di poterlo fare, ha bisogno di prendere in mano due bacchette. Un filosofo può prendere solo le due bacchette che stanno alla sua destra e alla sua sinistra, una per volta, e solo se sono libere, ovvero non può sottrarre la risorsa bacchetta ad un altro filosofo che l'ha già acquisita (no prelazione). Finché non riesce a prendere le bacchette, il filosofo deve aspettare. Quando è sazio, posa le bacchette al loro posto e si mette a pensare per un certo tempo.La situazione è tale che due filosofi vicini non possono mai mangiare contemporaneamente. Obiettivi:

1. *non può accadere che tutti i filosofi rimangono indefinitamente bloccati in attesa reciproca che l'altro rilasci l'ulteriore bacchetta di cui hanno bisogno per mangiare (no deadlock)*
2. *non accade mai che uno o più filosofi non riescano mai a mangiare e quindi muoiano di fame perché gli altri sono più svelti di loro a prendere le bacchette (no starvation)*

Soluzioni

- ✓ *Ogni filosofo può acquisire prima la bacchetta che sta a destra e poi quella che sta a sinistra (usando un semaforo per ogni bacchetta)→ possibile che si crei la situazione di stallo se capita che tutti i filosofi abbiano fame contemporaneamente e quindi prendono la bacchetta di destra nello stesso istante.*
- ✓ *Solo quattro filosofi possono stare contemporaneamente a tavola*
- ✓ *Un filosofo può prendere le sue bacchette sole se entrambe disponibili (operazione in sezione critica)*
- ✓ *Un filosofo pari prende prima la bacchetta di sinistra, invece un filosofo pari prima quella di destra.*

In rari casi si possono riscontrare errori si sincronizzazione anche nei semafori→ bisognerebbe analizzare ciascun caso analizzando attentamente ciascun processo.

Per questo esistono costrutti di sincronizzazione ad alto livello:

- ✓ ***Regione critica***
- ✓ ***Monitor***

Si presuppone che un processo sia composto da alcuni dati locali e da un programma sequenziale che può operare su di essi. Ai dati locali può accedere solo il programma sequenziale, che è incapsulato all'interno del processo stesso; ciò significa che i processi non possono accedere direttamente ai dati locali di altri processi, pur potendo condividere i dati globali

REGIONE CRITICA

1. *Si dichiara la variabile che deve essere condivisa*
2. *si può accedere alla variabile solo dall'interno di un'istruzione* `region`

3. *all'interno di tale istruzione c'è un'espressione booleana*
    a. *se risulta vera, si esegue l'istruzione*
    b. *se risulta falsa si ritarda l'esecuzione finchè non risulta vera*

Questo costrutto non elimina tutti gli errori di sincronizzazione, ma ne riduce il numero.

MONITOR

- ✓ *Per permettere che un processo attenda all'interno del monitor sono necessarie delle variabili sulle quali ci si possa sospendere (*`condition x, y`*).*
- ✓ Su x e y si possono solo effettuare operazioni `x.wait` → sospende il processo che effettua questa operazione finchè un altro processo non effettua una `x.signal`
- ✓ L'operazione `x.signal` risveglia esattamente un processo, se nessun processo è in attesa l'operazione non ha alcun effetto.

Capitolo 8

Se un processo è in attesa di risorse trattenute da altri processi, anch'essi in stato d'attesa, il processo potrebbe non cambiare più il suo stato→ **stallo** (deadlock)

Un sistema è composto da un numero finito di risorse da distribuire tra più processi in competizione, rispettando la sequenza:

1. Richiesta: se un processo richiede una risorsa non immediatamente disponibile, deve attendere finchè non può acquisirla
2. Uso: il processo utilizza la risorsa
3. Rilascio: finito l'utilizzo, il processo rilascia la risorsa.

La richiesta e il rilascio avvengono tramite chiamate di sistema, quindi è il sistema operativo a controllare che il processo utente abbia richiesto una risorsa e che questa gli venga assegnata.
Lo stato di ogni risorsa e l'eventuale processo a cui è assegnata (+ i processi in attesa di utilizzarla), sono registrati in un'apposita tabella di sistema.

SITUAZIONI DI STALLO

CONDIZIONI NECESSARIE

Si può avere una situazione di stallo se e solo se si verificano contemporaneamente queste condizioni:

- **Mutua esclusione**: almeno una risorsa deve essere non condivisibile, ossia che può essere usata da un solo processo alla volta → se un altro processo richiede tale risorsa, lo si deve ritardare fino al rilascio della risorsa stessa.
- **Possesso e attesa:** un processo in possesso di almeno una risorsa attende di acquisire risorse già in possesso di altri processi
- **Impossibilità di prelazione**: una risorsa può essere rilasciata dal processo che la possiede solo volontariamente, dopo aver terminato il proprio compito.
- **Attesa circolare**: deve esistere un insieme di processi tale che il primo attende il secondo, il secondo il seguente e così via, fino all'ultimo che attende il primo.

GRAFO DI ASSEGNAZIONE DELLE RISORSE

E' un insieme di:

- Vertici (V)
  - $P= \{P_1, P2, \ldots P_n\}$ insieme dei processi del sistema
  - $R= \{R_1, R2, \ldots R_n\}$ insieme delle risorse del sistema
- Archi (E)
  - $P_i \rightarrow R_j$ arco di richiesta (il processo P_i richiede la risorsa R_j)
  - $R_j \rightarrow P_i$ arco di assegnazione (un'istanza del tipo di risorsa R_j è assegnata a P_i)

L'assenza di cicli nel grafo di assegnazione delle risorse implica l'assenza di situazioni di stallo.

Nel caso in cui ciascun tipo di risorsa presente nel ciclo abbia esattamente un'istanza, allora questa è una condizione necessaria e sufficiente per l'esistenza di uno stallo.
Nel caso in cui i tipi di risorse abbiano più istanze, la presenza di un ciclo è una condizione necessaria ma non sufficiente per l'esistenza di uno stallo.

METODI PER LA GESTIONE DELLE SITUAZIONI DI STALLO

1. **PREVENIRE** che si verifichi una situazione di stallo → significa usare metodi per assicurare che non si verifichi almeno una delle condizioni necessarie.
   - **Mutua esclusione:** un processo non dovrebbe mai attendere una risorsa non condivisibile → poiché alcune risorse lo sono intrinsecamente, non si può in generale evitare situazioni di stallo negando questa condizione;

- **Possesso e attesa:** occorre garantire che un processo che richiede una risorsa non ne possegga altre → si possono usare i seguenti protocolli:
  - ✓ Ogni processo, prima di iniziare la propria esecuzione, richiede tutte le risorse necessarie e gli vengono assegnate;
  - ✓ Un processo può richiedere una risorsa solo se non ne possiede

  Svantaggi:
  - ✓ Utilizzo delle risorse poco efficiente
  - ✓ Si può presentare una situazione di attesa indefinita
- **Impossibilità di prelazione:** occorre garantire che non sia mai possibile avere la prelazione su risorse già assegnate → si possono usare i seguenti protocolli:
  - ✓ Se un processo che ha già delle risorse richiede una risorsa per la quale deve attendere, si esercita la prelazione sulle risorse già assegnategli;
  - ✓ Se un processo richiede delle risorse e queste sono assegnate a un processo a sua volta in attesa, si sottraggono a quest'ultimo e si assegnano al richiedente.
- **Attesa circolare:** un modo per evitare che si verifichi questa condizione consiste nell'imporre un ordinamento totale sull'insieme R di tutti i tipi di risorse, e un ordine crescente di numerazione per le risorse richieste da ciascun processo. Protocolli:
  - ✓ Ogni processo può richiedere risorse solo seguendo un ordine crescente di numerazione [ossia, dopo aver richiesto un tipo di risorsa R_i, può chiedere istanze di R_j se e solo se $f(R_j) > f(R_i)$]
  - ✓ Si può stabilire che un processo, prima di richiedere un'istanza di R_j, rilasci qualsiasi R_i tale che se $f(R_i) \geq f(R_j)$

2. **EVITARE** che si verifichi una situazione di stallo → occorre che il sistema operativo abbia informazioni aggiuntive riguardanti le risorse che un processo richiederà ed userà durante le sue attività.
   Stato sicuro:stato in cui il sistema è in grado di assegnare risorse a ciascun processo (fino al suo massimo) in un certo ordine. Essere in uno stato non sicuro è una condizione necessaria ma non sufficiente per il verificarsi di uno stallo. Assicurando che il sistema sia in uno stato sicuro si evitano situazioni di stallo.
   - **Algoritmo con grafo di assegnazione delle risorse**

   Se le risorse di un sistema hanno una sola istanza, allora per garantire lo stato sicuro si può usare una variante del grafo di assegnazione delle risorse. Si aggiunge un altro tipo di arco, detto <u>arco di reclamo</u> (P_i R_j) che indica la possibilità che Pi richieda Rj. Le risorse devono essere reclamate a priori nel sistema. Se Pi richiede Rj, l'arco di reclamo può divenire arco di richiesta <u>se e solo se</u> non si crea un ciclo (lo si verifica tramite un algoritmo di rilevamento dei cicli)

   - Algoritmo del banchiere

   Quando si presenta un nuovo processo deve dichiarare il numero massimo di istanze di ciascun tipo di risorsa di cui necessita e verificare che il sistema si mantenga in uno stato sicuro.

n = **numero processi**

m= **tipi di risorsa**

Disponibili**: vettore di lunghezza m, indica il numero di istanze disponibili per ciascun tipo di risorsa {Disponibili [j]=k → sono disponibili k istanze del tipo di risorsa Rj}**

Massimo**: matrice n x m, definisce la richiesta massima di ciascun processo {Massimo [i,j]=k→ Pi può richiedere al più k istanze di Rj}**

Assegnate**: matrice n x m, definisce il numero di istanze di ciascun tipo attualmente assegnate a ogni processo {Assegnate [i,j]=k → a Pi sono assegnate k istanze di Rj}. Si può vedere anche come un insieme di vettori Assegnate$_i$, dove i è il numero del processo.**

Necessità **matrice m x n, indica la necessità residua di risorse relative a ogni processo {Necessità [i,j]=k → Pi necessita ancora k istanze di Rj}. Si può vedere anche come un insieme di vettori Necessità$_i$.**

Richieste$_i$**: vettore delle richieste del processo Pi {Richieste$_i$[j]=k → Pi richiede k istanze di Rj}**

Lavoro: **vettore di lunghezza m**

Fine: **vettore di lunghezza n**

i. **Se Richieste$_i$ ≤ Necessità$_i$** → ii
   Altrimenti errore, poiché il processo ha richiesto più risorse del dovuto

ii. **Se Richieste$_i$ ≤ Disponibili$_i$** → iii

Altrimenti Pi attende che si rendano disponibili le risorse richieste

iii. **Il sistema simula**
$Disponibili_i := Disponibili_i - Richieste_i$

$Assegnate_i := Assegnate_i + Richieste_i$

$Necessità_i := Necessità_i - Richieste_i$

E verifica che lo stato di assegnazione delle risorse sia sicuro; se non lo è ripristina il vecchio stato di assegnazione delle risorse.

iv. **Inizializza Lavoro := Disponibili; Fine [i] := falso**
v. **Cerca un indice i per cui valgono contemporaneamente**
Fine [i]:= falso ; $Necessità_i \leq$ Lavoro

Se tale i non esiste → vii

Altrimenti vi

vi. **Lavoro:= Lavoro + $Assegnate_i$; Fine[i]:= vero**
vii. **Se Fine [i]:= vero per ogni i, allora il sistema è in stato sicuro**

a) *Un processo richiede delle risorse*
- *Se sono più di quelle di cui necessita, si genera un errore*
- *Se sono uguali o meno di quante necessita*

b) *Si controlla che esse siano disponibili*
- *Se non lo sono, il processo attende che si liberino*
- *Altrimenti*

c) *Si simula l'assegnazione delle risorse al processo, aggiornando i valori relativi alle risorse disponibili (togliendo a quelle iniziali quelle appena richieste), a quelle assegnate (aggiungendo al valore iniziale quelle richieste, ossia il valore delle nuove risorse che si simula di assegnare) e a quelle necessarie (sottraendo al valore iniziale quelle appena richieste)*

d) *Si controlla che lo stato di assegnazione delle risorse sia sicuro. Come?*

e) *Iterativamente, per ogni processo la cui esecuzione non è ancora terminata e necessita meno o tante risorse quante quelle disponibili, gliele si assegna, lo si fa terminare e si 'restituiscono' le risorse assegnate a quelle disponibili (poiché ora sono 'libere', essendo terminato il processo). Se alla fine terminano tutti i processi, allora lo stato di assegnazione delle risorse è sicuro.*

3. RILEVARE **le situazioni di stallo e** RIPRISTINARE **il sistema**
   - Rilevamento**: si ricorre a questo tipo di algoritmi in base alla frequenza con la quale si prevede si verifichi uno stallo e il numero di processi coinvolti.**
     - **<u>Istanza singola di ciascun tipo di risorsa.</u>**
       Si può usare un algoritmo che rilevi dei cicli all'interno di un grafo di attesa, ottenuto togliendo al grafo di assegnazione delle risorse i nodi dei tipi di risorse e componendo gli archi tra i soli processi
     - **<u>Più istanze di ciascun tipo di risorsa.</u>**
       Si utilizza un algoritmo analogo a quello del banchiere per verificare lo stato sicuro del sistema, solo che al passo v **si verifica che $Richieste_i \leq$ Lavoro non $Necessità_i$ → se al passo** vii **esiste un i per cui Fine [i]= falso, allora il sistema è in stallo, in particolare lo è Pi.**
   - Ripristino**:**
     - **<u>Segnalazione di stallo e gestione manuale</u>**
     - **<u>Terminazione</u>**
       - ✓ **di tutti i processi in stallo**
       - ✓ **di un processo alla volta fino all'eliminazione del ciclo di stallo. Criteri di scelta:**

- **priorità dei processi**
- **tempo trascorso dalla computazione e tempo necessario al completamento**
- **quantità e tipo di risorse impiegate dai processi**
- **quantità e tipo di risorse ancora necessarie ai processi**
- **numero e tipo di processi che si devono terminare**

o **<u>Prelazione su risorse</u>**

Si sottraggono le risorse in successione ad alcuni processi assegnandole ad altri finchè non si interrompe il ciclo di stallo

- ✓ **Selezione della vittima**
- ✓ **Ristabilimento di un precedente stato sicuro → occorre stabilire che fare del processo vittima; terminarlo costa più che riportarlo ad un precedente stato sicuro, d'altronde per fare ciò bisogna mantenere più informazioni su tutti i processi in esecuzione**
- ✓ **Attesa indefinita → bisogna garantire che le risorse non vengano sottratte sempre allo stesso processo**

4. IGNORARE IL PROBLEMA **'fingendo' che le situazioni di stallo non possano mai verificarsi → degrado di prestazioni del sistema.**

Capitolo 9

ASSOCIAZIONE DEGLI INDIRIZZI

In genere un programma risiede in un disco in forma di un file binario eseguibile e per esser messo in esecuzione dei esser caricato in memoria centrale.
Coda d'ingresso: insieme dei processi presenti nei dischi in attesa di esser trasferiti in memoria.

1. si sceglie un processo nella coda d'ingresso
2. lo si carica in memoria, lo si esegue
3. al termine, si dichiara disponibile il suo spazio in memoria

Nella maggior parte dei casi un programma utente, prima di essere eseguito, deve passare attraverso diversi stadi in cui gli indirizzi possono esser rappresentati in modo differente.

- **Compilazione:** se in questa fase si sa dove il processo risiederà nella memoria, si può generare codice assoluto → se la locazione cambiasse in un momento successivo, sarebbe necessario ricompilare il codice.
- **Caricamento:** se in questa fase non si può sapere dove risiederà in memoria il processo, il compilatore deve generare codice rilocabile, ritardando l'associazione finale degli indirizzi alla fase del caricamento → se la locazione cambiasse, sarebbe sufficiente ricaricare il codice d'utente incorporando il valore modificato.
- **Esecuzione:** se un processo può essere spostato da un segmento di memoria ad un altro si deve ritardare l'associazione degli indirizzi fisici fino alla fase d'esecuzione. Sono necessarie specifiche caratteristiche dell'architettura.

SPAZI DI INDIRIZZI LOGICI E FISICI

Indirizzo logico: indirizzo generato dalla cpu
Indirizzo fisico: indirizzo visto dalla memoria, caricato nel registro dell'indirizzo di memoria (MAR)

In fase di compilazione e caricamento indirizzo fisico=indirizzo logico, mentre in fase d'esecuzione indirizzi fisici ≠ indirizzi logici (virtuali).

Spazio degli indirizzi logici: insieme di tutti gli indirizzi logici generati da un programma;
Spazio degli indirizzi fisici: insieme degli indirizzi fisici corrispondenti allo spazio degli indirizzi logici.
Memory Management Unit (MMU): dispositivo che associa gli indirizzi virtuali a quelli fisici in fase d'esecuzione.
Registro di rilocazione: registro di base a cui si somma l'indirizzo generato da un processo per ottenere l'indirizzo fisico.

Il programma non considera mai gli indirizzi fisici reali, ma li tratta semplicemente come numeri. Solo quando questi assumono un ruolo di indirizzo di memoria (una `load` o una `store` indiretta, per esempio), si riloca il numero in base al valore contenuto nel registro di rilocazione→la locazione finale di un riferimento a un indirizzo di memoria non è determinata finché non si compie effettivamente il riferimento.
Il programma utente tratta gli indirizzi logici, mentre l'architettura li converte in indirizzi fisici.

CARICAMENTO DINAMICO

Tutte le procedure si tengono in memoria secondaria in un formato di caricamento rilocabile e si carica una procedura solo quando viene richiamata.

1. si carica il programma principale in memoria
2. quando una procedura deve richiamarne un'altra, controlla che non sia già in memoria
    a. se lo è, la utilizza
    b. se non lo è, la carica in memoria e le dà il controllo

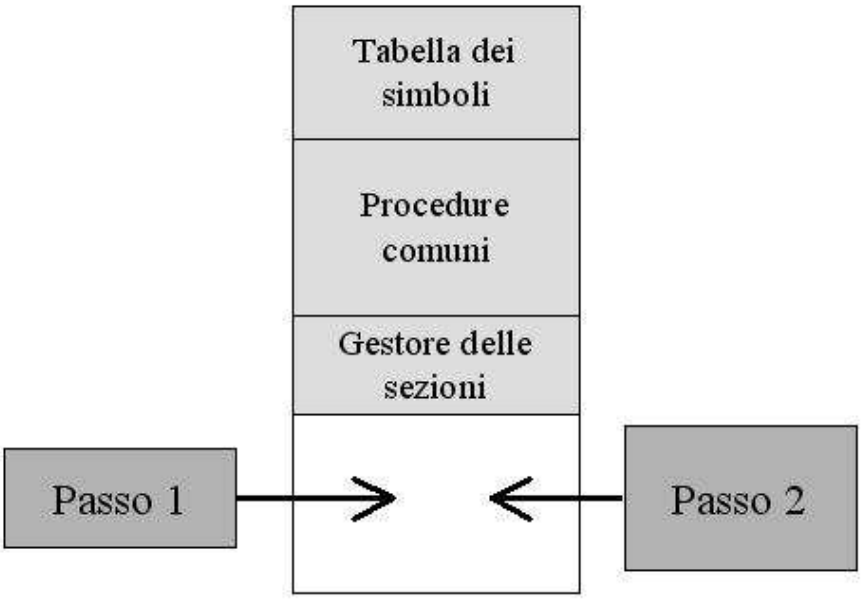
Quindi un programma può avere dimensioni elevate, anche se la parte effettivamente caricata in memoria è inferiore.
Questo metodo non richiede un intervento particolare del sistema operativo, se non il fornire librerie di procedure adatte.

COLLEGAMENTO DINAMICO E LIBRERIE CONDIVISE

Collegamento statico: le librerie di sistema del linguaggio sono trattate come qualsiasi altro modulo oggetto e sono combinate dal caricatore nell'immagine binaria del programma.
Collegamento dinamico: per ogni riferimento a una procedura di libreria s'inserisce all'interno dell'immagine eseguibile una piccola porzione di codice di riferimento (stub), che indica come localizzare la giusta procedura di libreria residente nella memoria o come caricare la libreria se la procedura non è già presente
Durante l'esecuzione, lo stub controlla se la procedura richiesta è già nella memoria, altrimenti provvede a caricarla; in ogni caso tale codice sostituisce se stesso con l'indirizzo della procedura, che viene poi eseguita → quando si raggiunge nuovamente quel segmento di codice, si esegue direttamente la procedura di libreria, senza costi aggiuntivi per il

collegamento dinamico. Tutti i processi eseguono la stessa copia del codice della libreria. Questo metodo si può estendere agli aggiornamenti delle librerie.
Librerie condivise: può succedere che esistano diverse versioni di una libreria, e che i programmi collegati prima dell'installazione della nuova versione debbano avvalersi della vecchia.

SOVRAPPOSIZIONE DI SEZIONI (Overlay)
Si basa sul principio di mantenere nella memoria soltanto le istruzioni e i dati che si usano con maggior frequenza. Quando son necessarie altre istruzioni, queste si caricano nello spazio precedentemente occupato dalle istruzioni che non sono più in uso.
Questa tecnica può essere realizzata direttamente dall'utente per mezzo di semplici strutture di file, copiandone il contenuto nella memoria e quindi trasferendo il controllo a quest'ultima per eseguire le istruzioni appena lette. Il sistema operativo si limita ad annotare la presenza di istruzioni di I/O supplementari.



AVVICENDAMENTO DEI PROCESSI
Avvicendamento dei processi (swapping): il gestore di memoria scarica dalla memoria il processo appena terminato (per esempio, in un RR) e carica un altro processo nello spazio di memoria appena liberato.

- Se l'associazione degli indirizzi logici agli indirizzi fisici della memoria si effettua nella fase di assemblaggio o di caricamento, il processo deve essere ricaricato nello spazio di memoria occupato in precedenza;
- Se l'associazione degli indirizzi logici agli indirizzi fisici della memoria di compie in fase d'esecuzione, un processo può esser riversato in uno spazio di memoria diverso, poiché gli indirizzi fisici si calcolano in fase d'esecuzione.

L'avvicendamento dei processi richiede una memoria ausiliaria, che deve essere:

- Veloce
- Abbastanza ampia da contenere le copie di tutte le immagini di memoria di tutti i processi utente e permettervi un accesso diretto.

Quando lo scheduler della cpu decide di eseguire un processo:

1. chiama il dispatcher, che controlla se il primo processo nella coda dei processi pronti è in memoria
    a. se non c'è e la memoria è piena
        i. scarica un processo dalla memoria
        ii. carica il processo richiesto
        iii. ricarica i registri
2. il controllo passa al processo richiesto.

Con questo modo il tempo di cambio di contesto è elevato.
Il tempo di trasferimento totale è direttamente proporzionale alla quantità di memoria interessata.
E' utile sapere quanta memoria è effettivamente usata da un processo utente e non solo quanto questo potrebbe teoricamente usarne, poiché in questo caso è necessario scaricare solo quanto effettivamente utilizzato, riducendo il tempo d'avvicendamento → l'utente deve tenere informato il sistema su tutte le modifiche apportate ai requisiti di

memoria; in un processo con requisiti di memoria dinamici deve impiegare chiamate di sistema (`request memory` e `release memory`) per informare il sistema operativo delle modifiche apportate alla memoria.
L'avvicendamento dei processi è soggetto a altri vincoli:

- per scaricare un processo dalla memoria è necessario che il processo sia completamente inattivo. Problema: I/O pendenti
  - non scaricare dalla memoria un processo con I/O pendenti
  - eseguire le operazioni di I/O solo in aree di memoria per l'I/O del sistema operativo

Generalmente si assegna l'area di avvicendamento in una porzione del disco separata da quella riservata al file system → velocità.

ASSEGNAZIONE CONTIGUA DELLA MEMORIA

La memoria centrale di solito si divide in due partizioni, una per il sistema operativo residente e una per i processi utenti. Il sistema operativo si colloca generalmente nella stessa parte del vettore delle interruzioni. Un metodo di assegnazione della memoria e quello di assegnare una sezione contigua di memoria ad ogni processo nella coda d'ingresso dei processi che attendono di essere caricati

Protezione della memoria
Il registro di rilocazione contiene il valore dell'indirizzo fisico minore, mentre il registro di limite contiene l'intervallo di indirizzi logici. Ogni indirizzo logico deve essere minore del contenuto del registro di limte; la MMU fa corrispondere dinamicamente l'indirizzo fisico a quello logico sommando a quest'ultimo il contenuto del registro di rilocazione. Il dispatcher confronta ogni indirizzo generato dalla cpu con i valori contenuti in questi registri, proteggendo il sistema operativo e i programmi/dati utenti da tentativi di modifiche da parte del processo in esecuzione.
Lo schema con registro di rilocazione permette al sistema operativo di cambiare dinamicamente le proprie dimensioni (codice transiente del sistema operativo: si inserisce solo se necessario).

Assegnazione della memoria
MTF
Si divide la memoria in partizioni di dimensione fissa, che può contenere esattamente un processo → il grado di multiprogrammazione dipende dal numero di partizioni
MVT
Il sistema operativo conserva una tabella nella quale sono indicate le partizioni di memoria disponibili e quelle occupate, e le relative dimensioni. Inizialmente tutta la memoria a disposizione viene vista come un unico blocco. Quando si carica un processo in memoria, si cerca un blocco di spazio sufficientemente grande per contenerlo, assegnandogli solo la parte di memoria di cui necessita. Generalmente è sempre presente un insieme di blocchi di diverse dimensioni liberi e sparsi per la memoria. Per assegnare un blocco ad un processo, si possono adottare diverse politiche:

- First-fit: si assegna il primo blocco abbastanza grande.
- Best-fit: si assegna il più piccolo blocco in grado di contenere il processo.
- Worst-fit: si assegna il blocco più grande.

Gli ultimi due metodi richiedono una ricerca in tutta la lista delle dimensioni dei blocchi disponibili; il *best-fit* e il *first fit* sono migliori poiché riducono il tempo e l'utilizzo della memoria.

Frammentazione esterna:quando si caricano e si rimuovono processi dalla memoria, si frammenta lo spazio libero nella memoria in tante piccole parti. Si ha questo tipo di frammentazione se lo spazio di memoria totale è sufficiente ma non contiguo.
Possibili soluzioni:

- compattare i vari blocchi, ma lo si può fare solo se la rilocazione degli indirizzi è dinamica (cioè avviene in fase d'esecuzione). Si può per esempio spostare tutti i processi in un'estremità della memoria, creando un blocco libero all'altra estremità --> metodo oneroso
- Consentire la non contiguità dello spazio degli indirizzi logici di un processo, permettendo di assegnare la memoria fisica ai processi ovunque essa sia disponibile.
  - Paginazione
  - Segmentazione

Frammentazione interna: si può suddividere la memoria fisica in blocchi di dimensione fissata, che costituiscono le unità di assegnazione --> la memoria richiesta può essere inferiore a quella assegnata, lo spazio che avanza si dice frammentato interiormente.

PAGINAZIONE

Si suddivide la memoria fisica in blocchi di memoria di dimensioni fisse (pagine fisiche o frame), e la memoria logica in blocchi di eguale dimensione (pagine logiche). Quando si deve eseguire un processo, si caricano le sue pagine nei

blocchi di memoria disponibili, prendendole dalla memoria ausiliaria, che è divisa in blocchi di dimensione fissa, uguale a quella dei blocchi di memoria. Ogni indirizzo generato dalla cpu è diviso in due parti: un numero di pagina e uno scostamento di pagina. Il numero di pagina serve come indice per la tabella delle pagine, che contiene l'indirizzo di base nella memoria fisica per ogni pagina. Questo indirizzo di base si appaia allo scostamento di pagina per definire l'indirizzo della memoria fisica, che s'invia all'unità di memoria.
La dimensione di una pagina è definita dall'architettura del calcolatore.
La paginazione è una forma di rilocazione dinamica: a ogni indirizzo logico l'architettura di paginazione fa corrispondere un indirizzo fisico. L'uso della tabella delle pagine è simile all'uso di una tabella di registri di base, uno per ciascun blocco di memoria.
Con questo metodo si elimina la frammentazione esterna: qualsiasi blocco di memoria libero si può assegnare a un processo che ne abbia bisogno. Si ha però la frammentazione interna poiché i blocchi di memoria si assegnano come unità.
Quando si deve eseguire un processo, si controlla di avere abbastanza blocchi liberi, glieli si assegna e si carica la prima pagina, inserendo il numero del blocco di memoria nella tabella delle pagine relativa al processo in questione.
Il programma utente 'vede' la memoria come un unico spazio contiguo, mentre in realtà questo è sparso nella memoria fisica, insieme ad altri programmi. Tutto questo lo si deve al sistema operativo che gestisce varie informazioni:

- Tabella dei blocchi di memoria: contiene un elemento per ogni blocco di memoria, indicante se è libero o assegnato e , se è assegnato, a quale pagina di quale processo.
- Tabella delle pagine di ciascun processo (+ valore del program counter e dei registri), che usa per tradurre gli indirizzi logici in fisici ogni volta che gli è richiesto esplicitamente. Il dispatcher usa questa tabella per impostare l'architettura di paginazione ogni volta che un processo sta per essere assegnato alla cpu.

La paginazione aumenta la durata dei cambi di contesto.

Esistono vari metodi per memorizzare la tabella delle pagine.

1. usando uno specifico insieme di registri veloci. Le istruzioni di caricamento e scaricamento di questi registri sono privilegiate, quindi vi può operare solo il sistema operativo. Questo metodo si può usare solo se la tabella delle pagine è di dimensioni ridotte.
2. si mantiene la tabella delle pagine in memoria centrale e un registro di base della tabella delle pagine (page-table base register – PTRB) punta alla tabella stessa. Il cambio delle tabelle delle pagine richiede soltanto di modificare questo registro, riducendo il tempo dei cambi di contesto. Con questo metodo però l'accesso alla memoria è rallentato.
   Una soluzione consiste nell'impiegare una piccola cache (TLB) in cui ogni suo elemento è costituito da due parti: una chiave e un valore. Il TLB contiene una piccola parte degli elementi della tabella delle pagine; quando la cpu genera un indirizzo logico, si presenta il suo numero di pagina al TLB, se tale numero è presente, il corrispondente blocco di memoria è immediatamente disponibile e si usa per accedere alla memoria; altrimenti si ha un insuccesso del TLB e si deve consultare la tabella delle pagine nella memoria.
   Quando il TLB è pieno, il sistema operativo può decidere si sostituirne un elemento, eliminando, per esempio scartando l'elemento usato meno recentemente o usando un criterio di scelta casuale. Alcuni elementi possono essere vincolati, per esempio elementi per il codice del nucleo.
   Alcuni TLB memorizzano gli identificatori dello spazio di indirizzi (ASID) in ciascun elemento del TLB --> ciò garantisce la protezione del processo corrispondente.

PROTEZIONE
In un ambiente paginato, la protezione della memoria è assicurata dai bit di protezione associati a ogni blocco di memoria --> normalmente tali bit si trovano nella tabella delle pagine e possono determinare se una pagina si può leggere e scrivere o solamente leggere.
Di solito si associa un ulteriore bit (bit di validità) a ciascun elemento della tabella delle pagine.

- valido: la pagina corrispondente è nello spazio d'indirizzi logici del processo
- non valido: la pagina non è nello spazio logico di indirizzi del processo-->indirizzo illegale.

Capita raramente che un processo utilizzi tutto il suo intervallo di indirizzi, perciò in alcune architetture ci sono dei registri di lunghezza della tabella delle pagine (PTLR) che indicano la dimensione della tabella e controlla che ogni indirizzo logico sia valido.
STRUTTURA DELLA TABELLA DELLE PAGINE

Gerarchica
In un ambiente che dispone di uno spazio di indirizzi logici molto grande, la tabella delle pagine è altrettanto grande, perciò è consigliabile evitare di collocare questa tabella in modo contiguo nella memoria centrale, suddividendola in parti più piccole.
Usando un algoritmo di paginazione a due livelli, si può paginare la stessa tabelle delle pagine. L'indirizzo logico è quindi diviso in:

- numero di pagina
  - indice della tabella delle pagine di primo livello (tabella esterna delle pagine)

    - scostamento all'interno della pagina indicata dalla tabella esterna delle pagine
  - scostamento

Questo metodo si chiama anche tabella delle pagine ad associazione diretta.

Di tipo hash

L'argomento della funzione di hash è il numero della pagina virtuale.
Per la gestione delle collisioni, ogni elemento della tabella hash contiene una lista concatenata di elementi che la funzione hash fa corrispondere alla stessa locazione. Ciascun elemento è composta da tre campi:

a) numero della pagina virtuale
b) indirizzo del blocco della pagina fisica che corrisponde alla pagina virtuale
c) puntatore al successivo elemento della lista

L'algoritmo funziona come segue:

1. si applica la funzione di hash al numero della pagina contenuto nell'indirizzo virtuale, identificando un elemento nella tabella
2. si confronta il numero di pagina virtuale con l'elemento contenuto in a) del primo elemento della lista concatenata corrispondente
   - **i.** se i valori coincidono, si usa il campo b) per generare l'indirizzo fisico
   - **ii.** altrimenti si esamina il seguente elemento della lista

Questo metodo è particolarmente utile per spazi d'indirizzo sparsi.
Una variante è la tabella delle pagine a gruppi (clustered table), in cui ciascun elemento della tabella delle pagine contiene i riferimenti alle pagine fisiche corrispondenti a un gruppo di pagine virtuali contigue--> si risparmia spazio e si sfrutta il fatto che i programmi solitamente non sono completamente sparsi, ma distribuiti a gruppi.

Invertita

Questa tabella ha un elemento per ogni pagina reale, ciascun elemento è quindi costituito dall'indirizzo virtuale della pagina memorizzata in quella reale locazione di memoria, con informazioni relative al processo che possiede tale pagina → una tabella delle pagine di questo tipo ha un elemento solo per ciascuna pagina di memoria fisica.
Ciascun indirizzo virtuale è del tipo <id-processo,numero di pagina, scostamento>.
Ogni elemento della pagina virtuale è del tipo <id-processo, numero di pagina>; l'id-processo è l'identificatore dello spazio d'indirizzi.

1. la cpu genera un indirizzo logico
2. si confronta la coppia <id.processo,numero di pagina>, cercando una corrispondenza nella tabella delle pagine invertita
   a. se non si trova, si è tentato un accesso illegale → errore
   b. se si trova l'elemento, si forma un indirizzo fisico dato da <numero elemento, scostamento>

Questa struttura riduce la quantità di memoria necessaria per memorizzare ogni tabella delle pagine, ma aumenta il tempo di ricerca, poiché le ricerche si fanno per indirizzi virtuali, mentre la tabella è ordinata per indirizzi fisici→ perciò occorre scandire l'intera tabella finché non si trova l'elemento.
Si può migliorare questo aspetto utilizzando congiuntamente una tabella di hash e un TLB.

PAGINE CONDIVISE

Un vantaggio della paginazione consiste nella possibilità di condividere codice comune.
Codice rientrante (codice puro): è un codice non automodificante, non cambia durante l'esecuzione → due processi possono eseguire quindi lo stesso codice nello stesso momento.
Nei sistemi che fanno uso delle tabelle delle pagine invertite, la condivisione risulta difficile, poiché la memoria condivisa si realizza facendo corrispondere più indirizzi virtuali allo stesso indirizzo fisico, mentre in suddette tabelle è presente un solo elemento di pagina virtuale per ogni pagina fisica.

SEGMENTAZIONE

E' uno schema di gestione della memoria che consente di gestire la visione della memoria da parte dell'utente. Uno spazio di indirizzi logici è una raccolta di segmenti, ciascuno dei quali ha un nome e una lunghezza. Gli indirizzi specificano sia il nome sia lo scostamento all'interno di un segmento, quindi l'utente fornisce ogni indirizzo come una coppia ordinata di valori → un indirizzo logico è una coppia del tipo <numero di segmento,scostamento>.
Normalmente il programma utente è stato compilato, e il compilatore struttura automaticamente i segmenti secondo il programma sorgente. Esempi di segmenti possono essere

- variabili locali
- pila per le chiamate di procedure
- codice di ogni procedura e funzione
- variabili locali di ciascuna procedura e funzione

Architettura

Occorre tradurre gli indirizzi bidimensionali definiti dall'utente negli indirizzi fisici unidimensionali. Per far ciò si utilizza una tabella dei segmenti, formata da elementi del tipo <base del segmento, limite del segmento >. La base contiene l'indirizzo fisico iniziale della memoria al quale il segmento risiede, il limite contiene la lunghezza di tale segmento.
Indirizzo logico <numero segmento, scostamento >. Il numero segmento si usa come indice della tabella dei segmenti, lo scostamento deve essere compreso tra 0 e il limite del segmento (altrimenti →errore)
Si somma lo scostamento alla base e si produce così l'indirizzo fisico desiderato.

Protezione e condivisione
L'architettura di traduzione degli indirizzi della memoria controlla i bit di protezione associati ad ogni elemento, in modo da impedire accessi illegali in memoria.
Ogni processo ha una tabella di segmenti, associata al proprio PCB, che il dispatcher usa per definire la tabella fisica dei segmenti quando assegna la cpu al processi. I segmenti si condividono se gli elementi delle tabelle dei segmenti di due processi diversi puntano alle stesse locazioni fisiche. Poiché la condivisione avviene al livello dei segmenti, si può condividere qualsiasi informazione definita come segmento, e siccome si possono condividere parecchi segmenti, si può condividere anche un intero programma composto da svariati segmenti, oppure solo una sua parte.

I segmenti di dati in sola lettura, che non contengono puntatori fisici, si possono condividere assegnando a ciascun segmento un valore diverso, così come nei segmenti che non fanno riferimenti a loro stessi, si possono condividere senza problemi.
Problema: come numerare un segmento che fa riferimento a se stesso, quando più utenti vi hanno accesso? (si necessiterebbe di un unico valore che individui quel segmento)→ facendo riferimento in modo indiretto, per esempio uno scostamento al valore corrente del program counter..

Frammentazione
La segmentazione può causare la frammentazione esterna (non interna perché i segmenti hanno lunghezza variabile).
La segmentazione è un algoritmo di rilocazione dinamica, grazie al quale la memoria di può compattare.

SEGMENTAZIONE CON PAGINAZIONE

Si stabilisce un numero massimo di segmenti per processo, la dimensione massima di ogni segmento e il numero di pagine in esso contenute (le pagine hanno dimensione fissa).
Lo spazio di indirizzi logici di un processo è diviso in due partizioni:

- la prima contiene segmenti riservati al processo e le informazioni riguardanti questa partizione (compresi i registri di base e di limite) sono contenute nella tabella locale dei descrittori (LDT)
- la seconda contiene segmenti condivisi fra tutti i processi e le informazioni relative a questa partizione (compresi i registri di base e di limite) sono contenute nella tabella globale dei descrittori (GDT)

Un indirizzo logico è strutturato come segue:

- selettore
  - numero di segmento
  - bit che indica se appartiene alla LDT o alla GDT
  - informazioni relative alla protezione
- scostamento: posizione del byte all'interno del segmento

1. la cpu genera un indirizzo logico
2. per mezzo del 'bit apposito' vado nella LDT o nella GDT e vado nell'elemento indicato dal numero di segmento. Tramite i valori dei registri di base e di limite di tale segmento si genera un indirizzo lineare

Un indirizzo lineare è strutturato come segue:

- numero di pagina (utilizzo una schema di paginazione a due livelli, quindi)
  - directory delle pagine
  - Pagina
- Scostamento di pagina

3. Traduzione dell'indirizzo lineare in indirizzo fisico
   a. Nella directory delle pagine cerco la tabella delle pagine che mi interessa
   b. Trovata la tabella 'uso' il numero di pagina → ho così individuato la pagina
   c. Uso lo scostamento (dell'indirizzo lineare) per formare l'indirizzo fisico.

[vedi immagine p.319]

Capitolo 10

In molti casi non è necessario avere in memoria l'intero programma, per esempio il codice per la gestione degli errori, parti di tabelle e matrici e altre sezioni. In altri casi, varie sezioni non devono essere eseguite contemporaneamente (overlay).

Memoria virtuale: tecnica che permette di eseguire processi che possono anche non essere completamente nella memoria.

- Astrae la memoria centrale in un vettore molto grande e uniforme → separa la memoria fisica da quella logica, permettendo di sfruttare uno spazio di indirizzi molto grande.
- Consente ai processi di condividere facilmente files e spazi d'indirizzi.
- Permette di eseguire più programmi contemporaneamente→ aumento utilizzo della cpu.
- Per caricare (o scaricare) ogni programma utente nella memoria sono necessarie meno operazioni di I/O → più veloce.

Si può realizzare la memoria virtuale nelle forme:

- Paginazione su richiesta
- Segmentazione su richiesta

PAGINAZIONE SU RICHIESTA

I processi risiedono in memoria secondaria e per eseguirli occorre caricarli in memoria centrale. Anziché caricare il processo per intero, si può eseguire un criterio di avvicendamento (lazy swapping) e caricare solo le pagine necessarie.
Paginatore: modulo del sistema operativo che si occupa della sostituzione delle pagine.
Per distinguere se una pagina è in memoria o no, l'architettura deve fornire un bit (impostato a 'valido' indica che la pagina è valida ed è in memoria, altrimenti potrebbe essere non valida, o valida ma non in memoria). Per quel che riguarda la tabella delle pagine, una pagina valida viene segnata normalmente, mentre una non valida ha il bit settato a 'non valido' oppure contiene l'indirizzo che consente di trovare la pagina nel disco.

Durante l'esecuzione un processo può:

- Accedere a pagine residenti in memoria → esecuzione normale
- Tentare di accedere ad una pagina non caricata in memoria →eccezione di pagina mancante (page fault trap)

1. segnale di eccezione al sistema operativo;
2. salvataggio dei registri d'utente e dello stato del processo;
3. determinazione della natura di eccezione di pagina mancante del segnale di eccezione;
4. controllo della correttezza del riferimento alla pagina e determinazione della locazione della pagina nel disco;
5. lettura dal disco e trasferimento in un blocco di memoria libero:
    a. attesa nella coda relativa a questo dispositivo fino a che la richiesta di lettura non è servita,
    b. attesa del tempo di posizionamento e latenza del dispositivo,
    c. inizio del trasferimento della pagina in un blocco di memoria libero;
6. durante l'attesa, assegnazione della cpu a un altro processo utente
7. segnale d'interruzione emesso dal controllore del disco (I/O completato)
8. salvataggio dei registri e dello stato dell'altro processo utente
9. determinazione della provenienza dal disco dell'interruzione
10. aggiornamento della tabella delle pagine e di altre tabelle per segnalare che la pagina richiesta è attualmente presente nella memoria
11. attesa che la CPU sia nuovamente assegnata a questo processo
12. recupero dei registri d'utente, dello stato del processo e della nuova tabella delle pagine, quindi ripresa dell'istruzione interrotta.

Paginazione su richiesta pura: quando si avvia un processo senza pagine in memoria → una pagina non si trasferisce in memoria finchè non è richiesta.

In teoria, un'istruzione potrebbe causare più eccezioni di pagina mancante, in pratica questo non avviene spesso grazie alla località dei riferimenti.

Per realizzare la paginazione su richiesta, l'architettura deve fornire:
- Tabella delle pagine → deve avere il bit di validità
- Memoria ausiliaria → conserva le pagine non caricate in memoria. La sezione del disco che si occupa di ciò è l'area di avvicendamento (swap space)

Un'assenza di pagina può modificare parecchie locazioni e questo potrebbe causare problemi nel momento in cui si deve riprendere l'esecuzione dell'istruzione.
- Al verificarsi di un'assenza di pagina, si salvano tutti i vecchi valori presenti nella memoria prima che sia emesso il segnale di eccezione di pagina mancante→ in questo modo si riporta la memoria allo stato in cui si trovava prima che l'istruzione fosse avviata, perciò si può ripetere la sua esecuzione
- Si potrebbe creare un registro di stato per memorizzare il numero di registro e l'entità della modifica per ogni registro modificato durante l'esecuzione di un'istruzione → questo permette al sistema operativo di annullare gli effetti di un'istruzione eseguita solo in parte, che ha causato un'eccezione di pagina mancante.

Il sistema di paginazione si colloca tra la CPU e la memoria di un calcolatore e deve esser completamente trasparente al processo utente.

PRESTAZIONI

Tempo di accesso effettivo = (1-p) x ma + p x tempo di gestione dell'assenza di pagina
p= probabilità che si verifichi un'assenza di pagina
ma= tempo di accesso alla memoria
1-p = probabilità che non si verifichi un'assenza di pagina

Il tempo di accesso effettivo è direttamente proporzionale alla frequenza delle assenze di pagine, per questo è importante tener bassa la frequenza di quest'ultime. Un altro aspetto importante per ciò che riguarda le prestazioni della paginazione su richiesta è la gestione e l'uso generale dell'area d'avvicendamento.
- Si può copiare tutta l'immagine di un file nell'area di avvicendamento all'avvio del processo e da lì eseguire la paginazione su richiesta
- Si possono chiedere tutte le pagine al file system, ma scrivere le pagine nell'area di avvicendamento al momento della sostituzione → si leggono dal file system solo le pagine necessarie, ma tutta la paginazione successiva è fatta dall'area d'avvicendamento

CREAZIONE DEI PROCESSI

Il meccanismo della memoria virtuale ha reso possibili due tecniche per migliorare le prestazioni relative alla creazione ed esecuzione dei processi.

- **Copiatura su scrittura** (copy-on-write)

E' una tecnica che si fonda sulla condivisione iniziale delle pagine da parte dei processi genitori e dei processi figli. Le pagine condivise si contrassegnano come pagine da copiare su scrittura, a significare che, se un processo (genitore o figlio) scrive su una pagina condivisa, il sistema deve creare una copia di tale pagina. In questo modo si copiano solo le pagine modificate da un processo, mentre tutte le altre (incluse quelle non modificabili) rimangono condivise. Per effettuare la copiatura di una pagina è necessaria una pagina libera. A questo scopo, molti sistemi operativi forniscono un pool di pagine libere e l'assegnazione di queste ultime avviene secondo la tecnica dell'azzeramento su richiesta (zero-fill-on-demand) → si riempiono di zeri le pagine prima dell'assegnazione, cancellandone tutto il contenuto precedente.

- **Associazione dei file alla memoria**

E' una tecnica che permette a una parte dello spazio di indirizzi virtuali di essere associata logicamente a un file. Si fa corrispondere un blocco di disco a una pagina (o più) della memoria fisica. Quando si verifica un'assenza di pagina, si carica in una pagina fisica una porzione di file della dimensione di una pagina, leggendola dal file system. Le operazioni successive (read, write) si gestiscono come normali accessi alla memoria. Poiché le modifiche avvengono in memoria centrale, si potrebbe aggiornare la memoria secondaria periodicamente e, naturalmente, alla chiusura del file (che comporta anche la rimozione del processo dalla memoria virtuale). Si può permettere l'associazione dello stesso file alla memoria virtuale di più processi, allo scopo di condividere i dati. Le operazioni di scrittura fatte da uno qualunque di questi processi modificano i dati presenti nella memoria virtuale e sono visibili a tutti i processi coinvolti nella condivisione.

SOSTITUZIONE DELLE PAGINE

Aumentando il grado di multiprogrammazione, si sovrassegna la memoria, ossia si può arrivare al punto di non avere più blocchi liberi da assegnare. Una possibile soluzione (anche se non ottimale) è terminare un processo utente, scaricarlo completamente dalla memoria e così ridurre il grado di multiprogrammazione, oppure si può ricorrere alla sostituzione delle pagine → se non c'è alcun blocco libero, se ne libera uno inutilizzato.

1. s'individua la locazione nel disco della pagina richiesta;
2. si cerca un blocco di memoria libero;
    a. se esiste, lo si usa,
    b. altrimenti si impiega un algoritmo di sostituzione delle pagine per scegliere un blocco di memoria 'vittima',
    c. si scrive la pagina 'vittima' nel disco; si modificano nel modo che ne consegue le tabelle delle pagine e dei blocchi di memoria;
3. si scrive la pagina richiesta nel blocco di memoria appena liberato; si modificano le tabelle delle pagine e dei blocchi di memoria;
4. si riavvia il processo utente.

Questo raddoppia il tempo di servizio dell'eccezione di pagina mancante. Per ridurre questo sovraccarico si può utilizzare un bit di modifica (dirty bit) che indica se la pagina è stata modificata da quando è stata caricata dalla memoria secondaria. Se il bit è

- Attivo→ pagina modificata: deve essere riscritta in memoria secondaria se viene scelta come 'vittima' della sostituzione;
- Non attivo → pagina non modificata: è uguale a quella in memoria secondaria e non necessita quindi di esser riscritta.

Per realizzare la paginazione su richiesta bisogna sviluppare un algoritmo di sostituzione delle pagine e uno per l'assegnazione dei blocchi di memoria.

ALGORITMI PER LA SOSTITUZIONE DELLE PAGINE

Un criterio per valutare la bontà di un algoritmo di questo tipo è la frequenza delle assenze di pagina, che è inversamente proporzionale al numero di blocchi di memoria disponibili.

- **Secondo l'ordine d'arrivo** (FIFO)

Si associa ad ogni pagina l'istante di tempo in cui quella pagina è stata portata nella memoria → quando si deve sostituire una pagina, si sceglie come 'vittima' quella che è da più tempo in memoria. Si può anche evitare di registrare l'istante in cui si carica una pagina in memoria, è sufficiente strutturare le pagine presenti in memoria come una coda FIFO. Nonostante sia facile da implementare, questo algoritmo non è ottimale, in quanto la 'vittima' potrebbe essere una pagina ancora necessaria a breve.

Anomalia di Belady: con alcuni algoritmi di sostituzione delle pagine la frequenza delle assenze di pagina può *aumentare* con l'aumentare del numero di blocchi di memoria assegnati.

- **Ottimale** (OPT o MIN)

Si sostituisce la pagina che non si userà per il periodo di tempo più lungo. Sfortunatamente è di difficile realizzazione poiché richiede la conoscenza della futura successione di riferimenti→ per questo si utilizza soprattutto come termine di paragone per gli altri algoritmi. Naturalmente non presenta l'anomalia di Belady.

- **Pagine usate meno recentemente** (LRU –least recently used)

Questo algoritmo associa a ogni pagina l'istante in cui è stata usata per l'ultima volta. La 'vittima' è la pagina che non è stata usata per il periodo più lungo. Il problema principale per la realizzazione di questo algoritmo è per l'appunto stabilire un ordinamento secondo il momento dell'ultimo uso.

- Contatori: si utilizza un registro contatore il cui contenuto viene copiato per ogni riferimento alla memoria, in modo da avere per ogni pagina il momento in cui è stato fatto l'ultimo riferimento. La 'vittima' è quella con il valore associato minimo. Qusto meccanismo comporta
  i. Ricerca all'interno della tabella delle pagine per individuare quella usata meno recentemente;
  ii. Scrittura nella memoria (il campo del momento d'uso) per ogni accesso in memoria;
  iii. Mantenimento dei riferimenti temporali anche quando le tabelle vengono modificate dallo scheduling della CPU;
  iv. Attenzione al superamento della capacità del contatore (overflow)

- Pila: si mettono i numeri di pagina in una pila e ogni volta che si fa un riferimento, la pagina in questione viene spostata in cima. In questo modo la pagina in fondo alla pila sarà la Lru. Questa struttura viene realizzata con una lista doppiamente concatenata poiché si devono spostare elementi che stanno 'in mezzo' alla pila → gli aggiornamenti sono un po' costosi, ma non bisogna effettuare una ricerca poiché l'elemento di coda punta alla LRU. Questo algoritmo, come la classe di algoritmi di sostituzione delle pagine detti 'a pila', non presenta l'anomalia di Belady.

- **Per approssimazione a LRU**

Molti sistemi forniscono un bit di riferimento associato a ciascun elemento della tabella delle pagine. Questo bit è impostato direttamente dall'architettura del sistema ogni volta che si effettua un riferimento. In questo modo è possibile stabilire quali pagine sono state usate, ma non in che ordine.

- Algoritmo con bit supplementari di riferimento: si può conservare in una tabella nella memoria una serie di bit per ogni pagina. A intervalli regolari il sistema operativo sposta il bit di riferimento per ciascuna pagina nel bit più significativo, traslando gli altri bit a destra di un bit e scartando quello meno significativo. Interpretando questa successione di bit come interi senza segno, la pagina a cui è associato il numero minore è la pagina LRU.
- Algoritmo con seconda chance: è un algoritmo per la sostituzione di tipo FIFO. I bit di riferimento delle pagine vengono strutturati in una coda circolare. Un puntatore scandisce la coda, mettendo a zero i bit che trova a 1 (dando una seconda chance) e passando all'elemento successivo finchè non incontra uno zero. Questa è la pagina 'vittima'; la nuova pagina prenderà il suo posto nella coda.
- Algoritmo con seconda chance migliorato: si possono considerare un bit di riferimento e uno di modifica come una coppia ordinata
  1. (0,0) né recentemente usato, né modificato →migliore pagina da sostituire
  2. (0,1) non usato recentemente, ma modificato → non è una buona 'vittima' in quanto prima di essere sostituita deve esser riscritta in memoria secondaria
  3. (1,0) usata recentemente, ma non modificato → la pagina potrebbe esser ancora richiesta a breve
  4. (1,1) usata recentemente e modificata →oltre alla possibilità che sia presto richiesta, deve anche esser riscritta in memoria secondaria.

  Si usa lo stesso meccanismo dell'algoritmo con 2° chance, però anziché controllare il solo bit di riferimento, si controlla la classe a cui la pagina appartiene, scegliendo la pagina di classe minima → bisogna scandire più volte la coda.

- **Basato su conteggio**: contando il numero di riferimenti che sono stati fatti a ciascuna pagina
  - Algoritmo di sostituzione delle pagine meno frequentemente usate (LFU- least frequently used): si sostituisce la pagina con il conteggio più basso, basandosi sul principio che non è usata attivamente;
  - Algoritmo di sostituzione delle pagine più frequentemente usate (MFU- Most frequently used): si sostituisce la pagina con il conteggio più alto, basandosi sul fatto che la pagina appena inserita ha il conteggio più basso e non è ancora stata usata.

  LFU e MFU, oltre ad essere costosi, non approssimano bene la sostituzione OPT.

- **Con memorizzazione transitoria della pagine**

Generalmente i sistemi hanno un gruppo di blocchi liberi. Quando si verifica un'assenza di pagina, si sceglie un blocco di memoria vittima, lo si mette in uno di questi blocchi e si trasferisce la pagina richiesta. In questo modo non bisogna attendere che la 'vittima' venga traferita in memoria secondaria. Analogamente, si può conservare un elenco di pagine modificate e sfruttare i periodi in cui il paginature è inattivo, copiando le pagine modificate in memoria secondaria e reimpostando il loro bit di modifica.

ASSEGNAZIONE DEI BLOCCHI DI MEMORIA

- Non si possono assegnare più blocchi di memoria di quelli disponibili, sempre che non vi sia condivisione delle pagine;
- È necessario assegnare almeno un numero minimo di blocchi di memoria → questo numero è definito dalla struttura della serie di istruzioni di macchina, poiché quando si verifica un'assenza di pagina si deve poter salvare la situazione dell'istruzione per poi riavviarla;
- Il numero massimo di blocchi è definito dalla quantità di memoria fisica disponibile.

Assegnazione uniforme: se ho m blocchi e n processi, un'assegnazione di questo tipo dà m/n blocchi a ciascun processo.
Assegnazione proporzionale: si assegna la memoria disponibile a seconda della dimensione del processo [s_i= dimensione memoria virtuale di p_i; $S=\sum s_i$ → a pi si assegnerà (s_i/S) x m]
Entrambi questi metodi di assegnazione variano a seconda del grado di multiprogrammazione. Per affrontare il problema delle priorità si potrebbe adottare un assegnamento proporzionale che tenga conto sia della dimensione, sia del grado di priorità del processo.

Sostituzione globale: un processo può scegliere un blocco di memoria per la sostituzione dall'insieme di tutti i blocchi,

anche quelli di altri processi → il numero di blocchi assegnati a ogni processo può cambiare; inoltre un processo non

può controllare la propria frequenza di assenze di pagina poiché influenzato da altri processi.

Sostituzione locale: ogni processo può scegliere un blocco di memoria solo dal suo insieme di blocchi di memoria. Il numero di blocchi per processo non cambia, però può limitare molto un processo→ la sostituzione globale è quindi la più usata.

PAGINAZIONE DEGENERE (thrashing)

Si verifica quando si spende più tempo per la paginazione che per l'esecuzione dei processi.
Cause:
Si usa un algoritmo di sostituzione delle pagine globale→ aumentando il grado di multiprogrammazione aumenta anche l'utilizzo della CPU, fino a raggiungere il massimo. Se qa questo punto si aumenta ulteriormente il grado di multiprogrammazione, l'attività di paginazione degenera e fa crollare l'utilizzo della CPU. Per ri-aumentarne l'utilizzo occorre ridurre il grado di multiprogrammazione. Questo problema si può limitare utilizzando un algoritmo di sostituzione locale o di sostituzione per priorità. In queste soluzioni si rischia di aumentare il tempo di servizio di un'eccezione di pagina mancante → per evitarlo occorre fornire a un processo tutti i blocchi di memoria di cui necessita.
Località: insieme di pagine usate attivamente da un programma.

- **Modello di località**: un processo, durante la sua esecuzione, si sposta di località in località. Un programma è formato da parecchie località diverse, che si possono sovrapporre. Questo principio sta alla base del caching.
- **Modello dell'insieme di lavoro**: si tiene conto di una finestra di lavoro(δ). L'insieme di lavoro è l'insieme delle pagine nei più recenti δ riferimenti. Se una pagina è in uso attivo si trova nell'insieme di lavoro; se non è più usata esce dall'insieme di lavoro δ unità di tempo dopo il suo ultimo riferimento. La somma della dimensione dell'insieme di lavoro di ogni processo deve essere uguale alla richiesta totale di blocchi. Se questi sono superiori al numero totale di blocchi liberi, l'attività di paginazione degenera.

Il sistema operativo controlla l'insieme di lavoro di ogni processo e assegna opportunamente un numero di blocchi

di memoria:

a) Se i blocchi di memoria ancora liberi sono in numero sufficiente, si può iniziare un nuovo processo;

b) Se la somma delle dimensioni degli insiemi di lavoro aumenta, superando il numero totale di blocchi disponibili, il sistema operativo individua un processo da sospendere, scrivendone le pagine in memoria secondaria e assegnando i suoi blocchi ad altri processi.

La finestra di lavoro è *dinamica*

- **Frequenza delle assenze di pagina** (PFF- page fault frequency)

Per evitare l'attività di paginazione degenere si può controllare la PFF. Se la PFF di un processo è elevata, sarà opportuno assegnargli più blocchi di memoria, viceversa bisognerà togliergliene per darli ad altri processi. A questo scopo si fissano un limite superiore ed uno inferiore alla frequenza delle assenze delle pagine (se supera il limite superiore → al processo vengono assegnati altri blocchi; se scende dal livello inferiore → si riduce il numero di blocchi assegnati a quel processo)

CONSIDERAZIONI

Prepaginazione

In un sistema di paginazione su richiesta, all'avvio di un processo si verifica un alto numero di assenze di pagina. La prepaginazione consiste nel caricare tutte le pagine richieste in un'unica soluzione, per prevenire l'elevato livello di paginazione iniziale. Bisogna valutare se il costo della prepaginazione è inferiore al costo del servizio delle eccezioni di pagina mancante corrispondenti.

Dimensione delle pagine

Grandi

- Tabelle delle pagine più piccole
- Si riduce il tempo di I/O
- Si riduce il numero di assenze di pagina

Piccole

- La memoria viene utilizzata meglio
- Si riduce l'I/O totale poiché si migliora la località del codice
- Migliore risoluzione

Portata del TLB

Tasso di successi (hit ratio) di un TLB = percentuale di traduzione di indirizzi virtuali risolta dal TLb anziché dalla tabella delle pagine.

Portata del TLB (TLB reach)= quantità di memoria accessibile dal TLB (numero di elementi x la dimensione delle pagine)

Aumentare la portata del TLB aumenta anche il tasso di successi. Si può aumentare la portata:

- Aumentando il numero di elementi del Tlb,
- Aumentando la dimensione delle pagine→ si rischia una maggior frammentazione
- Impiegare più dimensioni delle pagine → richiede che il TLB sia gestito dal sistema operativo e non dall'architettura

Tabella delle pagine invertita

Questa tabella non contiene informazioni complete sullo spazio di indirizzi logici di un processo che sono richieste se una pagina alla quale si è fatto riferimento risultasse mancante → si necessita quindi una tabella esterna per ciascun processo, che contiene la locazione di ciascuna pagina virtuale. Queste tabelle sono paginate dentro e fuori la memoria secondo necessità. Il sistema operativo deve far particolare attenzione quando si fanno riferimenti a una stessa pagina dello stesso gestore della memoria virtuale: questa potrebbe venire richiesta proprio quando viene caricata per opera di un altro processo (backing store)

Struttura dei programmi

- Un'attenta scelta della struttura di dati e delle strutture di programmazione può aumentare la località → riduce la frequenza di assenza di pagine e il numero di pagine dell'insieme di lavoro (es: pila → buona località, tabella hash → pessima località). Altri fattori sono la rapidità di ricerca, il numero totale di riferimenti e di pagine coinvolte.

- Separare codice e dati migliora la paginazione in quanto il codice non si modifica e quindi le rispettive pagine non devono essere riscritte in memoria secondaria se non contengono dati.
- Anche il linguaggio di programmazione usato influisce sulla paginazione.

Vincoli di I/O

Se si usa la paginazione su richiesta, talvolta occorre vincolare alcune pagine alla memoria → per es. quando si esegue l'I/O verso/dalla memoria d'utente (virtuale). A ogni blocco di memoria si associa un bit di vincolo; se tale bit è attivato, la pagina contenuta non può essere selezionata per la sostituzione. Spesso il nucleo del sistema operativo è vincolato alla memoria. Si può usare anche per evitare che venga sostituita una pagina appena caricata.

Elaborazione in tempo reale

I sistemi in tempo reale non sono dotati di memoria virtuale poiché questa, pur aumentando la produttività del sistema,

introduce ritardi inattesi, causati dalle assenze di pagina e dal trasferimento di dati dalla memoria ai dischi e viceversa.

Capitolo 11

Il FILE SYSTEM è l'aspetto più visibile agli utenti del sistema operativo e fornisce il meccanismo per la registrazione e l'accesso in linea a dati e programmi. Il file system consiste in:

- Un insieme di file → dati
- Struttura di directory → organizza i file e fornisce informazioni su di loro
- Partizioni (minisichi, volumi)→ presenti in alcuni sistemi, separano grandi gruppi di directory.

FILE: insieme di informazioni correlate e registrate nella memoria secondaria, cui è stato assegnato un nome. Dal punto di vista dell'utente, un file è la più piccola porzione di memoria secondaria logica. I file offrono una visione uniforme della memorizzazione delle informazioni (tipo di dato astratto)

ATTRIBUTI

- **Nome** → nome simbolico umanamente comprensibile associato al file;
- **Identificatore**→ etichetta unica che identifica il file all'interno del file system;
- **Tipo**→ informazione necessaria ai sistemi che gestiscono più tipi di file;
- **Locazione**→ puntatore al dispositivo e alla locazione del file in tale dispositivo;
- **Dimensione**→ dimensione corrente ed eventualmente anche la massima consentita;
- **Protezione**→ informazioni di controllo degli accessi (lettura/scrittura/esecuzione);
- **Ora, data e identificazione dell'utente**→ informazioni relative alla creazione,modifica e ultimo uso.

Queste informazioni sono conservate nella struttura di directory in memoria secondaria. Un elemento di directory è un nome di file e il suo identificativo, che individua a sua volta gli attributi del file.

OPERAZIONI

- **Creazione**
  a) Trovare spazio nel file system
  b) Creare un nuovo elemento nella directory (registrandovi nome, posizione e altre info)
- **Scrittura**
  a) Chiamata di sistema con il nome del file e le informazioni che si vogliono scrivere
  b) Il sistema cerca il file nella directory
  c) Il file system ha un puntatore di scrittura alla locazione nel file in cui deve avvenire la prossima scrittura. Si deve aggiornare ad ogni scrittura.
- **Lettura**
  a) Chiamata di sistema con il nome del file e e la posizione dove mettere il prossimo blocco del file
  b) Il sistema cerca il file nella directory
  c) Analogamente alla scrittura, il file system deve avere un puntatore di lettura. [Poiché un processo o legge, o scrive, si usa un unico puntatore→ *puntatore alla posizione corrente del file*]
- **Riposizionamento (seek)**→ si ricerca l'elemento appropriato nella directory e si assegna un nuovo valore al puntatore della posizione corrente del file.
- **Cancellazione**
  a) Si ricerca l'elemento nella directory
  b) Si rilascia lo spazio associato al file
  c) Si elimina l'elemento della directory
- **Troncamento**→ si azzera la lunghezza del file rilasciando lo spazio occupato, ma si mantengono immutati gli attributi.

Esistono altre operazioni, come l'aggiunta di informazioni, la ridenominazione e la copiatura, che non sono altro che combinazioni delle primitive.

Per evitare di ricercare ogni volta il file, si usa la chiamata di sistema `open` (che può contenere informazioni sui modi d'accesso) la prima volta che si adopera un file in maniera attiva → il sistema operativo tiene una tabella dei file aperti (un file viene inserito quando si invoca una `open` e rimosso dopo una `close`).

In un ambiente multiutente il sistema operativo introduce due livelli di tabelle interne: una per ciascun processo (contenente i riferimenti di tutti i file aperti da quel processo) e una di sistema (contenente informazioni indipendenti dai processi). Ogni elemento della tabella dei processi punta a quella di sistema.

Informazioni associate ad un file aperto:

- Puntatore alla posizione corrente del file (nei sistemi che non prevedono lo scostamento nelle chiamate di sistema read e write) → è unico per ogni processo, quindi è separato dagli attributi del file
- Contatore dei file aperti → poiché più processi possono aprire lo stesso file, si tiene un contatore per poter rimuovere un elemento dalla tabella dei file aperti quando tutti i processi l'han chiuso (contatore = 0)
- Posizione nel disco del file
- Diritti d'accesso

Per condividere sezioni di file tra più processi, alcuni sistemi operativi offrono funzioni per l'accesso bloccante.

TIPI

Un sistema che riconosce il tipo di un file ha la possibilità di gestirlo al meglio. Una tecnica comune per riconoscerlo è quella di suddividere il nome del file in due parti: nome + estensione. L'estensione individua il tipo di file. Le estensioni non sono gestite dal sistema operativo, quindi si possono considerare un 'suggerimento' rivolto alle applicazioni che operano su di esso.

STRUTTURA

Il tipo di file indica la struttura interna del file. Alcuni sistemi operativi prevedono un insieme di strutture di file gestite gestite dal sistema con un insieme di operazioni specifiche per la manipolazione dei file. Per poter caricare ed eseguire i programmi, il sistema operativo deve prevedere almeno un tipo di struttura, quella dei file eseguibili.
Un numero eccessivamente limitato di strutture rende scomoda la programmazione (poiché qualsiasi programma d'applicazione deve contenere il proprio codice per interpretare in modo appropriato la struttura di un file); mentre troppe strutture appesantiscono il sistema operativo.

STRUTTURA INTERNA

E' improbabile che la dimensione di un record fisico corrisponda esattamente alla lunghezza del record logico desiderato, che può anche essere variabile→ per questo si 'impaccano' un certo numero di record logici in blocchi fisici. L'impaccamento può essere fatto dal programma d'applicazione oppure dal sistema operativo, e insieme alla dimensione dei blocchi fisici e dei record logici, concorre a determinare il numero di record logici all'interno di un blocco fisico. Per questo tutti i file system soffrono di frammentazione interna, che aumenta all'aumentare della dimensione dei blocchi.

METODI D'ACCESSO

SEQUENZIALE

Le informazioni del file si elaborano ordinatamente, un record dopo l'altro. La lettura fa scorrere il puntatore alla posizione corrente, avanzando dalla prima posizione; mentre la scrittura aggiunge in coda al file e sposta il puntatore nel nuovo punto di fine.

DIRETTO (relativo)

Il file si considera come una sequenza numerata di blocchi o record, che si possono leggere o scrivere in modo arbitrario

→ non esistono limiti nell'ordine.

Si devono modificare le chiamate di sistema, aggiungendovi il numero di blocco (`read` n, `write` n), oppure utilizzare le normali `read next` e `write next,` ma avendo anche una chiamata del tipo `position` n.
Il numero di blocco è relativo all'inizio del file:

- Il sistema operativo decide dove posizionare il file
- L'utente non può accedere a porzioni che non fan parte del suo file

Lettura, scrittura e cancellazione sono più facili.

ALTRI METODI

Un indice contiene puntatori ai vari blocchi; per trovare un elemento del file occorre prim cercare nell'indice, e quindi usare il puntatore per accedere direttamente al file e trovare l'elemento desiderato. Nel caso di file lunghi, può risultare lungo anche il file indice→ si può utilizzare un indice per il file indice.

STRUTTURA DI DIRECTORY

Poiché un file system può essere molto ampio, di solito lo si suddivide in più partizioni.

Directory del dispositivo: registra le informazioni di tutti i file della partizione.

Operazioni che si possono eseguire su una directory

- **Ricerca di un file** → si deve poter scorrere una directory per trovare l'elemento associato a un particolare file
- **Creazione e cancellazione di un file**
- **Elencazione di una directory**→ si deve poter elencare tutti i file di una directory e il contenuto degli elementi ad essi associati
- **Ridenominazione di un file**
- **Attraversamento del file system** → si potrebbe voler accedere a ogni directory e a ciascun file contenuto nella directory. Per motivi di affidabilità è opportuno salvare il contenuto e la struttura dell'intero file system a intervalli regolari (copie di backup).

STRUTTURE LOGICHE

A SINGOLO LIVELLO

Tutti i file sono contenuti nella stessa directory, perciò devono avere nomi unici → all'aumentare del numero dei file risulta difficile ricordarsene tutti i nomi.

A DUE LIVELLI

Ogni utente dispone della propria directory d'utente (UFD- user file directory). Quando comincia l'elaborazione di un lavoro d'utente si fa una ricerca nella directory principale (MFD- master file directory) del sistema. La directory principale viene indirizzata con l'identificativo dell'utente e ogni duo elemento punta alla relativa directory d'utente. Si possono avere file con lo stesso nome, purchè non ci siano file con lo stesso nome all'interno della stessa directory [nome d'utente + nome del file definiscono il nome di percorso (path name)]. Questa struttura isola un utente dagli altri. Alcuni sistemi non permettono l'accesso a file di utenti locali da parte di altri utenti, mentre altri li autorizzano.
I file di sistema sono contenuti in una speciale directory → ogni volta che si fa riferimento a un file il sistema operativo lo cerca nella directory d'utente locale, se non lo trova, lo cerca automaticamente nella directory d'utente che contiene i file di sistema. La sequenza di directory in cui è stata fatta la ricerca si chiama percorso di ricerca (search path).

CON STRUTTURA AD ALBERO

L'albero ha una directory radice (root directory) e ogni file del sistema ha un unico nome di percorso (descritto a partire dalla radice. Una directory contiene un insieme di file o sottodirectory.
Le directory sono semplicemente file e la distinzione tra i due è data da un bit (0 file, 1 directory).
Con una struttura di questo tipo l'accesso ai file di altri utenti è di facile realizzazione, utilizzando nome di percorso assoluti o relativi.
Nome di percorso assoluto → comincia dalla radice dell'albero di directory e segue un percorso che lo porta al file specificato, indicando i nomi delle directory intermedie.
Nome di percorso relativo → definisce un percorso che parte dalla directory corrente.
Alcuni sistemi consentono di cancellare una directory solo se è vuota, altri permettono di cancellarla eliminandone anche l'intero contenuto.

CON STRUTTURA A GRAFO ACICLICO

La struttura ad albero non consente la condivisione di file o directory, mentre una struttura a grafo aciclico lo permette. Il fatto che un file o una directory siano condivisi non significa che ci siano più copie. Si può realizzare la condivisione tramite:

1. Un collegamento (link) → è un puntatore
2. Duplicando tutte le informazioni relative ai file in entrambe le directory di condivisione → le copie e l'originale sono indistinguibili; sorge il problema di mantenere coerenza se avvengono delle modifiche.

Con questa struttura sorgono alcuni problemi

- Un file può avere più nomi di percorso assoluti → nomi diversi possono riferirsi allo stesso file. Diventa particolarmente problematico quando si vuole percorrere il file system e si deve attraversare solo una volta le strutture condivise
- Quando si cancella un file bisogna stabilire come rassegnare lo spazio associato al file condiviso e cosa fare dei collegamenti rimasti 'pendenti'
  a) È possibile cercare tutti i collegamenti al file appena cancellato e rimuoverli
  b) Si possono lasciare i collegamenti finchè non si prova ad usarli → si 'scopre' che il file con il nome dato al collegamento non esiste e non riesce a determinare il collegamento rispetto al nome
  c) Si può impedire la cancellazione del file finchè non sono stato cancellati tutti i riferimenti ad esso (per far questo, si può utilizzare un contatore)

CON STRUTTURA A GRAFO GENERICO

Poiché è difficile assicurare che un grafo sia aciclico, si possono strutturare le directory secondo un grafo generale. La

presenza di cicli tuttavia introduce altri problemi.

- Per motivi di correttezza e prestazioni è preferibile evitare una duplice ricerca di un elemento (potrebbe causare un ciclo infinito di ricerca) → si può limitare arbitrariamente il numero di directory cui accedere durante una ricerca
- Stabilire quando è possibile cancellare un file → un contatore a 0 indica la mancanza di riferimenti, come nel grafo aciclico, ma anche un valore non nullo potrebbe corrispondere a un file senza più riferimenti. Per questo si ricorre alla garbage collection (si 'segna' tutto ciò che è accessibile e si cancella ciò che non lo è in un secondo passaggio del file system)

MONTAGGIO DI UN FILE SYSTEM

Un file system per esser reso accessibile ai processi di un sistema deve esser montato.

Procedura di montaggio:

1. si fornisce al sistema operativo il nome del dispositivo e la sua locazione (punto di montaggio)[solitamente una directory vuota] nella struttura di file e directory alla quale agganciare il file system
2. Il sistema operativo verifica la validità del file system contenuto nel dispositivo → controlla che il formato della directory sia quello richiesto
3. Il sistema operativo annota nella sua struttura di directory che un certo file system è montato al punto di montaggio specificato.

CONDIVISIONE DI FILE

UTENTI MULTIPLI

Il sistema può permettere a ogni utente, in modo predefinito, di accedere ai file degli altri utenti, oppure può richiedere che un utente debba esplicitamente concedere i permessi di accesso ai file.

Il proprietario di un file o directory è l'utente che può cambiare gli attributi di un file o directory e concederne l'accesso. L'attributo di gruppo di un file si usa per definire il sottoinsieme di utenti autorizzati a condividere l'accesso al file. La maggior parte dei sistemi realizza gli attributi di proprietà gestendo un elenco di nomi di utenti e di identificatori di utenti (user ID) → numerici, unici per ogni utente. Allo stesso modo il gruppo si può realizzare come un elenco esteso di nomi di gruppi e relativi identificatori. Gli identificatori del gruppo e del proprietario sono memorizzati insieme con gli altri attributi del file. Quando un utente richiede di compiere un'operazione su un file, si può confrontare l'ID dell'utente con l'attributo che identifica il proprietario, lo stesso vale per i gruppi → il sistema considera se consentire o impedire l'operazione richiesta.

FILE SYSTEM REMOTI

Modello client-server

Il server dichiara che determinate risorse (file) sono disponibili ai client, specificando esattamente quali risorse ed esattamente a quali client. I file si specificano solitamente rispetto a una partizione o a un livello di directory.

L'identificazione certa dei client è difficile, in quanto è possibile 'imitare' l'identificativo di un client (spoofing) → una soluzione sicura è data dall'autenticazione reciproca tra client e server, per esempio, tramite chiavi di cifratura.
Una volta che il file system remoto è stato montato, le richieste delle operazioni su un file sono inviate al server.

1. Una richiesta di apertura di file si invia insieme all'ID dell'utente richiedente
2. Il server controlla se all'utente è consentito l'accesso
3. Se l'esito è positivo, riporta una 'maniglia' d'accesso (file handle) all'applicazione client
4. L'applicazione client usa la file handle per eseguire sul file le operazioni di lettura, scrittura e altro. Terminato, chiude il file.

Sistemi informativi distribuiti (servizi di dominazione distribuiti).
Forniscono un accesso unificato alle informazioni necessarie per il calcolo remoto.
Esempio → sistema di dominazione di dominio (DNS- domain name system): fornisce le traduzioni dai nomi dei calcolatori agli indirizzi di rete per l'intera Internet

Malfunzionamenti
Si può ovviare alla perdita di dati in caso di malfunzionamenti adottando strategie come le batterie RAID, ma in ogni caso un qualsiasi malfunzionamento può interrompere il flusso dei comandi. Per recuperare dai malfunzionamenti occorre mantenere informazioni di stato sia sui client che sui server. Nel caso di un server malfunzionante che ha esportato il fyle system in un sistema remoto, si realizza un DFS senza stato, ossia non tiene traccia dei client che hanno montato le sue partizioni esportate.

Semantica della coerenza
Semantica delle operazioni in cui più utenti accedono contemporaneamente a un file condiviso; specifica quando le modifiche apportate ai dati da un utente possono essere viste da altri utenti. Una serie di accessi (lettura/scrittura) sta tra una `open` e una `close` e si chiama sessione del file.

PROTEZIONE
Affidabilità: salvaguardia delle informazioni contenute in un sistema di calcolo dai danni fisici (garantita generalmente da copie di backup dei dati).
Protezione: salvaguardia delle informazioni da accessi impropri.

Accesso controllato → si limitano gli accessi a seconda di vari fattori, per esempio secondo i vari tipi di operazioni (tipi di accessi)

- Lettura
- Scrittura
- Esecuzione
- Aggiunta
- Cancellazione
- Elencazione

Più utenti possono richiedere diversi tipi di accesso a un file o una directory. Per realizzare gli accessi dipendenti dall'identità si associa un elenco di controllo degli accessi (ACL- access control list) a ogni file e directory. Quando un utente richiede un accesso a un file specifico il sistema operativo esamina questo elenco, decidendo se autorizzare o no l'accesso.
Vantaggio: son permessi complessi metodi d'accesso.
Svantaggio: l'elenco di controllo risulta lungo e l'elemento della directory passa da una dimensione fissa a una variabile, rendendo la gestione dello spazio più complicata.
Per condensare la lunghezza dell'elenco si possono raggruppare gli utenti in tre classi:

- Proprietario
- Gruppo → insieme di utenti che condividono il file e richiedono tipi di accesso simili;
- Universo → tutti gli altri utenti del sistema

Un altro metodo di protezione consiste nell'associare una parola d'ordine a ciascun file o directory.

Capitolo 12

STRUTTURA

La maggior parte della memoria secondaria è costituita da dischi, che

1. si possono riscrivere localmente (leggo un blocco, lo modifico e lo riscrivo nella stessa posizione);
2. è possibile accedere direttamente a qualsiasi blocco sia in modo sequenziale che diretto.

Anziché trasferire 1 byte alla volta, per migliorare l'efficienza dell'I/O, i trasferimenti tra memoria centrale e i dischi si eseguono per blocchi. Ciascun blocco è composto da uno o più settori.

Problemi di progettazione di un file system:

1. **Definizione dell'aspetto del file system agli occhi dell'utente →definizione di file e dei suoi attributi, operazioni permesse, struttura delle directory.**
2. **Creazione di algoritmi e strutture dati per far corrispondere il file system logico ai dispositivi fisici di memoria secondaria.**

Un file system è composto da livelli, ciascuno dei quali si serve dei livelli inferiori per creare nuove funzioni che sono impiegate dai livelli superiori.

1. Dispositivi
2. Controllo dell'I/O → **sono i driver dei dispositivi e i gestori dei segnali di interruzione che si occupano del trasferimento delle informazioni tra la memoria centrale e la memoria secondaria.**
3. File system di base → **invia generici comandi all'appropriato driver di dispositivo per leggere e scrivere blocchi fisici nel disco.**
4. Modulo di organizzazione dei file → **è a conoscenza dei file e dei loro blocchi logici, così come dei blocchi fisici dei dischi. Si occupa della traduzione degli indirizzi dei blocchi logici negli indirizzi dei blocchi fisici. Comprende anche il gestore dello spazio libero.**
5. File system logico → **gestisce i metadati (ossia tutte le strutture del file system eccetto il contenuto dei file). E' responsabile anche della protezione e della sicurezza.**
6. Programmi d'applicazione

Esistono molti tipi di file system e la maggior parte dei sistemi operativi ne impiega più di uno. L'uso della struttura

stratificata consente di ridurre al minimo la duplicazione del codice.

REALIZZAZIONE

Strutture presenti nei dischi

- Blocco di controllo d'avviamento→ contiene le informazioni necessarie al sistema per l'avviamento di un sistema operativo da quella partizione; solitamente è il primo blocco di una partizione;
- Blocchi di controllo delle partizioni → contiene dettagli relativi alla partizione (numero e dimensione dei blocchi, contatore dei blocchi e dei file liberi coi relativi puntatori);
- Strutture di directory
- Descrittori di file→ contiene dettagli dei file (permessi d'accesso, proprietari, dimensioni, locazione dei blocchi di dati).

Ci sono strutture che contengono informazioni utili per gestire il file system e migliorare le prestazioni della cache:

- Tabella delle partizioni → contiene informazioni su ciascuna delle partizioni montate.
- Struttura di directory, tenuta in memoria → contiene informazioni sulle directory a cui i processi hanno avuto accesso di recente.
- Tabella generale dei file aperti
- Tabella dei file aperti per ciascun processo → contiene un puntatore all'appropriato elemento nella tabella generale dei file aperti.

Per creare un nuovo file, un'applicazione effettua una chiamata di sistema al file system logiche che:

- Assegna un nuovo descrittore al file
- Carica nella memoria la directory appropriata e l'aggiorna con il nuovo nome e descrittore del file
- Riscrive la directory in memoria secondaria.

Per aprire e chiudere un file:
- Chiamata di sistema open
- Verifica nella tabella generale dei file aperti
    a) Se c'è, si crea un puntatore nella tabella del processo a quella generale
    b) Se non c'è, il file system cerca il file e ne copia il descrittore nella tabella generale dei file aperti e poi crea un puntatore nella tabella del processo
- Quando si chiude un file, si cancella l'elemento nella tabella del processo e si decrementa il contatore associato al file nella tabella generale dei file aperti
    a) Se il contatore raggiunge lo 0, tutti i processi hanno chiuso il file→ lo si aggiorna in memoria secondaria e si rimuove l'elemento nella tabella.

Un disco si può configurare in vari modi, a seconda del sistema operativo che lo gestisce. Si può suddividere in più partizioni o una partizione può stare su più dischi.
Raw partition→ partizione priva di struttura logica, senza alcun file system.

Nella fase di caricamento del sistema operativo si esegue il montaggio della partizione radice (root partition) che contiene il nucleo del sistema operativo e anche altri file si sistema. Il montaggio delle altre partizioni può avvenire automaticamente in questa fase o successivamente in modo esplicito. Infine il sistema annota nella struttura della tabella di montaggio nella memoria che un file system, e relativo tipo, è stato montato.

FILE SYSTEM VIRTUALE
Un metodo per realizzare più tipi di file system sarebbe di creare procedure di gestione separate, ma non è ottimale.
La maggior parte dei sistemi operativi adotta un file system composto da 3 strati:
1. Interfaccia del file system (chiamate di sistema open, read, write, close e descrittori di file)
2. File system virtuale
    - Separa le operazioni generiche del file system dalla loro realizzazione. Possono coesistere più interfacce.
    - Vnode: indicatore numerico unico per tutta la rete che rappresenta ciascun file e directory →permette di gestire le richieste remote
3. File system locali

REALIZZAZIONE DELLE DIRECTORY
Lista lineare
Si può realizzare una directory come una lista lineare contenente i nomi dei file coi puntatori ai blocchi di dati.
Vantaggio: di facile programmazione
Svantaggio: per trovare un file bisogna ricorrere a una onerosa ricerca lineare. Per ridurre i tempi di ricerca si può utilizzare una cache con le directory utilizzate più recentemente, o ricorrere ad una lista ordinata.

Tabella hash
Un lista lineare contiene gli elementi di directory, e si utilizza anche una struttura dati hash che riceve come operando il nome del file e riporta un puntatore all'elemento corrispondente nella lista. Per evitare collisioni (ossia il caso in cui due nomi di file facciano riferimento alla stessa locazione) si può realizzare ciascun elemento della tabella di hash come una lista concatenata

METODI DI ASSEGNAZIONE
Contigua
Ogni file deve occupare un insieme contiguo di blocchi del disco ed è definito dall'indirizzo del primo blocco e dalla

sua lunghezza. Accedere a un file il cui spazio è assegnato in modo contiguo è facile.

Accesso sequenziale → il file system memorizza l'indirizzo dell'ultimo blocco cui è stato fatto riferimento e , se necessario, legge il blocco successivo.
Accesso diretto → accedere al blocco i di un file che inizia nel blocco b corrisponde ad accedere direttamente al blocco b+i

Problemi
- Individuare lo spazio per un nuovo file → problema generale dell'assegnazione dinamica della memoria. Si sceglie generalmente un buco secondo il criterio first-fit o best-fit, ma questo genera frammentazione esterna. Una soluzione potrebbe essere la compattazione, ma richiede tempo in cui il sistema non può funzionare, quindi risulta troppo onerosa per i calcolatori in continua attività.

- Determinare la quantità di spazio necessaria al file → infatti se riceve troppo poco spazio si rischia di non poterlo estendere, invece troppo spazio potrebbe risultare uno spreco di memoria
    - Il programma può terminare con un messaggio d'errore e 'costringere' l'utente ad assegnare più spazio
    - Si può trovare un buco più grande, copiarvi il file e liberare lo spazio precedentemente occupato.
    - Schema di assegnazione contigua modificato → si assegna una porzione di spazio contiguo al file e, se questa non risultasse sufficiente, si aggiunge un'estensione (la locazione dei blocchi dei file sarà quindi <blocco iniziale, lunghezza, 1° blocco estensione>

Concatenata

Ogni file è composto da una lista concatenata di blocchi del disco i quali possono essere sparsi in qualsiasi punto del

disco (ogni blocco contiene un puntatore al successivo) → ciò risolve i problemi dell'assegnazione contigua.

Problemi

- Può essere usata efficacemente solo per i file ad accesso sequenziale
- I puntatori richiedono spazio → una soluzione può essere riunire un certo numero di blocchi contigui in gruppi (cluster) e assegnare gruppi di blocchi anziché singoli blocchi.
- Poiché i file sono tenuti insieme da puntatori sparsi per tutto il disco, sarebbe particolarmente grave se uno di questi venisse perso o danneggiato
    - Si potrebbero usare liste doppiamente concatenate oppure
    - Memorizzare nome del file e numero di blocco in ogni blocco.

Una variante dell'assegnazione concatenata è l'uso della tabella di assegnazione dei file (FAT-file allocation table). Questa tabella è tenuta all'inizio di ciascuna partizione, contiene un elemento per ogni blocco del disco ed è indicizzata dal numero del blocco. L'elemento di directory contiene il numero del primo blocco del file; la tabella contiene come elemento corrispondente ad un blocco il numero di blocco successivo. La fine del file ha un valore speciale; a un blocco libero corrisponde il valore 0. Questo metodo risulta particolarmente efficiente se si utilizza una cache.

Indicizzata

Si raggruppano i puntatori in un'unica locazione → blocco indice: è un vettore il cui i-esimo elemento punta all'i-esimo blocco del file. La directory contiene l'indirizzo del blocco indice. In questo modo è possibile l'accesso diretto senza soffrire della frammentazione esterna; d'altra parte lo spazio richiesto dai puntatori del blocco indice è superiore a quello richiesto dai puntatori nell'assegnazione concatenata.
Sorge la questione di trovare una giusta dimensione del blocco indice.

- Schema concatenato: il blocco d'indice è formato da un solo blocco → in caso di file lunghi è possibile collegare tra loro più blocchi d'indice, usando l'ultimo elemento del primo come puntatore al secondo e così via.
- Indice a più livelli: un blocco di primo livello punta a un insieme di blocchi indice di secondo livello, che puntano ai blocchi dei file.
- Schema combinato: in un blocco d'indice da 15 elementi, se ne possono usare 12 che puntano direttamente ai blocchi del file, uno che punta indirettamente a un singolo blocco indice e i restanti 2 di secondo e terzo livello.

GESTIONE DELLO SPAZIO LIBERO

Per creare un file occorre cercare nell'elenco dei blocchi liberi la quantità di spazio necessaria e assegnarla al nuovo file, quindi rimuovere questo spazio dall'elenco. Quando si cancella un file, si aggiungono i blocchi liberati all'elenco. L'elenco dei blocchi liberi può essere realizzato come:

- Vettore di bit → ogni blocco è identificato da un bit (1: blocco libero, 0: blocco assegnato). E' facile trovare il primo blocco libero o n blocchi liberi consecutivi nel disco, tuttavia è efficiente se tenuto in memoria centrale e questo non è possibile per dischi di grandi dimensioni.
- Lista concatenata → si tiene un puntatore al primo blocco libero in una speciale locazione del disco; il primo avrà un puntatore al secondo e così via. In questo modo attraversare tutta la lista richiede un notevole tempo di I/O; d'altra parte questa non è un'operazione frequente.

- Raggruppamento→ si memorizzano gli indirizzi dei primi n-1 blocchi liberi nel primo di questi. L'ultimo elemento può puntare a un altro blocco. Questo metodo consente di trovare rapidamente gli indirizzi di un gran numero di blocchi liberi.

EFFICIENZA E PRESTAZIONI

L'efficienza dipende dagli algoritmi usati per l'assegnazione del disco, dalla informazioni che si vuole associare a un file, dalla dimensione dei puntatori, etc..

Le prestazioni di possono migliorare:

- Dotando i controllori dei dischi di una cache interna.
- Quando i blocchi vengono trasferiti dal controllore alla memoria centrale, il sistema operativo può inserirli in una propria cache nella memoria centrale (in una zona separata → cache del disco; oppure impiegando techiche di memoria virtuale→ cache delle pagine [memoria virtuale unificata]).
  Buffer cache unificata: sia l'associazione alla memoria sia le chiamate di sistema read e write usano la stessa cache delle pagine, col vantaggio di evitare il double caching e di permettere al sistema di memoria virtuale di gestire i dati del file system.

Indipendentemente dalla gestione della cache per blocchi di disco oppure per pagine, l'algoritmo LRU è in generale ragionevole per la sostituzione dei blocchi o delle pagine.

- Scritture sincrone: avvengono nell'ordine in cui le riceve il sottosistema per la gestione del disco e non subiscono la memorizzazione transitoria → la procedura chiamante per proseguire deve attendere che i dati raggiungano il disco

Scritture asincrone: si memorizzano i dati nella cache e si restituisce immediatamente il controllo alla procedura chiamante

- Gli accessi sequenziali possono essere migliorati con
  - Rilascio indietro (free-behind) → rimuove una pagina dalla memoria di transito non appena si verifica una richiesta della pagina successiva
  - Lettura anticipata (read-ahead) → si leggono e si mettono nella cache la pagina richiesta a parecchie pagine successive.
- Disco virtuale (disco RAM): è una porzione della memoria centrale i cui driver accettano tutte le operazioni ordinarie dei dischi, eseguendole per l'appunto in memoria al posto che su un disco. La differenza con una cache è che quest'ultima è sotto il controllo del sistema operativo, mentre il disco RAM è controllato dall'utente.

RIPRISTINO

Durante il riavvio del sistema si esegue uno speciale programma, il verificatore della coerenza, che confronta i dati nelle directory con quelli contenuti nei blocchi dei dischi, tentando di correggere ogni incoerenza. Gli algoritmi di assegnazione e di gestione dello spazio libero determinano il genere di problemi che questo programma può riconoscere e con quanto successo riuscirà a risolverli

Si possono usare programmi di sistema che consentano di fare copie di riserva dei dati residenti nei dischi di altri dispositivi di memorizzazione dei dati. Il ripristino della situazione antecedente il guasto richiederà il recupero dei dati persi dalle copie di backup. Al fine di ridurre al minimo la quantità dei dati da copiare, è possibile sfruttare informazioni contenute nell'elemento della directory associato a ogni file (es data dell'ultima scrittura, copiatura…)

FILE SYSTEM CON ANNOTAZIONE DELLE MODIFICHE

Le strutture dati del file system risiedono nei dischi possono diventare incoerenti nel caso di crollo del sistema. In un file system orientato alle transazioni e basato sull'annotazione delle modifiche, si registrano tutte le modifiche dei metadati in maniera sequenziale nel *giornale delle modifiche.*

L'insieme di operazioni che esegue un determinato compito si chiama transazione; quanto una transazione va a buon fine, si cancellano le relative registrazioni nel giornale (che è realizzato come una struttura dati circolare). In questo modo, se si verifica un crollo del sistema, le transazioni nel giornale sono quelle non terminate e che devono essere completate. (*Nota della riassuntrice☺funziona come la ripresa a caldo*)

NFS (net file system)

E' un esempio di file system di rete client-server, dove ogni stazione di lavoro ha un file system indipendente e si vuole permettere un certo grado di condivisione dei dati, su richiesta esplicita.

- Un calcolatore può essere sia client sia server → la condivisione è ammessa per ogni coppia di calcolatori
- La condivisione ha effetto solamente nei calcolatori client
- Affinché una directory remota sia accessibile in modo trasparente a un calcolatore, il suo client deve eseguire un'operazione di montaggio in una directory locale. La directory così montata sostituisce il sottoalbero preesistente e diventa parte integrante del file system
- In alcune versioni dell'NFS sono permessi anche montaggi a cascata: un file system si può montare in corrispondenza di altri file system montati in modo remoto.
- Un calcolatore è sottoposto solo ai montaggi da lui richiesti
- Il meccanismo di montaggio non ha la proprietà transitiva → quando monta un file system remoto, il client non ha l'accesso ai file system montati sopra il primo
- Mobilità dell'utente→ se si monta un file system condiviso in corrispondenza delle directory iniziali degli utenti di tutti i calcolatori di una rete, un utente può aprire una sessione in qualsiasi stazione di lavoro e prelevare il proprio ambinte di lavoro iniziale.
- L'NFS usa RPC costruite sul protocollo XDR, usato tra due interfacce indipendenti dalla realizzazione → l'NFS può operare in un ambiente eterogeneo.

La definizione dell'NFS distingue tra

o Protocollo di montaggio

Stabilisce la connessione logica iniziale tra un server e un client

1. la richiesta di montaggio si associa alla RPC corrispondente e s'invia al server di montaggio in esecuzione nello specifico calcolatore server
2. il server conserva una lista di esportazione che specifica i file system locali esportati per il montaggio e i nomi dei calcolatori ai quali è permessa tale operazione
3. Se la richiesta è conforme alla propria lista di esportazione, il server riporta al client una maniglia di file (file handle) da usare come chiave per ulteriori accessi ai file che si trovano all'interno del file system montato

o Protocollo NFS

Si occupa degli accessi ai file remoti. Offre un insieme di RPC per svolgere le seguenti operazioni:

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

Non ci sono open e close perché l'NFS è senza informazione di stato (vedi capitolo 16).

Di una singola chiamata di procedura di scrittura dell'NFS sono garantite l'atomicità e la non interferenza con altre chiamate di scrittura sullos tesso file; tuttavia il protocollo NFS non offre meccanismi per il controllo della concorrenza → si deve ricorrere a un servizio esterno.

L'NFS è integrato nel sistema operativo tramite un VFS.

Traduzione dei nomi di percorso

Si compie suddividendo il percorso stesso in nomi componenti ed eseguendo una chiamata di lookup dell'NFS separata per ogni coppia formata da un nome componente e un nome di directory. Quando s'incontra un punto di montaggio, la ricerca di un componente causa una RPC separata al server.

Una cache per la ricerca dei nomi delle directory, nel sito client, conserva i vnode per i nomi delle directory remote.

Nel caso di montaggio a cascata, ogni ricerca di componente si compie tra il client originale e alcuni server, perciò quando un client fa una ricerca in una directory sulla quale il server ha montato un file system, il client vede la directory sottostante e non la directory montata.

Funzionamento remoto

Dal punto di vista concettuale l'NFS aderisce al paradigma del servizio remoto, ma in pratica usa tecniche di memorizzazione transitoria e cache per migliorare le prestazioni. Non c'è una corrispondenza diretta tra un'operazione remota e una RPC, le RPC prelevano blocchi e attributi di file che memorizzano localmente nelle cache. Le successive operazioni remote usano i dati nella cache, soggetti a vincoli di coerenza. Esistono due cache: una degli attributi del file e una dei blocchi del file. La cache degli attributi viene aggiornata ogni volta che arrivano nuovi attributi dal server. Tra server e client si usano tecniche di lettura anticipata e scrittura differita→ i client non liberano i blocchi di scrittura differita finchè il server non ha confermato che i dati sono stati scritti nei dischi.

Esistono vari problemi legati alla coerenza della cache, ma nonostante ciò l'NFS è il sistema distribuito più usato grazie alla sua utilità e alte prestazioni.

Capitolo 13

I due compiti principali del calcolatore sono l'I/O e l'elaborazione.
I driver dei dispositivi offrono al sottosistema di I/O un'interfaccia uniforme per l'accesso ai dispositivi, così come le chiamate di sistema forniscono un'interfaccia uniforme tra le applicazioni e il sistema operativo.

ARCHITETTURE E DISPOSITIVI DI I/O

Porta: punto di connessione tramite il quale un dispositivo comunica con il calcolatore, inviando segnali attraverso un cavo o l'etere.
Una porta di I/O consiste in 4 registri

- Status: indica lo stato della porta
- Control: si usa per attivare un comando o cambiare modo di funzionamento del dispositivo
- Data-in: la CPU lo legge per ricevere dati
- Data-out: la CPU vi scrive per emettere dati

Bus: insieme di fili e protocollo che specifica l'insieme di messaggi che si possono inviare attraverso i fili.

- Bus PCI → comune bus di sistema del pc
- Bus di espansione → vi si connettono i dispositivi relativamente lenti.

Daisy chain→ un dispositivo è collegato ad un altro dispositivo a sua volta collegato al calcolatore.
Controllore: insieme di componenti elettronici che può far funzionare una porta, un bus o un dispositivo. Per dispositivi particolarmente complessi, si tratta di un adattatore, ossia una scheda circuitale contenente un'unità di elaborazione, microcodice e memoria, che gli permettono di elaborare i messaggi di un particolare protocollo (ad esempio, SCSI).

La CPU dà comandi e fornisce dati al controllore tramite operazioni di lettura e scrittura in appositi registri per dati e segnali di controllo.

a) Si utilizzano speciali istruzioni di I/O per il trasferimento di un byte a un indirizzo di porta di I/O
b) I registri di controllo del dispositivo si fanno corrispondere a un sottoinsieme dello spazio di indirizzi della CPU, che esegue le richieste di I/O usando le ordinarie istruzioni di trasferimento dei dati (I/O associato alla memoria). Questo tipo di registri son più vulnerabili alle modifiche accidentali, anche se le tecniche di protezione della memoria riducono questo rischio.

Interrogazione ciclica (polling, attesa attiva)

1. La CPU legge ripetutamente il registro di stato finché non risulta che il dispositivo è disponibile
2. La CPU segnala nel registro command il tipo di istruzione da eseguire e inserisce dati nel registro data-out
3. La CPU segnala nel registro command che l'istruzione è pronta
4. quando il controllore si accorge di avere un'istruzione da eseguire, segnala nel registro di stato che la periferica è occupata
5. il controllore legge nel registro command l'istruzione e nel registro data-out i dati da utilizzare. Svolge l'operazione richiesta
6. il controllore azzera il segnale di istruzione pronta nel registro command e nel registro di stato indica che l'operazione è stata eseguita senza errori e ora la periferica è disponibile.

Interrupt

I controllori segnalano tramite un contatto con la CPU (linea di controllo dell'interruzione) il fatto di avere delle operazioni da eseguire. La CPU controlla questo contatto dopo l'esecuzione di ogni istruzione e quando rileva una richiesta, salva lo stato dell'elaborazione corrente e salta alla procedura di gestione delle interruzioni, che porta a termine l'elaborazione necessaria ed invia un'istruzione per riportare la CPU nello stato precedente all'interruzione.
Per gestire al meglio le interruzioni, CPU e controllore delle interruzioni devono:

1. poter differire la gestione delle interruzioni durante le elaborazioni critiche → a questo scopo, le CPU hanno 2 linee di richiesta delle interruzioni
    a. non mascherabili → errori irrecuperabili
    b. mascherabili → la linea può essere disattivata dalla CPU prima dell'esecuzione di una sequenza critica delle operazioni.
2. avere un modo efficiente per recapitare un segnale d'interruzione al corretto gestore → il vettore delle interruzioni contiene gli indirizzi di memoria degli specifici gestori delle interruzioni. Il meccanismo delle interruzioni accetta un indirizzo, che è uno scostamento relativo a questo vettore. Poiché i calcolatori hanno più dispositivi che elementi nel vettore, questi non sono altro che puntatori agli effettivi gestori delle interruzioni
3. poter distinguere tra interruzioni di alta e bassa priorità → si utilizza un sistema di livelli di priorità.

I segnali di interruzione sono usati per gestire eventi asincroni (operazioni di I/O) e per effettuare nel modo supervisore le procedure del nucleo (eccezioni causate da chiamate di sistema e gestione del controllo del flusso all'interno del nucleo). Questi eventi hanno in comune il fatto che inducono la CPU a eseguire una procedura autonoma.

Accesso diretto alla memoria (DMA)
Quando un dispositivo compie trasferimenti di grandi quantità di dati (es. disco) risulta costoso impiegare la CPU per controllare il registro di stato e dei dati un byte alla volta → si ricorre a un'unità di elaborazione specializzata, detta controllore dell'accesso diretto alla memoria (DMA).

1. al driver del dispositivo si chiede di trasferire dati dal disco al buffer di indirizzo X
2. il driver chiede al controllore del disco di trasferire n byte dal disco alla locazione di memoria di indirizzo X
3. il controllore del disco avvia il trasferimento DMA
4. il controllore del disco invia i dati byte per byte al controllore DMA
5. il controllore DMA trasferisce i byte alla locazione di memoria di indirizzo X, incrementando l'indirizzo di memoria e decrementando n finché n=0
6. quando n=0, il controllore DMA genera un segnale di interruzione per segnalare alla CPU che il trasferimento è terminato.

Sottrazione di cicli: quando il DMA prende possesso del bus della memoria, la CPU non può accedere alla memoria centrale pur potendo continuare a elaborare i dati contenuti nella sua cache

Il DMA può utilizzare indirizzi fisici o virtuali (DVMA- direct virtual memory access)→ in quest'ultimo caso è

possibile compiere trasferimenti di dati tra due dispositivi di I/O associato alla memoria senza l'intervento della CPU

INTERFACCIA DI I/O PER LE APPLICAZIONI
Si possono identificare alcuni tipi generali di dispositivi I/O, astraendoli e incapsulando le differenze in moduli del nucleo specializzati per gli appositi dispositivi (driver di dispositivo). Si accede ad ogni 'tipo' per mezzo di un unico insieme di funzioni, l'interfaccia. I dispositivi possono differire per:

- Modo di trasferimento dei dati
  - A caratteri
  - A blocchi
- Modo d'accesso
  - Sequenziale
  - Diretto
- Prevedibilità dell'I/O
  - Sincrono
  - Asincrono
- Condivisione
  - Dedicato
  - Condiviso
- Velocità
  - Latenza
  - Tempo di ricerca
  - Velocità di trasferimento
  - Attesa tra le operazioni
- Direzione dell'I/O
  - Lettura e scrittura
  - Solo lettura
  - Solo scrittura

I 'tipi' principali sono:

- I/O a blocchi → il file system funge da interfaccia tra l'applicazione e i blocchi. I dispositivi vengono considerati come una sequenza lineare di blocchi (I/O di basso livello)
- Accesso ai file associato alla memoria→ di un gradino superiore all'I/O di basso livello, permette di usare un'unità a disco tramite un vettore di byte in memoria centrale. La chiamata di sistema che associa un file a una regione di memoria restituisce l'indirizzo di memoria virtuale di un vettore di caratteri che contiene una copia del file
- I/O a flusso di caratteri → le chiamate di sistema fondamentali per le interfacce di questo tipo permettono di acquisire o inviare un carattere alla volta

- Socket di rete → una socket è come una presa di corrente che fornisce una base per poter effettuare una connessione. Un'applicazione crea una socket e si collega ad una socket creata da un'altra applicazione nella rete. A questo punto si possono inviare e ricevere pacchetti lungo la connessione.

Back door: permette il passaggio trasparente di comandi arbitrari da un'applicazione a un driver di dispositivo.

Orologi e temporizzatori

Le loro funzioni principali sono:
- Segnare l'ora corrente
- Segnalare il tempo trascorso
- Regolare un temporizzatore per avviare un'operazione dopo un certo tempo

I/O bloccante: una chiamata di sistema bloccante sospende l'esecuzione dell'applicazione e la riprende una volta terminata la chiamata di sistema
I/O non bloccante: a volte è necessario non interrompere continuamente l'elaborazione (es.mouse)
- Si può scrivere un'applicazione a più thread, alcuni bloccanti, altri no
- Chiamate di sistema asincrone

SOTTOSISTEMA PER L'I/O DEL NUCLEO

Il sottosistema per l'I/O coordina un'ampia raccolta di servizi disponibili per le applicazioni e per altre parti del nucleo; in generale sovrintende alle seguenti funzioni:
- Gestione dello spazio dei nomi dei file e dei dispositivi
- Controllo dell'accesso ai file e ai dispositivi
- Controllo delle operazioni
- Assegnazione dello spazio per il file system
- Assegnazione dei dispositivi
- Memorizzazione transitoria, gestione della cache e delle code per la gestione asincrona dell'I/O
- Scheduling dell'I/O
- Controllo dello stato dei dispositivi, gestione degli errori e procedure di ripristino
- Configurazione e inizializzazione dei driver dei dispositivi

I livelli superiori del sottosistema per la gestione dell'I/O accedono ai dispositivi per mezzo dell'interfaccia uniforme fornita dai driver.

Quando un'applicazione richiede l'esecuzione di una chiamata del sistema di i7o bloccante, si aggiunge la richiesta alla coda relativa al dispositivo. Lo scheduler dell'I/O riorganizza l'ordine della coda per migliorare l'efficienza del sistema.

Una memoria di transito (buffer) è un'area di memoria che contiene dati mentre essi sono trasferiti fra due dispositivi o tra un'applicazione e un dispositivo. Si usa per:
- Gestire la differenza di velocità tra mittente e destinatario dei dati
- Trasferire dati in blocchi di dimensioni diverse
- Realizzare la semantica delle copie nell'ambito dell'I/O delle applicazioni. Questa garantisce che la versione dei dati scritta nel disco sia conforme a quella contenuta nell'area di memoria al momento della chiamata di sistema, indipendentemente da ogni successiva modifica.

Una cache è una regione di memoria veloce che serve per mantenere copie di certi dati → l'accesso a questa copia è più rapido degli accessi agli originali, quindi, quando riceve una richiesta di I/O relativa a un file, il nucleo controlla se è già presente nella cache.

Un buffer può contenere dati di cui non esistono altre copie, mentre una cache contiene copie di dati già memorizzati.

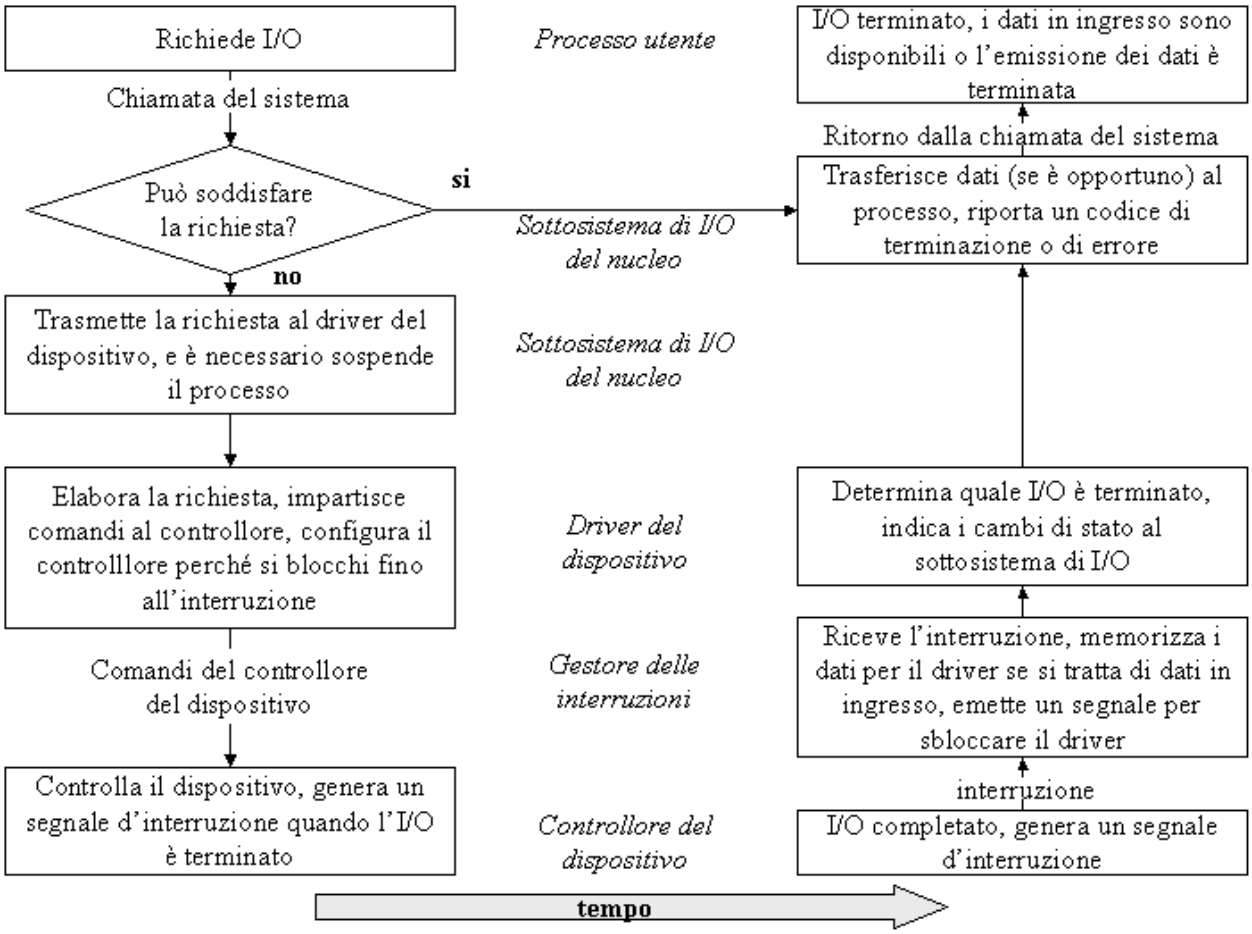
L'accodamento è uno dei modi in cui il sistema operativo può coordinare le emissioni simultanee di dati.

Alcuni sistemi operativi permettono di accedere a un dispositivo in modo esclusivo, riservandone l'uso a un processo finché esso non termina.

Di norma, una chiamata di sistema per l'I/O riporta un bit d'informazione sullo stato d'esecuzione della chiamata, segnalando se l'operazione è riuscita o no.
L'errore può avvenire per ragioni temporanee o permanenti. Generalmente i sistemi operativi riescono a gestire le prime, ma è improbabile che ci riescano con le seconde.

Il nucleo ha bisogno di mantenere informazioni sullo stato dei componenti coinvolti nelle operazioni di I/O, e usa a questo scopo diverse strutture di dati interne (es. tabella dei file aperti).

TRASFORMAZIONE DELLE RICHIESTE DI I/O IN OPERAZIONI DEI DISPOSITIVI



STREAMS

Stream → in UNIX, è una connessione full-duplex fra un driver di dispositivo e un processo utente. E' composto da
- Elemento iniziale d'interfaccia per il processo utente (stream head)
- Elemento terminale che controlla il dispositivo (driver end)
- Moduli intermedi (stream modules)

Ognuno di questi elementi contiene una coppia di code (una di lettura, una di scrittura) e il trasferimento tra le due code avviene tramite lo scambio di messaggi. Poiché i messaggi sono scambiati tra code di moduli adiacenti, si può disporre di un controllo di flusso per evitare che la coda di un modulo sconfini in quella di un'altra.
Indipendentemente dalla chiamata di sistema del processo utente, l'elemento iniziale d'interfaccia copia i dati in un messaggio e li recapita alla coda del modulo successivo. La copiatura continua finché il messaggio giunge all'elemento terminale di controllo e quindi al dispositivo. Analogamente, il processo legge i dati dall'elemento iniziale di interfaccia. L'I/O per mezzo di STREAMS è asincrono (non bloccante).

Vantaggi
- Si dispone di un ambiente che permette uno sviluppo modulare e incrementale di driver dei dispositivi e protocolli di rete

- Invece di trattare l'I/O di dispositivi a caratteri come una sequenza non strutturata di byte, permette la gestione dei limiti di messaggi e delle informazioni di controllo tra i diversi moduli.

PRESTAZIONI DELL'I/O

La gestione delle interruzioni è un processo oneroso, poiché comporta che il sistema cambi stato, gestisca l'interruzione e ripristini lo stato originario. Se il numero di cicli impiegato nell'attesa attiva non è eccessivo, l'I/O programmato può essere più efficiente di quello basato sulle interruzioni.

Canali di I/O: unità di elaborazione specializzate nella gestione dell'I/O → utilizzati nei sistemi di alto profilo (es. mainframe) riducono il carico della CPU.

Per migliorare le prestazioni dell'I/O

- Ridurre il numero di cambi di contesto
- Ridurre il numero di copiature dei dati durante i trasferimenti fra dispositivi

1. Inizialmente gli algoritmi sperimentali per l'I/O si codificano a livello delle applicazioni → maggior flessibilità a scapito dell'impossibilità di sfruttare dati del nucleo e i suoi meccanismi interni (minor efficienza)
2. Una volta messo a punto, è possibile codificare l'algoritmo all'interno del nucleo
3. Per ottenere le migliori prestazioni, si integrano gli algoritmi nei dispositivi eo nei controllori, anche se ciò comporta un elevato costo di eventuali successive migliorie e una minore flessibilità.

Capitolo 14

STRUTTURA DEI DISCHI
I moderni dischi si considerano come un grande vettore monodimensionale di blocchi logici (blocco logico → minima unità di trasferimento). Il vettore corrisponde in modo sequenziale ai settori del disco. Si potrebbe teoricamente sfruttare ciò per trasformare un numero di blocco logico in un indirizzo fisico, ma non lo si può fare in pratica perché:

- I dischi possono contenere settori difettosi, ma la corrispondenza lo nasconde sostituendoli con settori funzionanti in altre parti del disco
- Il numero dei settori per traccia in certi dischi non è costante

(CVL- costant linear velocity) velocità lineare costante → la densità di bit per traccia è uniforme. La velocità di rotazione aumenta all'avvicinarsi delle tracce più interne per mantenere costante la quantità di dati che scorrono sotto le testine

(CAV- costant angular velocity) velocità angolare costante → la densità di bit decresce man mano che ci si sposta verso tracce più esterne. La velocità di rotazione rimane costante.

SCHEDULING DEI DISCHI

Tempo di ricerca (seek time) = tempo necessario affinché il braccio dell'unità a disco sposti le testine fino al cilindro

contenente il settore desiderato.

Latenza di rotazione (rotational latency) = tempo aggiuntivo affinché il disco ruoti fino al settore desiderato.
Ampiezza di banda = numero di byte trasferiti nell'unità di tempo

Quando un processo invoca una chiamata di sistema per compiere operazioni di I/O, questa contiene informazioni del tipo

- Se l'operazione è di immissione o emissione dati
- L'indirizzo nel disco e quello di memoria rispetto al quale effettuare il trasferimento
- Il numero di byte da trasferire

Queste richieste possono essere soddisfatte subito, o in caso contrario, vengono messe nelle code relative agli specifici dispositivi.

Scheduling in ordine di arrivo (FCFS)
Si 'servono' le richieste in ordine di arrivo. Questo è un metodo equo, ma non garantisce una buona velocità di servizio.

Scheduling per brevità (SSTF-shortest seek time first)
Si sceglie la richiesta che dà il minimo tempo di ricerca rispetto alla posizione corrente della testina; poiché questo tempo cresce al crescere della distanza dei cilindri dalla testina, si scelgono le richieste relative ai cilindri più vicini. Questo metodo può presentare situazioni di attesa indefinita.

Scheduling per scansione (SCAN)
Il braccio dell'unità a disco parte da un estremo del disco e si sposta nella sola direzione possibile, servendo le richieste mentre attraversa i cilindri, fino a che non raggiunge l'altro estremo del disco: inverte la marcia continuando la procedura. E' chiamato algoritmo dell'ascensore perché serve tutte le richieste 'in salita' e poi quelle 'in discesa',

Scheduling per scansione circolare (C-SCAN)
E' una variante dell'algoritmo SCAN. Si comporta come quest'ultimo, solo che quando giunge a un estremo, la testina

riprende dall'inizio del disco, trattandolo come una lista circolare → questo garantisce un tempo di attesa meno

variabile.

Scheduling LOOK e C-LOOK

Sono delle versioni dello SCAN e C-SCAN che non giungono tutte le volte agli estremi del disco, ma seguono una direzione finché vi sono richieste da servire, dopo di che la invertono.

Le prestazioni dipendono da:

- Numero e tipo di richieste
- Metodo adottato per l'assegnazione dei file
- Posizione delle directory e dei blocchi indice (utilizzando la memoria centrale come cache per questi, si riducono i movimenti del braccio dell'unità a disco)

I controllori delle unità a disco incorporano algoritmi di scheduling → quando il sistema operativo invia un gruppo di richieste, il controllore le organizza in una coda e applica gli opportuni scheduling.

Purtroppo il sistema operativo non può appoggiarsi in pieno a questo poiché deve tener conto delle priorità e rendere il file system robusto rispetto ai crolli di sistema.

GESTIONE DELL'UNITA' A DISCO

Formattazione

Prima di poter memorizzare dati su un disco magnetico questo deve essere diviso in settori → formattazione fisica (di basso livello). Questo processo riempie ogni settore con una speciale struttura dati, consistente in un'intestazione, un'area dati e una coda. Intestazione e coda contengono informazioni usate dal controllore del disco (es. numero del settore, codice per la correzione degli errori –ECC)

Per utilizzare un disco come contenitore d'informazioni, il sistema operativo deve registrare le proprie strutture dati nel disco

1. suddividere il disco in partizioni
2. procedere con la formattazione logica, ossia la creazione di un file system registrandone le strutture iniziali (descrizioni dello spazio libero e assegnato, una directory iniziale vuota)

Blocco d'avviamento

Il programma d'avviamento è il programma iniziale che si esegue quando un calcolatore viene acceso e si occupa di trovare il nucleo del sistema operativo nei dischi, lo carica in memoria e salta a un indirizzo iniziale per avviare l'esecuzione del sistema operativo. Il programma d'avviamento è memorizzato in una ROM:

- non richiede inizializzazione

- ha un indirizzo iniziale dal quale la Cpu può cominciare l'esecuzione
- non può essere contaminato da virus poiché è a sola lettura

Purtroppo in questo modo per cambiare programma d'avviamento bisogna cambiare la ROM→ per questo spesso si memorizza solo un caricatore d'avviamento (bootstrap loader) che si occupa di caricare da un disco(disco d'avviamento o si sistema) il programma d'avviamento completo.

GESTIONE DELL'AREA D'AVVICENDAMENTO

La gestione dell'area d'avvicendamento è un compito di basso livello del sistema operativo. La memoria virtuale usa lo spazio dei dischi come estensione della memoria centrale → poiché l'accesso alla memoria secondaria è più lento, occorre non diminuire troppo le prestazioni del sistema. Alcuni sistemi operativi permettono l'uso di aree d'avvicendamento multiple in unità a dischi distinte→ ciò distribuisce su più dispositivi il carico della paginazione e dell'avvicendamento dei processi.
Nei sistemi che adottano l'avvicendamento dei processi si può tenere nell'area d'avvicendamento l'intera immagine, mentre nei sistemi a paginazione solo le pagine non contenute in memoria centrale.

L'area d'avvicendamento può essere

1. all'interno del normale file system → si utilizzano le normali funzioni del file system.
   Svantaggi:
   - frammentazione esterna
   - tempo d'avvicendamento elevato → si può in parte ridurre impiegando la memoria centrale come cache per le informazioni relative alla posizione dei blocchi.
2. In una partizione distinta del disco → si usa uno speciale gestore dell'area d'avvicendamento per assegnare e rimuovere i blocchi usando algoritmi che ottimizzano la velocità, ma non lo spazio occupato (può aumentare la frammentazione interna). Inoltre, se lo spazio non basta, bisogna ripartire il disco o creare un'altra area d'avvicendamento.

Alcuni sistemi operativi permettono l'uso di entrambe le collocazioni per l'area d'avvicendamento.

STRUTTURE RAID

Batterie ridondanti dischi (RAID- Redundant array of independent [inexpensive] disks)

- ✓ Per quanto riguarda l'affidabilità, la possibilità che uno dei dischi in un insieme di n dischi si guasti è più elevata della possibilità che uno specifico disco isolato si guasti → la soluzione sta nell'introdurre una certa ridondanza, ossia memorizzare informazioni che non sono normalmente necessarie, ma che si possono usare nel caso di un guasto a un disco per ricostruire le informazioni perse.
- ✓ Attraverso l'uso di più dischi è possibile migliorare la capacità di trasferimento, distribuendo i dati in sezioni su più dischi. Sezionamento dei dati (data striping)→ si distribuiscono i bit di ciascun byte su più dischi (sezionamento a livello dei bit) [es. il bit i di ciascun byte starà nel disco i, in una batteria di 8 dischi]. Il sezionamento può avvenire anche a livello dei blocchi.

  Vantaggi:
  - Aumento della produttività per accessi multipli a piccole porzioni di dati (accessi di pagine)
  - Riduzione del tempo di risposta relativo ad accessi a grandi quantità di dati.

Livelli RAID

- Livello 0 → *sezionamento a livello dei blocchi*, senza ridondanza
- Livello 1 → *copiatura speculare (mirroring o shadowing):* ogni disco logico consiste in due dischi fisici e ogni scrittura si effettua in entrambi i dischi (prima in uno e poi nell'altro, per poter garantire la coerenza nel caso il guasto si verifichi durante una scrittura)
- Livello 2 → *organizzazione per codici per la correzione degli errori (ECC-error correcting codes)*. Ogni byte di memoria ha associato un bit di parità che indica se i bit con valore1 nel byte sono pari (p=0) oppure dispari (0). In questo modo s'identificano tutti gli errori di un singolo bit. Gli schemi degli errori memorizzano 2 o più bit supplementari e possono ricostruire i dati nel caso di un singolo bit danneggiato
- Livello3→ *organizzazione con bit di parità intercalati.* Si basa sul fatto che i controllori dei dischi possono individuare se un settore è danneggiato, e quindi si può ricostruire il dato danneggiato controllando la parità dei restanti dischi [se è uguale a quella memorizzata, il bit danneggiato era 0, altrimenti 1].

  Vantaggi:

- Questo livello è altrettanto efficace del 2, e poiché utilizza un solo disco supplementare, il livello 2 non è utilizzato nella pratica
- Lettura e scrittura del byte più veloce poiché distribuito su più dischi

Svantaggi:

- È possibile fare meno operazioni di I/O al sec poiché ogni operazione coinvolge ogni disco
- Il calcolo e la scrittura del bit di parità richiede tempo → per questo molte batterie dispongono di un controllore capace di gestire il calcolo della parità per alleggerire il carico della CPU.

- Livello 4 → *organizzazione con blocchi di parità intercalati*. S'impiega il sezionamento a livello dei blocchi e si tiene un blocco di parità in un blocco separato.
- Livello 5 → *organizzazione con blocchi intercalati a parità distribuita*. Differisce dal livello 4 per il fatto che non memorizza la parità in un disco separato, ma la distribuisce tra i dischi insieme ai dati. Un blocco di parità non può contenere informazioni relative ai blocchi che risiedono nello stesso disco. Questo disco, a differenza del 4, evita un uso intensivo del disco dove risiede la parità.
- Livello 6 → *schema di ridondanza P+Q*. Simile al livello 5, memorizza però ulteriori informazioni ridondanti per poter gestire guasti contemporanei di più dischi. Invece di usare la parità, utilizza codici per la correzione degli errori come i codici Reed-Solomon.
- Livello 0+1 → Si sezionano i dati in un insieme di dischi e si duplica ogni sezione con la tecnica della copiatura speculare
- Livello 1+0 → Si fa la copia speculare dei dischi a coppie e si procede poi al sezionamento.

Il RAID di livello 0 si usa in applicazioni ad alte prestazioni in cui le perdite di dati non sono critiche, mentre il livello 1 è utilizzato per quelle applicazioni che richiedono un'alta affidabilità e un rapido ripristino. I livelli 0+1 e 1+0 sono usati dove sia prestazioni che affidabilità sono importanti (es. piccole basi di dati). Per la memorizzazione di grandi quantità di dati si utilizzano i livelli 5 o 6.

Disco di scorta (hot spare) → è un disco non impiegato per i dati, ma configurato in modo da poter usato per sostituire un disco in caso di guasto.

CONNESSIONE DEI DISCHI

I calcolatori accedono alla memoria secondaria in 2 modi:

1. memoria secondaria connessa alla macchina → si accede alle porte locali di I/O, disponibili in diverse tipologie (es. IDE o ATA)
2. memoria secondaria connessa alla rete → si accede tramite un'interfaccia RPC. Utilizza protocolli di rete (UDP o TCP) su una rete IP.

Reti di memoria secondaria (SAN- storage area network)→ rete privata che impiega protocolli specifici alla memorizzazione (anziché di rete) tra i server e le unità di memoria secondaria, separata dalla WAN o LAN che collega i server ai client. Si possono collegare alla SAN molte macchine e molte batterie di memoria e la memoria può essere assegnata dinamicamente alle macchine → flessibilità.

DISPOSITIVI PER LA MEMORIZZAZIONE TERZIARIA

Caratterizzati dal loro basso costo, sono rappresentati dai dischi rimovibili (floppy, cd…) e i nastri.

Capitolo 15

Un **sistema distribuito** è un insieme di più unità d'elaborazione che non condividono la memoria o un clock, ma ogni unità d'elaborazione ha la propria memoria locale. Le unità d'elaborazione sono debolmente connesse tramite una rete di comunicazione

I motivi principali per costruire sistemi distribuiti sono:

- Condivisione delle risorse → si possono condividere file su siti remoti per elaborare informazioni, usare dispositivi remoti ed eseguire altre operazioni.
- Accelerazione dei calcoli → se un calcolo si può dividere in sottocalcoli eseguibili concorrentemente si può ripartire tra i diversi siti. Inoltre se un sito è sovraccarico, si può condividere il carico.
- Affidabilità → se si guasta un sito, i restanti possono continuare a funzionare.
- Comunicazione → a un livello basso, tra i sistemi si scambiano informazioni in modo simile a quello dei messaggi di un singolo calcolatore. In questo modo tutti i servizi di livello superiore di un sistema centralizzato si possono estendere in modo da adattarsi al sistema distribuito (trasferimento del file, sessioni di lavoro remote, posta elettronica, RPC)

TIPI DI SISTEMI OPERATIVI (orientati alle reti)

Sistema operativo di rete → gli utenti possono accedere alle risorse remote iniziando una sessione di lavoro sugli appropriati calcolatori remoti oppure trasferendo dati dai calcolatori remoti al proprio.

Sessioni di lavoro remote

Si consente agli utenti di iniziare una sessione di lavoro a distanza in un altro calcolatore → internet fornisce la funzione `telnet`. Il processo nel calcolatore remoto chiede all'utente il suo nome d'utente e la relativa password; ricevute le corrette informazioni, il processo agisce per nome dell'utente, che può compiere le proprie elaborazioni servendosi del calcolatore remoto come un qualsiasi utente locale.

Trasferimento di file remoti

Si fornisce agli utenti un meccanismo per il trasferimento di file remoti tra calcolatori. Ogni calcolatore mantiene il proprio file system locale e se un utente di un sito vuole accedere un file che si trova su un altro calcolatore si deve copiare esplicitamente il file → internet offre il meccanismo `ftp` (file transfer protocol)

1. l'utente invoca il programma `ftp`
2. il programma `ftp` chiede nome utente e password
3. una volta che il programma ha ricevuto le corrette informazioni, l'utente deve connettersi alla directory in cui si trova il file e copiarlo.

Sistema operativo distribuito → gli utenti accedono alle risorse remote come accedono alle risorse locali. I trasferimenti di dati e processi tra siti sono sotto controllo del sistema operativo.

Migrazione dei dati

Se un utente deve accedere a dei dati di un file in un altro sito può

a) trasferire tutto il file (migrazione dei dati). A questo punto l'accesso al file è locale. Quando non serve più il file, se è stato modificato, se ne invia una copia al sito originario.
b) Trasferire solo le parti del file immediatamente necessarie. Analogamente, non appena non si utilizzerà più il file, si invieranno al sito originario solo le parti modificate

Migrazione delle computazioni

Si potrebbe voler eseguire comandi remoti

- RPC → usa un protocollo per datagrammi per far eseguire una procedura in un sistema remoto
- Un sito può inviare un messaggio al sito a cui vuol far eseguire un'elaborazione. Il destinatario creerà un nuovo processo, la cui funzione è svolgere il compito richiesto. Quando termina l'esecuzione, invia un messaggio al mittente

Migrazione dei processi

Un intero processo (o parti) si può eseguire in siti diversi per vari motivi, tra i quali:
- Bilanciamento del carico
- Accelerazione del calcolo
- Preferenza di sistemi e dispositivi → potrebbero esserci CPU specializzate per alcuni processi
- Preferenza di programmi → un processo potrebbe esser spostato in un sito dove c'è il programma adatto a eseguirlo
- Accesso ai dati → se i dati da usare nella computazione son molti, è più conveniente eseguire il processo in modo remoto piuttosto di trasferire i dati.

I processi si possono spostare con due metodi complementari:
1. nascondere che il processo è migrato → l'utente non deve codificare il programma per compiere le migrazioni (si utilizza nei sistemi omogenei per ottenere il bilanciamento del carico e l'accelerazione del calcolo)
2. l'utente deve specificare esplicitamente come il processi deve migrare (si utilizza per soddisfare una preferenza di dispositivo o di programmi).

TOPOLOGIE DI RETI

Esistono varie topologie di rete, caratterizzare da:
- costo d'installazione
- costo di comunicazione
- disponibilità

La rete può essere
- totalmente connessa
- parzialmente connessa
  - rete ad albero
  - rete ad anello
  - rete a stella

TIPI DI RETI

- Reti locali (LAN) → maggior velocità e affidabilità. Una LAN tipica è composta da diversi calcolatori, vari dispositivi periferici condivisi e uno o più gateway che danno accesso ad altre reti. Solitamente si usa una schema Ethernet → rete con bus multiaccesso, è facile aggiungere nuovi calcolatori alla rete.
- Reti geografiche (WAN) → collegamenti lenti e inaffidabili. I calcolatori sono collegati in LAN, a loro volta collegate a internet per mezzo di reti regionale, interconnesse tramite instradatori.

COMUNICAZIONE

Problemi fondamentali:

1. Nominazione e risoluzione dei nomi

In un sistema remoto i processi sono generalmente identificati dalla coppia <nome calcolatore; id > ; dove il nome calcolatore è un nome unico all'interno della rete e l'id può essere quello del processo o un altro numero unico all'interno del calcolatore. Poiché nei calcolatori è preferibile utilizzare numeri (per motivi di velocità e semplicità) bisogna ricorrere ad un meccanismo per la risoluzione del nome del calcolatore

a. Ciascun calcolatore può avere un file contenente i nomi e gli indirizzi di tutti i calcolatori della rete→ l'aggiunta o la rimozione di un calcolatore dalla rete richiede l'aggiornamento di tutti i file nei calcolatori
b. DNS (domain name system) → si distribuisce l'informazione tra i sistemi connessi alla rete, la quale adopera un protocollo per distribuire e recuperare queste informazioni.

2. Strategie d'instradamento

Se tra due siti esiste un solo percorso fisico, un messaggio deve per forza seguire quel percorso, mentre in presenza di più collegamenti ci sono più possibilità d'instradamento. Ogni sito ha una tabella d'instradamento che indica i percorsi che si possono seguire nel recapitare un messaggio ad altri siti.

a. Instradamento fisso → un percorso tra due siti viene determinato in anticipo e non cambia, se non per un guasto fisico che interrompe il percorso stesso
b. Instradamento virtuale → si fissa un percorso tra due siti per la durata di una sessione
c. Instradamento dinamico→ il percorso da impiegare per inviare un messaggio tra due siti si sceglie al momento dell'invio del messaggio.

Solitamente i calcolatori hanno un instradamento statico verso il gateway, che impiega l'instradamento dinamico per raggiungere ogni altro calcolatore nella rete.

3. Strategie riguardanti i pacchetti

I messaggi sono di lunghezza variabile, ma per semplificare la comunicazione si utilizzano messaggi di lunghezza costante (pacchetti, frame o datagrammi). La comunicazione tra mittente e destinatario può essere

a. Priva di connessione →inaffidabile
b. Con connessione → affidabile

4. Strategie di connessione

a. Commutazione di circuito → esiste un collegamento fisico tra i due siti che vogliono comunicare. Aperta una linea di comunicazione tra i due siti, nessun altro può usare il circuito finché non si termina esplicitamente la comunicazione
b. Commutazione di messaggio → si stabilisce un collegamento temporaneo per la durata del trasferimento del pacchetto. Sullo stesso collegamento si possono inviare i messaggi di utenti diversi.
c. Commutazione di pacchetto → un messaggio logico si può dividere in un dato numero di pacchetti, ciascuno dei quali si può inviare alla sua destinazione separatamente (perciò deve contenere, oltre ai dati, un indirizzo di provenienza e uno di destinazione e informazioni utili per ricomporre il messaggio, dato che arrivando per vie diverse potrebbero giungere disordinati)

5. Contesa

A seconda della topologia di rete, può accadere che più siti cerchino di trasmettere informazioni contemporaneamente

sulla stessa linea. Per evitare collisioni ripetute esistono varie tecniche.

a. CSMA/CD → prima di trasmettere un messaggio in una linea di comunicazione, un sito deve verificare che su quella linea non sia già in corso la trasmissione di un altro messaggio (ascolto della portante con accessi multipli- carrier sense with multiple access). Se due o più siti iniziano a trasmettere nello stesso istante, si ha un rilevamento di collisione (collision detection) e interrompono la trasmissione. Ciascun sito ritenta la trasmissione dopo un intervallo di tempo casuale. Questo metodo causa un calo di prestazioni quando il sistema è carico e si verificano molte collisioni, tuttavia è efficacemente utilizzato nelle reti Ethernet.
b. Passaggio di contrassegno → nel sistema, solitamente una struttura ad anello, circola continuamente un messaggio di tipo unico avente la funzione di contrassegno (token). Un sito che intenda trasmettere informazioni deve attendere finché non arriva il contrassegno, quindi lo rimuove dall'anello e inizia a trasmettere i propri messaggi; completata la trasmissione ritrasmette il contrassegno.
c. Intervalli di messaggi→nel sistema, solitamente una struttura ad anello, circolano continuamente un certo numero di 'intervalli' di messaggi (message slot) di lunghezza fissa. Un sito pronto per la trasmissione deve attendere fino all'arrivo di un intervallo vuoto, quindi inserisce il suo messaggio nell'intervallo, imposta l'opportuna informazione di controllo e lo mette nella rete. Quando arriva a un sito, questo controlla nell'informazione di controllo se è destinato a lui. In questo caso, preleva il messaggio e può impiegare l'intervallo per inviare un suo messaggio o rilasciarlo vuoto.

PROTOCOLLI DI COMUNICAZIONE

La progettazione e la realizzazione di sistemi che devono comunicare in rete è semplificata da una suddivisione in strati. Ogni strato può avere i suoi protocolli, che si possono realizzare in forma di dispositivo o possono essere programmi. Secondo il modulo OSI (open systems interconnection) dell'ISO (international standards organization) gli strati sono:

1. **Fisico**: questo strato si realizza nei dispositivi di rete. Il sistema di comunicazione deve accordarsi sulla rappresentazione elettrica delle cifre 0 e 1 in modo che il ricevitore possa interpretare correttamente il flusso di segnali elettrici.
2. **Data link:** si occupa della gestione dei frame, compresa l'individuazione degli errori.
3. **Rete**: è responsabile dei collegamenti e dell'instradamento dei pacchetti (gestione degli indirizzi in uscita, decodifica degli indirizzi dei pacchetti in arrivo, gestione delle informazioni d'instradamento)
4. **Trasporto**: è responsabile delle funzioni d'accesso a basso livello e del trasferimento dei messaggi tra i client (suddivisione dei messaggi in pacchetti, ordinamento dei pacchetti, controllo del flusso, generazione indirizzi fisici)
5. **Sessione**: è responsabile del dialogo e sincronizzazione per l'entità di applicazioni.
6. **Presentazione**: si occupa della risoluzione delle differenze di formato tra i diversi siti della rete
7. **Applicazione**: si occupa dell'interazione diretta con gli utenti (trasferimento, accesso e gestione dei file, scambio di messaggi e documenti, trasferimento e gestione dei processi)

Dal punto di vista logico, ciascuno strato di una pila di protocolli comunica con lo strato corrispondente negli altri sistemi.

Fisicamente un messaggio passa attraverso ciascuno strato più basso, che vi aggiunge un'intestazione per il corrispondente strato nel ricevente. Il messaggio raggiunge lo strato di rete della comunicazione e vine trasferito sotto forma di uno o più pacchetti. Infine lo strato data link del destinatario riceve questi dati e il messaggio risale la pila.

ROBUSTEZZA
Per assicurare la robustezza del sistema occorre individuare i guasti, riconfigurare il sistema ed effettuare un ripristino una volta aggiustato il guasto.

Per rilevare un guasto si utilizza una procedura di negoziazione (handshaking).
Il sito A manda un segnale a B per segnalare che è funzionante. Se B non riceve questo messaggio si può supporre che A sia guasto, che sia guasto il collegamento oppure che il segnale sia andato perso. A questo punto B può mandare un segnale ad A per verificare se è funzionante, utilizzando un altro percorso ed impostando un limite di tempo d'attesa della risposta (time out)

PROBLEMI DI PROGETTAZIONE

- Un sistema distribuito trasparente non deve fare distinzione tra risorse locali e remote
- Per quanto riguarda la mobilità, è opportuno permettere ad ogni utente di aprire sessioni in qualsiasi calcolatore del sistema
- Un sistema tollerante ai guasti deve continuare a funzionare, sia pure in modo ridotto, se i guasti sono in misura tollerabili
- La capacità di un sistema di adattarsi a un maggior carico di servizio è la scalabilità. In un sistema scalabile, quando si presenta un maggiore carico, le prestazioni degradano moderatamente e le sue risorse raggiungono la saturazione più tardi. [tolleranza ai guasti e scalabilità sono interdipendenti e richiedono una progettazione caratterizzata dalla distribuzione del controllo e dei dati]
- L'approssimazione pratica di una configurazione simmetrica e autonoma è rappresentata dalla strutturazione del sistema in batterie di calcolatori (clustering) → si suddivide il sistema in un insieme di batterie semi-autonome di calcolatori, ciascuna delle quali è a sua volta composta da un insieme di calcolatori e di server. Le richieste dei calcolatori di una batteria dovrebbero essere soddisfatte per la maggior parte delle volte dal relativo server.
- Per quanto riguarda la migliore struttura dei processi server è quella a thread.

Capitolo 16

Un file system distribuito (distribuited file system –DFS) è una realizzazione distribuita del modello classico di un file system in un sistema a partizione di tempo dove più utenti condividono file e risorse di memorizzazione.

Servizio: programma in esecuzione in una o più macchine e fornisce un tipo particolare di funzione a client sconosciuti a priori.
Server: programma di servizio in esecuzione in una singola macchina
Client: processo che può richiedere un servizio
Interfaccia del client: serie di operazioni di cui si serve il client per richiedere un servizio
Interfaccia intermacchina: interfaccia di livello inferiore per l'interazione tra le macchine.

NOMINAZIONE E TRASPARENZA

Generalmente l'utente fa riferimento a un file attraverso un nome testuale. A quest'ultimo si fa corrispondere un identificatore numerico di livello inferiore che a sua volta si fa corrispondere ai blocchi dei dischi. Quest'associazione a più livelli fornisce agli utenti un'astrazione del file che nasconde i particolari concernenti la sua memorizzazione. In un DFS trasparente, all'astrazione si aggiunge un nuovo aspetto: nascondere la posizione all'interno della rete.

- Trasparenza di locazione: il nome di un file non rivela alcun indizio sulla sua locazione fisica.
- Indipendenza dalla locazione: non si deve modificare il nome di un file se cambia la sua locazione in memoria fisica.

Si differenziano per:

- La separazione dei dati dalla locazione offre una migliore astrazione per i file (come nell'indipendenza dalla locazione). Se è prevista solo la trasparenza di locazione statica, il nome del file continua a indicare un gruppo specifico, anche se nascosto, di blocchi fisici dei dischi.
- Con la trasparenza di locazione statica gli utenti possono condividere file remoti come se fossero locali. La condivisione dello spazio di memorizzazione è scomoda perché i nomi logici sono vincolati in modo statico ai dispositivi fisici. L'indipendenza di locazione consente la condivisione dello spazio di memorizzazione stesso e degli oggetti di dati
- L'indipendenza dalla locazione separa la gerarchia di nominazione dalla gerarchia dei dispositivi di memorizzazione e dalla struttura esistente tra i calcolatori. Al contrario, usando la trasparenza di locazione statica, si può evidenziare la corrispondenza tra unità componenti e macchine anche se i nomi sono trasparenti.

Esistono 3 schemi principali per la dominazione di un file in un DFS:

1. nominare il file per mezzo di una combinazione del suo nome di macchina e del nome locale → non gode né della trasparenza di locazione, né dell'indipendenza dalla locazione.
2. si possono offrire meccanismi per unire le directory remote alle directory locali, dando all'utente l'impressione di un albero di directory coerente → si può gestire la condivisione trasparente, sebbene sia limitata dal fatto che l'integrazione dei componenti non è uniforme, poiché ogni macchina può aggiungere diverse directory remote al proprio albero, creando una struttura incostante
3. una sola struttura globale dei nomi si estende a tutti i file del sistema. In teoria, la struttura composta del file system ha la stessa forma della struttura di un file system convenzionale; in pratica esistono molti file speciali (es. quelli che rappresentano i dispositivi) che rendono difficile il conseguimento di questo scopo.

ACCESSO AI FILE REMOTI

Meccanismo del servizio remoto: le richieste d'accesso sono inviate al server, che esegue gli accessi e ritrasmette i risultati all'utente. (uno dei metodi più diffuso per realizzarlo è il paradigma RPC).
Per assicurare prestazioni ragionevoli si usano delle cache che tengono in memoria blocchi di disco a cui si è fatto riferimento di recente. In questo modo, se i dati necessari a soddisfare la richiesta d'accesso non sono già stati copiati nella cache, si trasferisce una copia di quei dati dal server al sistema client. Gli accessi richiesti si eseguono alla memoria cache.
Problema di coerenza della cache: se si modifica una copia contenuta in una cache, le modifiche devono essere riportate anche sulla copia principale in modo da conservare la coerenza dei dati.
L'utilizzo della cache in un DFS si può chiamare memoria virtuale di rete.
La dimensione dei dati unitaria da trasferire nella cache dipende dall'unità di trasferimento della rete e l'unità di servizio delle RPC; una dimensione ampia aumenta il numero di successi, ma aumento anche il costo di ogni insuccesso nella ricerca nella cache.

Locazione delle cache

Su dischi:

- sono più affidabili

In memoria centrale:

- permettono l'uso di stazioni di lavoro prive di dischi
- si accede più rapidamente ai dati
- attualmente si producono memorie centrali più grandi e meno costose
- poiché i server utilizzano cache in memoria centrale, utilizzando cache in memoria centrale anche nei client si potrebbe realizzare un unico meccanismo di gestione.

Criteri di aggiornamento delle cache

1. Scrittura diretta → scrivere direttamente i dati nei dischi non appena questi vengono modificati in una cache qualsiasi. Questo garantisce un'alta affidabilità, a scapito di scarse prestazioni.
2. Scrittura differita → si differiscono gli aggiornamenti della copia principale, modificando prima la cache e poi il server.
    a. Gli accessi in scrittura sono più rapidi, poiché diretti alla cache
    b. Si possono sovrascrivere i dati prima che siano riportati al server, rendendo necessario riportare solo l'ultimo aggiornamento

    Con questo metodo risente di una scarsa affidabilità → se si verifica un guasto, i dati non scritti vengon persi.
    Esistono delle varianti:

    - Inviare un blocco quando questo sta per essere espulso dalla cache del client (migliora le prestazioni, ma accade che alcuni blocchi rimangano a lungo nella cache del client prima di esser riscritti nel server)
    - Un compromesso con la scrittura diretta è esaminare la cache a intervalli regolari e inviare al server solo i blocchi modificati dopo l'ultima verifica.
    - Scrittura su chiusura → si scrivono i dati nel server quando si chiude un file. Risulta particolarmente efficace quando i file rimangono aperti per lunghi periodi e vengono modificati spesso.

Esistono 2 metodi per verificare se una copia in una cache è coerente con la copia principale:

1. Metodo iniziato dal client→ il client inizia un controllo di validità mettendosi in contatto col server. A seconda della frequenza in cui avviene questo controllo, si può caricare sia la rete che il server
2. Metodo iniziato dal server→ il server registra, per ogni client, i file (o le parti di file) che vengono aperti e in che modo (lettura/scrittura). Quando il server individua un file che viene aperto in modo conflittuale, disabilita la possibilità di copiare quel file nella cache, passando a un funzionamento remoto.

USO DELLA CACHE	USO DEL SERVIZIO REMOTO
- Si gestisce un consistente numero di accessi remoti→ riduce il carico dei server e della rete, aumenta la scalabilità	- Ogni accesso remoto è gestito dalla rete
- Trasmette grandi quantità di dati alla volta	- Risponde a specifiche richieste → maggior carico della rete
- L'inconveniente principale è la coerenza→conveniente se le scritture sono poche	
- Adatta per macchine con dischi locali oppure grandi memorie centrali	- Conveniente con macchine prive di dischi e ridotta capacità di memoria
- L'interfaccia intermacchina inferiore è diversa dall'interfaccia utente superiore	- L'interfaccia tra le macchine rispecchia l'interfaccia locale tra l'utente e il file system

Relativamente alle informazioni del server, esistono due metodi:

1. **servizio con informazioni di stato**→ prima di accedere a un file, il client esegue una open. Il server preleva dai suoi dischi informazioni sul file, le registra nella propria memoria e fornisce al client un identificatore unico di connessione. Alla chiusura del file, il server può reclamare lo spazio di memoria centrale per client inattivi. La tolleranza ai guasti si basa sul fatto che il server conserva in memoria centrale informazioni sui suoi client.
    - migliori prestazioni
    - in caso di crollo del sistema il server perde completamente il proprio stato volatile→per un ripristino conveniente bisogna recuperarlo

    - se il server utilizza il metodo iniziato dal server per la validazione della cache deve per forza essere con informazioni di stato (poiché registra quali file sono stati copiati nella cache di quali client)
2. **servizio senza informazioni di stato** → ogni richiesta identifica completamente il file e la sua posizione. Il server non deve tenere in memoria centrale una tabella dei file aperti (anche se in realtà avviene per motivi di efficienza). Ogni operazione su file è indipendente e non si considera parte di una sessione (non è necessario stabilire e terminare una connessione e le letture/scritture avvengono come messaggi remoti o come ricerche nella cache)
    - i messaggi di richiesta son lunghi ed è necessario uno schema di denominazione uniforme a basso livello su tutto il sistema (per tradurre i nomi remoti in locali)→elaborazione delle richieste più lenta
    - in caso di guasto, una volta riparato risponde senza difficoltà alle nuove richieste, poiché indipendenti
    - le operazioni devono essere idempotenti, perché il client ritrasmettono richieste per operazioni su file.

REPLICAZIONE DEI FILE

- La replicazione di file tra macchine diverse è utile per migliorare la disponibilità.
- Repliche dello stesso file devono stare in macchine indipendenti → assenza di trasparenza di locazione.
- Spetta allo schema di dominazione associare un nome di file replicato a una replica particolare.
- Le repliche devono essere invisibili ai livelli superiori, ma devono essere differenziate le une dalle altre ai livelli più bassi
- Dal punto di vista dell'utente le repliche di un file indicano la stessa entità logica → l'aggiornamento di una replica deve riflettersi su tutte le altre copie

Capitolo 17

Un sistema distribuito non ha memoria e clock comuni, quindi talvolta è impossibile stabilire l'ordine degli eventi. Poiché definire un ordinamento totale è fondamentale per molte applicazioni, si è studiato un algoritmo basato sulla **relazione verificato-prima** (→)

1. Poiché si considerano solo processi sequenziali, tutti gli eventi eseguiti da un processo sono totalmente ordinati. Quindi se A e B sono eventi dello stesso processo ed A è stato eseguito prima di B, allora A→B
2. Se A corrisponde all'invio di un messaggio da parte di un processo e B corrisponde alla ricezione del messaggio da parte di un altro processo, allora A→B.
3. Se A→B e B→C, allora A → C
4. Poiché un evento non può accadere prima di se stesso, questa relazione non è riflessiva
5. Se A e B non sono correlati dalla relazione →, sono stati eseguiti concorrentemente, quindi nessuno dei 2 può influire in modo casuale sull'altro.

Realizzazione
Per ottenere un ordinamento totale in un ambiente distribuito, per ciascun processo Pi si definisce un clock logico LCi, il cui valore corrisponde a una marca temporale (timestamp) [se l'evento A avviene prima del B, allora LCi(A) < LCi(B)].
Questo schema però non assicura che il requisito d'ordinamento totale sia soddisfatto da processi diversi, in quanto i clock logici possono avere velocità differenti: questo si può risolvere facendo avanzare il clock logico di un processo quando questi riceve un messaggio con una marca temporale maggiore del valore attuale del proprio clock logico.

Se le marche temporali di due eventi sono uguali, questi sono concorrenti: in questo caso, per evitare ambiguità, si utilizza l'id dei processi per creare l'ordinamento totale.

MUTUA ESCLUSIONE

Si suppone che i processi siano numerati in modo unico da 1 a n e che esista una corrispondenza da uno a uno tra processi e unità di elaborazione

Metodo centralizzato
Si sceglie uno dei processi (coordinatore) del sistema per coordinare l'accesso alle sezioni critiche.

1. un processo che necessita della mutua esclusione invia un messaggio di richiesta al coordinatore
2. Il coordinatore controlla se qualche altro processo si trova nella propria sezione critica
    a. Se c'è un altro processo, la nuova richiesta viene accodata, altrimenti
    b. Il coordinatore invia un messaggio di risposta al processo richiedente
3. Quando riceve il messaggio di risposta, il processo esegue la propria sezione critica ed infine invia al coordinatore un messaggio di rilascio
4. Il coordinatore a questo punto può prelevare la richiesta di un processo in coda, scegliendo in base al proprio algoritmo di scheduling.

Metodo totalmente distribuito

Quando un processo Pi vuole entrare nella propria sezione critica, genera una marca temporale TS e invia una messaggio request(Pi,TS) a tutti gli altri processi del sistema, compreso se stesso. Quando un processo riceve questo messaggio di richiesta, può inviare immediatamente un messaggio di risposta, o differirne l'invio nel caso in cui:

a) Si trova nella propria sezione critica
b) Se, volendo anch'esso entrare nella propria sezione critica, ha una marca temporale inferiore a Pi (ha quindi fatto prima la richiesta)

Un processo può entrare nella propria sezione critica quando ha ricevuto un messaggio di risposta da tutti gli altri processi del sistema.

Vantaggi

- Si ottiene la mutua esclusione
- Assenza di situazioni di stallo e di attesa indefinita
- Il numero dei messaggi per ogni entrata in una sezione critica è il minimo richiesto quando i processi agiscono in modo indipendente e concorrente [2 x (n-1)]

Svantaggi

- I processi devono conoscere l'identità di tutti gli altri processi, per cui quando si aggiunge un nuovo processo bisogna:
    - Inviare al nuovo processo i nomi degli altri del gruppo
    - Inviare a tutti i processi del gruppo il nome del nuovo processo
- Se s'interrompe un processo, crolla tutto il sistema
- I processi che non sono entrati nelle rispettive sezioni critiche devono interrompersi spesso per dichiarare agli altri processi che intendono entrare nella propria sezione critica.

Questo protocollo è adatto a insiemi piccoli e stabili di processi cooperanti

ATOMICITA'

Direttamente dagli appunti di basi di dati:

Per garantire l'atomicità delle transazioni distribuite è necessario che tutti i nodi che partecipano ad una transazione giungano alla stessa decisione circa la transazione (commit o abort) → per far ciò si utilizzano particolari protocolli, detti di commit.

Protocollo two-phase commit

✶ Record del Transaction Manager (coordinatore)
✶ Record dei Resource Manager (siti)

Prima fase del protocollo

- Il TM scrive il record prepare nel suo log e manda un messaggio prepare a tutti gli RM. Imposta un timeout indicando il massimo tempo allocato al completamento della prima fase.
- Gli RM in stato affidabile scrivono nel loro log il record ready e trasmettono al TM il messaggio ready, che indica la scelta di partecipare al protocollo
- Gli RM in stato non affidabile mandano un messaggio non-ready e terminano il protocollo
- Il TM raccoglie i messaggi di risposta
    - se riceve un messaggio positivo da tutti gli RM, scrive un record global commit nel log
    - se uno o più dei messaggi ricevuti è negativo o non tutti i messaggi sono ricevuti entro il time-out, il TM scrive un record global abort nel log

Seconda fase del protocollo

- Il TM trasmette la sua decisione globale a tutti gli RM. Imposta poi un time-out.
- Gli RM che sono in uno stato ready ricevono la decisione, scrivono il record relativo (commit o abort) nel loro log e mandano un acknowledgement (ack) al TM. Poi eseguono il commit/abort scrivendo le pagine al database.
- IL TM raccoglie tutti gli ack dagli RM coinvolti nella seconda fase. Se il time-out scade, il TM stabilisce un altro time-out e ripete la trasmissione a tutti gli RM dai quali non ha ricevuto ack
- Quando tutti gli ack sono arrivati, il TM scrive il record complete nel suo log

La correttezza del protocollo di commit a due fasi può essere compromessa da vari cause d'errore → per rimediare a tali errori si ricorre a dei **protocolli di ripristino**

Caduta e ripristino di un partecipante

✓ Eseguito dal protocollo di ripresa a caldo. Dipende dall'ultimo record scritto nel log
  - se il record è una azione o un abort, le azioni sono disfatte (undo); se è commit, le azioni sono rifatte (redo). In entrambi i casi il guasto è successo prima di iniziare il protocollo di commit
  - se l'ultimo record scritto nel log è un ready, il guasto è successo durante il two-phase commit. Il partecipante è incerto sul risultato della transazione
✓ Durante la ripresa a caldo, gli identificatori delle transazioni in dubbio sono raccolte in un insieme ready. Per ognuna di queste la decisione finale deve essere richiesta al TM. Alternativamente può essere richiesta direttamente dal TM all'RM (recovery remoto)a seguito di una ripetizione della seconda fase del protocollo

Caduta e ripristino del coordinatore

✓ se l'ultimo record nel log è un prepare, il guasto del TM potrebbe aver messo alcuni RM in uno stato di blocco. Due opzioni di recovery:
  - Scrivere un global abort nel log, e poi eseguire la seconda fase del protocollo
  - ripetere la prima fase cercando di arrivare a un commit globale
✓ se l'ultimo record di log è una decisione globale, alcuni RM possono essere stati correttamente informati mentre altri essere ancora bloccati. Il TM deve ripetere la seconda fase.

CONTROLLO DELLA CONCORRENZA

Protocolli d'accesso bloccante
Si suppone l'esistenza dei modi d'accesso bloccante condiviso ed esclusivo

◊ Metodo con coordinatore singolo
Se si impiega un solo coordinatore il sistema mantiene un singolo gestore dei diritti d'accesso bloccante che risiede in un singolo sito (S). Tutte le richieste d'accesso bloccante e di rilascio s'inoltrano al sito S. Quando riceve una richiesta d'accesso S stabilisce se tale richiesta si può soddisfare immediatamente o se deve essere differita. Una transazione può leggere l'elemento da qualsiasi sito in cui risiede una replica del dato, mentre in caso di scrittura tutte le repliche devono essere coinvolte
Vantaggi
- Semplice realizzazione
- Semplice gestione delle situazioni di stallo (gli algoritmi di gestione delle situazioni di stallo si applicano a S)

Svantaggi
- S è una strozzatura, poiché deve elaborare tutte le richieste
- Vulnerabilità: se si guasta S, manca interamente il controllore della concorrenza

Un compromesso tra vantaggi e svantaggi si può ottenere impiegando un metodo con coordinatori multipli → ci sono più gestori; ogni gestore amministra le richieste d'accesso bloccante e di rilascio di un sottoinsieme di dati e risiede in un sito diverso

◊ Protocollo di maggioranza
Il sistema mantiene un gestore dei diritti d'accesso bloccante in ciascun sito, che si occupa dei dati e delle repliche memorizzate nel sito stesso.
Quando una transazione desidera accedere in modo bloccante a un elemento, invia una richiesta a più della metà dei siti in cui l'elemento risiede e lo otterrà quando avrà il diritto di accesso bloccante sulla maggioranza delle repliche dell'elemento. Gestendo i dati replicati in modo decentralizzato, questo metodo elimina gli inconveniente tipici dei sistemi centralizzati.

Svantaggi
- Realizzazione complessa
- Poiché le richieste non vengono inoltrate a uno stesso sito, gli algoritmi di gestione delle situazioni di stallo devono essere modificati.

◊ Protocollo sbilanciato
Il sistema mantiene un gestore dei diritti d'accesso bloccante in ciascun sito, che si occupa dei dati e delle repliche memorizzate nel sito stesso.
Le richieste si distinguono in:
a. Accesso bloccante condiviso: una transazione si limita a inoltrare una richiesta al gestore dei diritti d'accesso bloccante di un sito contenente una replica dell'elemento
b. Accesso bloccante esclusivo: una transazione inoltra la richiesta d'accesso bloccante a tutti i siti contenenti una replica dell'elemento.

Rispetto al protocollo di maggioranza c'è minor carico per le operazioni di lettura, ma uno maggiore per quelle di scrittura

Uso delle marche temporali

Esistono due modi per generare marche temporali uniche
1. Metodo centralizzato → le marche vengono distribuite da un singolo sito, che può adottare per lo scopo un contatore logico o il suo clock locale
2. Metodo distribuito → ogni sito genera una marca temporale unica locale. La marca temporale unica globale si ottiene concatenando <marca locale, identificativo del sito>

GESTIONE DELLE SITUAZIONI DI STALLO

Prevenzione
Si possono adattare degli algoritmi per la prevenzione di situazioni di stallo in un sistema distribuito. Per esempio, si può definire un ordinamento totale tra le risorse del sistema e impedire a un processo di ottenere una risorsa se è già in possesso di un'altra risorsa con identificativo più alto; oppure utilizzare l'algoritmo del banchiere, scegliendo uno dei processi del sistema (il banchiere) per mantenere le informazioni necessarie

Si può adottare uno schema basato su un ordinamento delle marche temporali con diritto di prelazione delle risorse. Per controllare il diritto di prelazione si assegna a ogni processo un unico numero di priorità. Le marche temporali si utilizzano invece per evitare le situazioni d'attesa indefinita, secondo due metodi complementari

a) Schema attesa-morte: quando un processo richiede una risorsa correntemente posseduta da un altro processo, viene messo in attesa solo se ha una marca temporale inferiore a quella dell'altro processo; altrimenti viene annullato
   - Il processo più vecchio deve attendere che un processo più giovane rilasci la risorsa
   - Un processo può morire più volte prima di acquisire la risorsa
b) Schema ferita-attesa: quando un processo richiede una risorsa correntemente posseduta da un altro processo, viene messo in attesa solo se ha una marca temporale maggiore; altrimenti si sottrae la risorsa al processo che la possiede attualmente, annullandone l'esecuzione; e la si assegna al nuovo processo
   - Un processo più vecchio non attende mai un processo più giovane
   - Si hanno meno annullamenti→ dopo esser annullato, un processo è messo in attesa

Quando un processo subisce un annullamento, mantiene la sua marca temporale

Rilevamento

Ogni sito ha un grafo d'attesa locale→ i nodi corrispondono a tutti i processi (locali e non) correntemente in possesso di

qualche risorsa locale di quel sito, o che la stanno chiedendo.

Se c'è un ciclo in un grafo locale, si è verificato uno stallo, ma il fatto che non ci siano cicli in ogni singolo grafo locale non significa che non ci siano stalli. Per provare che non si è verificato nessuno stallo bisogna dimostrare che l'*unione* dei grafi è aciclica.

◊ Metodo centralizzato

Un unico processo (coordinatore per il rilevamento delle situazioni di stallo) mantiene un grafo globale ottenuto unendo tutti i grafi d'attesa locali. Essendoci un ritardo di comunicazione, esistono due tipi di grafo d'attesa:

- Reale: descrive lo stato reale, ma sconosciuto

Costruito: approssimazione generata dal coordinatore.

Il grafo si può costruire:

a) ogni volta che s'inserisce o rimuove un nuovo arco in uno dei grafi d'attesa locali
b) periodicamente, quando si verifica un certo numero di cambiamenti nel grafo d'attesa.
c) Ogni volta che il coordinatore deve invocare l'algoritmo di rilevamento dei cicli.

Il grafo costruito deve essere corretto, ossia rispettare queste 2 proprietà:

1. se esiste uno stallo, la sua esistenza viene correttamente riportata
2. se si indica uno stallo, il sistema è effettivamente in stallo, infatti è possibile che
   a. esistano cicli falsi
   b. si possono compiere annullamenti non necessari anche quando si è effettivamente verificato uno stallo ed è stata scelta una vittima, ma contemporaneamente l'esecuzione di uno dei processi è stata terminata, per motivi indipendenti dallo stallo stesso.

Per evitare di indicare cicli falsi, è necessario che nei grafi d'attesa locali la prima richiesta proveniente da un altro sito sia indicata con un'etichetta. Il controllore costruisce il grafo creando un nodo per ogni processo del sistema e mette un arco se e solo se c'è un arco in uno dei grafi d'attesa oppure un arco con etichetta compare in più di un grafo d'attesa.

A questo punto, se c'è un ciclo, il sistema è in stallo→ si sceglie una vittima e si informano tutti i siti è stato scelto come vittima, di modo che ne annullino l'esecuzione.

◊ Metodo totalmente distribuito

Tutti i controllori condividono in eguale misura la responsabilità del rilevamento delle situazioni di stallo. Ogni sito costruisce un grafo d'attesa locale, utilizzando un nodo P_{ex} quando un processo locale chiede una risorsa di un altro sito, oppure quando un altro sito richiede una risorsa locale.

- Se in un grafo locale c'è un ciclo che non coinvolge questo nodo, allora c'è uno stallo
- Se in un grafo locale c'è un ciclo che comprende $P_{ex,}$ bisogna impiegare un algoritmo distribuito per il rilevamento delle situazioni di stallo
   a) si invia un messaggio al sito interessato
   b) questi controlla se c'è un ciclo nel proprio grafo di attesa locale

i. se no, non c'è stallo
ii. se si e non comprende in nodo $P_{ex,}$ c'è stallo
iii. se si e comprende il nodo $P_{ex,}$ → a)

ALGORITMI DI ELEZIONE

Può accadere che il coordinatore si guasti e sia necessario eleggerne uno nuovo. A ogni processo è assegnata una priorità unica, e il coordinatore è sempre il processo col numero di priorità maggiore→ se questo si guasta, il processo eletto per sostituirlo deve essere quello con priorità maggiore.

Algoritmo dello spaccone

Quando un processo invia una richiesta al coordinatore e non riceve risposta dopo un intervallo di tempo t, suppone che questo sia guasto e tenta di eleggersi nuovo coordinatore.

1. invia a tutti i processi con priorità maggiore della propria un messaggio di elezione
    a. se riceve una risposta, si mette in attesa per un tempo t' di riceve un messaggio dell'avvenuta elezione di un processo a priorità maggiore
        i. se lo riceve, registra l'informazione
        ii. se non lo riceve, riavvia l'algoritmo
    b. se non riceve alcuna risposta, suppone che tutti i processi con priorità maggiore alla sua siano guasti, e si elegge nuovo coordinatore
2. Il nuovo coordinatore invia un messaggio per informare tutti i processi attivi (ossia quelli con priorità inferiore alla propria, essendo quello a priorità maggiore funzionante).

Appena un processo guasto viene ripristinato, si riavvia l'algoritmo, di modo che se ha priorità maggiore dell'attuale coordinatore ne prende il posto→ da qui deriva il nome dell'algoritmo.

Algoritmo ad anello

Si basa sul fatto che i collegamenti sono unidirezionali e si utilizza una lista attiva per ogni processo.

1. quando si rileva un guasto del coordinatore, un processo crea una nuova lista attiva (inizialmente vuota), invia un messaggio di elezione con proprio numero al suo vicino e aggiunge la sua priorità alla lista.
2. quando riceve questo messaggio, un processo:
    a. se è il primo messaggio ricevuto o inviato, crea una nuova lista contenente le due priorità (la propria e quella del processo che gli ha inviato il messaggio), quindi invia un messaggio di elezione con il proprio numero e quello ricevuto.
    b. se il messaggio ricevuto non contiene il proprio numero, il processo lo aggiunge alla sua lista attiva e inoltra il messaggio
    c. se il messaggio ricevuto contiene il proprio numero, allora la lista attiva del processo contiene tutti i processi attivi nel sistema. In questo modo, può determinare il numero maggiore tra essi e identificare il nuovo coordinatore.

RAGGIUNGIMENTO DI UN ACCORDO

Affinché un sistema sia affidabile deve esistere un meccanismo che permetta a un insieme di processi di accordarsi su un valore comune. Tale accordo può non esserci per diversi motivi:

- comunicazione inaffidabile: può succedere che uno dei siti si guasti, e per questo un processo attenda invano una risposta. Si potrebbe impostare un tempo di attesa della risposta, allo scadere del quale il processo rinvia il messaggio, finchè non riceve risposta o gli arriva un messaggio che segnale il guasto del sito da cui attende risposta.
- Processi difettosi: anche se il mezzo di comunicazione adottato è affidabile, può accadere che uno o più processi siano difettosi. Per verificare ciò, si può adottare un algoritmo che costruisce un vettore con tutti i processi non difettosi. A ogni processo è associato un valore privato, che viene inserito nella posizione corrispondente al processo nel vettore di ogni processo. Processi non difettosi avranno lo stesso vettore.

Capitolo 18

Protezione: meccanismo per il controllo dell'accesso alle risorse definite da un sistema di calcolo da parte di programmi, processi o utenti.

Motivi:
1. Prevenire la violazione intenzionale e dannosa di un vincolo d'accesso da parte di un utente
2. Assicurare che ogni componente del programma attivo in un sistema impieghi le risorse del sistema in modo coerente con i criteri stabiliti
3. Migliorare l'affibabilità rilevando errori latenti nelle interfacce tra sottosistemi componenti.

Un sistema di protezione deve essere flessibile.

DOMINI DI PROTEZIONE

Un sistema di calcolo è un insieme di processi ed oggetti.
Oggetti → elementi sia fisici che logici. Hanno un nome unico che li distingue l'uno dall'altro.
Principio del privilegio minimo → un processo deve poter accedere alle sole risorse per le quali ha l'autorizzazione e di cui ha correntemente bisogno per eseguire il proprio compito
Dominio di protezione →specifica le risorse cui il processo può accedere e i tipi di operazioni che si possono compiere

su di esse (diritti di accesso9. Il dominio è formato da coppie ordinate del tipo <nome oggetto, insieme dei diritti >

I domini non devono necessariamente essere disgiunti → si possono condividere diritti di accesso

L'associazione tra processo e dominio può essere

- Statica → l'insieme delle risorse disponibili per un processo è fissato per tutta la durata del processo. Per aderire al principio del privilegio minimo si deve disporre di un meccanismo che permetta di modificare il contenuto di un dominio.
- Dinamica → le risorse vengono associate ad un processo in maniera dinamica. Deve essere disponibile un meccanismo per passare da un dominio all'altro.

Esistono diversi modi per realizzare un dominio

- Ogni utente può essere un dominio. Il cambio di dominio avviene quando cambia l'utente
- Ogni processo può essere un dominio. Il cambio di dominio corrisponde all'invio di un messaggio da un processo a un altro processo e quindi l'attesa di una risposta
- Ogni procedura può essere un dominio. Il cambio di dominio avviene quando s'invoca una procedura.

MATRICE D'ACCESSO

E' un modello di protezione, in cui le righe della matrice rappresentano i domini e le colonne gli oggetti. Ciascun elemento della matrice consiste in un insieme di diritti d'accesso. Occorre assicurare che un processo in esecuzione nel dominio Di possa accedere solo agli oggetti specificati nella riga i e solo nel modo indicato dagli elementi della matrice d'accesso.

La matrice d'accesso fornisce un meccanismo adeguato alla definizione e realizzazione di uno stretto controllo sia per l'associazione statica sia per quella dinamica tra processi e domini. Quando un processo passa da un dominio all'altro si

esegue un'operazione (switch) su un oggetto (il dominio). Il passaggio da un dominio all'altro si può controllare inserendo i domini tra gli oggetti della matrice d'accesso→quando si modifica il contenuto della matrice d'accesso, si esegue un'operazione su un oggetto: la matrice d'accesso.

Permettere la modifica controllata del contenuto degli elementi della matrice d'accesso richiede tre operazioni.

- *Copy* → permette di copiare il diritto d'accesso solo all'interno della colonna (cioè per l'oggetto) per la quale il diritto stesso è definito-
  - *Copy*: il diritto viene copiato e questa copia può a sua volta 'propagare' il diritto
  - *Transfer*: il diritto viene copiato da access (i,j) a access (k,j) e viene successivamente rimosso da access (i,j)
  - *Limited copy*: la propagazione del diritto copy può essere limitata, ossia il diritto 'copiato' non può esserlo a sua volta
- *Owner* → permette di aggiungere o rimuovere qualsiasi elemento della colonna j
- *Control* → permette di modificare gli elementi di una riga

Problema della reclusione → garantire che nessuna informazione tenuta inizialmente in un oggetto possa migrare all'esterno del proprio ambiente d'esecuzione.

Realizzazione della matrice d'accesso

◊ Tabella globale: si può realizzare la matrice 'accesso come una tabella costituita da triple del tipo <dominio, oggetto, insieme dei diritti>. Ogni volta che si vuole eseguire una data operazione su un oggetto di un dominio, si controlla se si trova la tripla corrispondente; se cià non avviene, si verifica una condizione di eccezione (errore). Svantaggio: la tabella è troppo grande per esser tenuta in memoria centrale; bisogna quindi ricorrere ad ulteriori richieste di I/O

◊ Liste d'accesso per oggetti: per ogni oggetto si crea una lista composta da coppie ordinate <dominio, insieme dei diritti> . Si può definire anche un insieme di diritti d'accesso predefinito. Quando si tenta un'operazione su un oggetto, si controlla prima nell'insieme predefinito e poi nella lista dell'oggetto.

◊ Liste delle abilitazioni per domini: abilitazione → nome fisico, o indirizzo, che rappresenta un oggetto. Una lista di questo tipo associa ogni riga della matrice al suo dominio. Per ogni dominio c'è una lista di oggetti insieme con le operazioni ammesse. La lista delle abilitazioni è associata ad un dominio, ma non è mai direttamente accessibile a un processo che si trova in esecuzione in quel dominio: è un oggetto protetto, mantenuto dal sistema operativo e al quale gli utenti possono accedere solo indirettamente.Per distinguere le abilitazioni dagli altri oggetti è possibile:
  - Associare ad ogni oggetto un'etichetta (tag) che indica il tipo → abilitazione o dati
  - Lo spazio d'indirizzi associato ad un programma si può dividere in due parti: una contenente i dati e le istruzioni del programma (accessibile al programma); l'altra, contenente la lista delle abilitazioni, accessibile solo al sistema operativo.

◊ Meccanismo chiave serratura: è un compromesso tra le liste d'accesso e le liste delle abilitazioni. Ogni oggetto ha una lista di sequenze di bit uniche (serrature); analogamente, ogni dominio ha una lista di sequenze di bit uniche (chiavi). Un processo in esecuzione in un dominio può accedere a un oggetto solo se quel dominio ha una chiave che corrisponde a una delle serrature dell'oggetto.

REVOCA DEI DIRITTI D’ACCESSO

In un sistema di protezione dinamica può talvolta essere necessario revocare i diritti d’accesso ad oggetti condivisi da diversi utenti.

PROTEZIONE BASATA SUL LINGUAGGIO

Il grado di protezione fornito negli attuali sistemi di calcolo è di solito ottenuto attraverso il nucleo del sistema operativo. Poiché una completa convalida degli accessi è potenzialmente una fonte di notevole sovraccarico, è necessario disporre dell’ausilio dell’architettura. Attualmente i sistemi di protezione non riguardano solo l’identità della risorsa alla quale si tenta di accedere, ma anche alla natura funzionale di quell’accesso. La protezione non è un interesse esclusivo del progettista del sistema operativo, ma anche di quello di applicazioni.

Capitolo 19

La **sicurezza** non richiede solo un adeguato sistema di protezione, ma anche la considerazione dell'ambiente esterno nel quale opera il sistema. Un sistema è sicuro se le sue risorse si adoperano e vi si accede soltanto nei modi previsti.

Gli abusi del sistema possono essere:

- ◊ Accidentali
- ◊ Intenzionali (dolosi)
  - o Letture non autorizzate dei dati (furto di informazioni)
  - o Alterazione non autorizzata dei dati
  - o Distruzione non autorizzata dei dati
  - o Impedimento del legittimo uso del sistema (rifiuto di servizio)

Per proteggere il sistema è necessario prendere misure di sicurezza a 4 livelli:
1. Fisico: impedire l'accesso agli intrusi al sito dov'è ospitato il sistema di calcolo
2. Umano: controllare gli utenti autorizzati affinché non forniscano informazioni a terzi
3. Rete: impedire l'intercettazione di dati da parte di altri calcolatori
4. Sistema operativo: il sistema deve auto-proteggersi dalle violazioni della sicurezza.

AUTENTICAZIONE DEGLI UTENTI

Normalmente un utente identifica se stesso, quindi si tratta di stabilire se l'identità di un utente è autentica. L'autenticazione si basa su uno o più di questi metodi:
- ◊ Oggetti posseduti dall'utente
- ◊ Attributo dell'utente
- ◊ Conoscenza dell'utente → il metodo più diffuso prevede l'utilizzo di password, che l'utente deve immettere dopo l'identificazione.Pur essendo un modo facile da capire e usare per autenticare un utente, risulta vulnerabile, in quando una password può essere scoperta o rivelata a terzi.

MINACCE AI PROGRAMMI

Cavallo di Troia→ segmento di codice che abusa del suo ambiente
Trabocchetto → è un 'buco' segreto in un programma che esegue istruzioni illecite
Attacchi basati sull'alterazione della pila (buffer overflow)→ riempiendo una pila di dati fino a superarne i limiti,la si invalida. Così facendo, un utente non autorizzato può ottenere un accesso, o uno autorizzato ottenere privilegi che non gli spettano.

MINACCE AI SISTEMI

Worm→ processo che sfrutta gli effetti della proliferazione per minare le prestazioni del sistema; esso genera continuamente copie di se stesso logorando le risorse del sistema, talvolta fino a renderlo inutilizzabile da tutti gli altri processi.
Virus → frammento di codice, che si inserisce in un programma legittimo, progettato per distruggere o modificare file, causare crolli di sistema o malfunzionamenti dei programmi.
Attacchi per rifiuto del servizio (denial of service) → prevede il blocco dell'utilizzo legittimo di un sistema o di un

servizio.

MIGLIORARE LA SICUREZZA DEI SISTEMI

Bisogna controllare
- o Parole d'ordine brevi o facile da ricordare
- o Programmi privilegiati non autorizzati
- o Programmi non autorizzati nelle directory di sistema
- o Processi dall'esecuzione inaspettatamente lunga
- o Improprie protezioni delle directory sia degli utenti sia di sistema
- o Improprie protezioni dei file di dati del sistema

- Elementi pericolosi nel percorso di ricerca dei programmi
- Modifiche ai programmi di sistema individuate con somme di controllo
- Demoni di rete inattesi o nascosti

RILEVAMENTO DELLE INTRUSIONI → *attività volta a scoprire tentativi di intrusione o intrusioni già avvenute nei sistemi di calcolo e ad attivare azioni appropriate in risposta alle intrusioni stesse.*

- Verifica e registrazione → si elaborano dei tracciati di verifica, in cui gli eventi rilevanti per la sicurezza si memorizzano in file di sistema che poi si confronta con tracce d'attacchi (nel rilevamento basato su tracce), oppure si analizzano per scoprire comportamenti anomali (nel rilevamento di anomalie)
- Rilevamento basato sulle tracce → si esaminano i sati in ingresso al sistema o il traffico della rete alla ricerca di particolari schemi o sequenze di azioni, chiamate tracce, che si ritengono indizi di attacchi
- Rilevamento di anomalie → tecniche che cercano d'identificare comportamenti anomali in un sistema

CRITTOGRAFIA

Da un punto di vista astratto, la crittografia si usa per imporre i potenziali mittenti e destinatari di un messaggio. Si fonda su codici segreti, chiamati chiavi, che si distribuiscono selettivamente ai calcolatori di una rete e si usano per gestire i messaggi. La crittografia permette al destinatario di verificare che il messaggio sia stato creato da un calcolatore che possiede una certa chiave. Analogamente, un mittente può codificare un messaggio in modo che solo un calcolatore in possesso di una determinata chiave lo possa decifrare.