Pipelining = tecnologia per incrementare il throughput, ovvero la quantità di istruzioni eseguite in un certo tempo, parallelizzando i flussi di elaborazioni di più istruzioni. Tecnica che permette di partire da un processo e dividerlo in sotto operazioni per sfruttare meglio la CPU
Superscalarità = strumento per risolvere situazioni in cui nel pipelining ci siano delle fasi che necessitano di un tempo maggiore rispetto ad altre.
Multicore = aumentare le componenti che si occupano di processare i dati (all'interno dei processori) in modo da eseguire più operazioni nello stesso momento

System call = chiamate che fanno i livelli superiori per accedere alla macchina astratta fornita dal sistema operativo. Sono operazioni presenti nell'API (Application Programming Interface)

CAPITOLO 2
La **memory** è la memoria centrale, chiamata anche RAM (Random Access Memory)
Il **System bus** collega tutte le unità (fili microscopici in cui passano i comandi e i dati). Quando si parla di architettura a 32 o 64 bit, ci si riferisce al numero di fili che corrispondono al numero di bit che possono transitare contemporaneamente sul bus
Il **controller** è un piccolo circuito che controlla l'hardware puro, contiene alcuni registri che vengono usati dal processore per inviare istruzioni a quella specifica periferica
Il **driver** è la parte del sistema operativo che dialoga con i controller
In una CPU possiamo distinguere due parti importanti:

1. Arithmetic-Logical Unit: unità che effettua i calcoli di aritmetica e operazioni logiche
2. Control Unit: cervello del processore, capisce qual è la prossima istruzione che deve essere eseguita, cerca di comprendere cosa succede se si esegue l'istruzione e impartisce gli ordini per eseguire le istruzioni. Contiene dei registri:
   - Program Counter (PC), registro nel quale è registrato il numero della cella di memoria che contiene l'istruzione che deve essere eseguita
   - Instruction Register (IR), contiene l'istruzione da eseguire

   L'attività della control unit si compone in fasi:
   - Fetch = viene recuperata l'istruzione da eseguire
   - Decode = comprensione dell'istruzione, cosa comporta e cosa bisogna fare
   - Execute = esecuzione

La RAM è suddivisa in celle che hanno una numerazione crescente, da 0 a n, dove n è l'indirizzo massimo per la nostra memoria. Ogni cella è un byte, quindi la memoria centrale è una sequenza di byte che sono indirizzabili dal processore attraverso opportuni registri. Registri:

- MAR = memory address register (ha le informazioni su dove trascrivere)
- MDR = memory data register (ha le informazioni da trascrivere)
- Command register (manda il comando, es. write)

All'interno di una macchina si trovano diversi tipi di memorie:

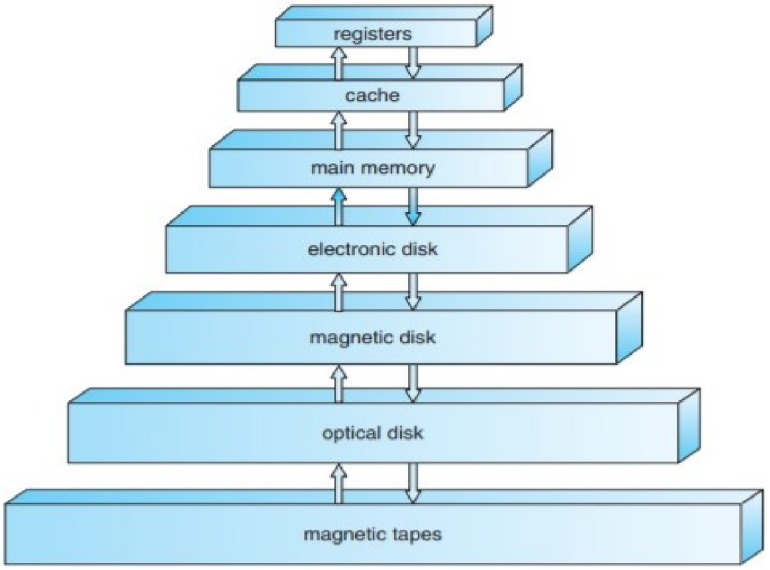
- Registers: registri all'interno del processore (sono limitati e nell'ordine dei KB)
- **Cache**: memoria intermedia tra i registri del processore e la memoria centrale (nell'ordine dei MB), molto veloce. Memorizza le informazioni che vengono usate frequentemente dal processore così che possano essere facilmente ritrovabili.

Principio di località
Le informazioni e i dati che un processo usa in un certo momento sono molto simili a quelli che userà nell'immediato futuro o che ha usato nell'immediato passato. L'insieme dei dati e delle informazioni che un processo usa in un certo momento, cambiano con il passare del tempo. Questo principio permette di utilizzare le memorie cache in maniera efficace e si può suddividere in:

- Principio di località temporale: insieme di istruzioni e dati in un uso per un certo processo che cambia lentamente nel tempo
- Principio di località spaziale: istruzioni e dati in uso in questo momento, sono in posizione contigua a quelli che userò nell'immediato futuro o che ho usato nell'immediato passato

Il principio di località viene utilizzato nelle memorie cache e nella paginazione
La cache principale è quella esterna al processore, esistono anche cache interne alla CPU

Il sistema operativo deve conoscere le cache e decidere cosa portarci dentro e quando. Il livello di cache viene attribuito in base a quante cache sono presenti. Le cache sono di dimensioni molto ridotte rispetto alla memoria centrale e si riempiono facilmente, quando sono piene, per far spazio a nuovi dati, bisogna eliminarne dei vecchi
Consistenza dei dati: può capitare che, quando sono presenti più processori in una macchina, un dato venga memorizzato su più memorie e quando viene richiamato, si prende un vecchio dato di una memoria che non è stato aggiornato
La memoria centrale è disponibile nell'ordine del MB/GB, le altre fonti di memorie sono tutte secondarie



Hardware Protection
Dual-Mode Operation: un bit all'interno della CPU che distingue se un'istruzione è di un processo (user mode) o di un sistema operativo (monitor mode)
Le periferiche I/O possono essere usate soltanto dal sistema operativo (istruzioni privilegiate), l'utente non ha accesso diretto ma deve utilizzare le system call
Memory Protection: garantisce che un job non interferisca nelle zone di memoria di altri processi
CPU Protection

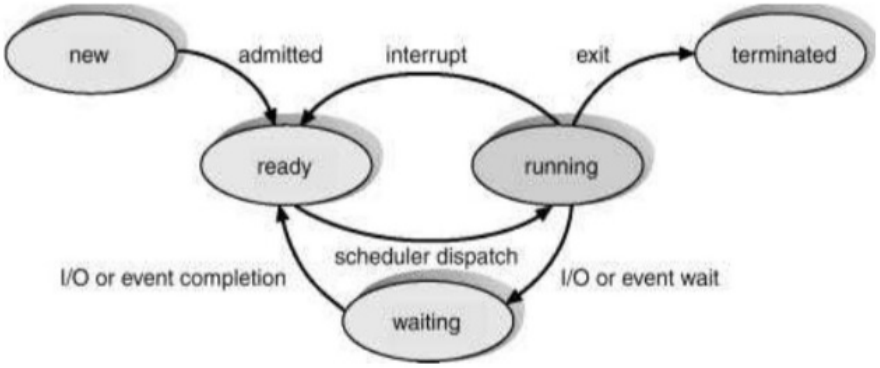
CAPITOLO 3
Un pezzo di codice viene chiamato programma (entità passiva), quando viene lanciato diventa un processo (entità attiva) un processo è un programma in esecuzione.
Un processo necessita di:

- Un codice = le istruzioni che deve eseguire
- Dati = dati globali che si aggiornano durante l'esecuzione
- Stack = per gestire la dinamicità del processo

Program counter: punto del codice in cui sono arrivato durante l'esecuzione del processo
Ci sono 5 stati per un processo:

1. New = creazione di un nuovo processo
2. Running = processo in esecuzione
3. Waiting = processo che sta aspettando un evento per partire
4. Ready = processo pronto per essere eseguito ma il processore è impegnato da un altro processo. Quando il processore si libera, il sistema operativo deve fare una scelta sui processi in stato ready
5. Terminated = processo che ha terminato la sua esecuzione

\



Process Control Block (PCB): è una struttura dati (interna al SO) che traccia delle informazioni principali associate ad un processo (caratteristiche e risorse utilizzate da esso). Il PCB è molto importante nel **context switch** (cambio di contesto), ovvero nella fase in cui il sistema operativo fa passare sul processore un nuovo processo rimuovendo quello precedente, perché tiene traccia dello stato del programma (come una fotografia)

CAPITOLO 4 THREADS

Il processo contiene un insieme di risorse e rappresenta un flusso di controllo (attività in esecuzione). Il passaggio ai threads si può pensare come la separazione delle risorse dal flusso di esecuzione. Attraverso i threads si possono avere più flussi in esecuzione.

I **threads** (flussi di controllo) sono più leggeri rispetto ai processi e effettuare un context switch, a livello di processi, richiede abbastanza tempo. All'interno di un processo c'è la possibilità di avere diversi flussi di controllo, ovvero diversi thread che condividono le stesse risorse.

Il vantaggio è l'efficienza: il fatto di poter strutturare un'applicazione in un insieme di thread cooperanti, permette di avere concorrenza e parallelismo (più attività che avanzano in modo parallelo)

Lo svantaggio è la sincronizzazione: i thread condividono le stesse risorse e ciò può portare a interferenze tra di loro ma anche a complicare la comprensione di quello che succederà con i dati (se entrambi i thread modificano i dati, può portare a del non determinismo, situazione finale in cui i dati sono diversi nei due thread).

Come implementare i thread in java

- Estendere la classe predefinita thread: "class abc extends Thread"

 Questo comporta l'obbligo di definire un metodo run() all'interno della classe

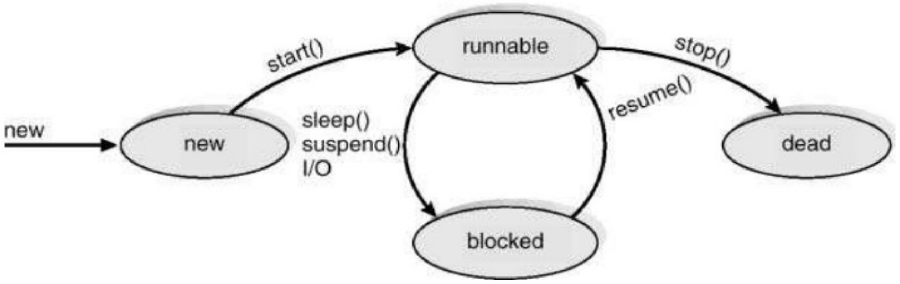
- Utilizzo dell'interfaccia runnable: "public interface Runnable{ public abstract void run() }

 un interfaccia è una classe in cui vengono specificati solo i metodi e i parametric che entrano in gioco; chi implementa l'interfaccia Runnable dovrà definire il metodo run()

```
public class Worker2 implements Runnable {
public void run(){
System.out.println("I am a Worker Thread");
}
}
```

```
public class Second {
public static void main(String[] args){
Runnable runner = new Worker2();
Thread thrd = new Thread(runner);
thrd.start();
System.out.println("I am the main thread"); }}
```

Anche i thread hanno uno stato come i processi



CAPITOLO 5 CPU SCHEDULING

Parte del sistema operativo che si occupa della gestione del processore (scelta del processo che deve essere eseguito). I processi attraversano periodi di uso del processore (CPU bound) e di uso delle periferiche di I/O (I/O bound). Tre tipi di scheduling:

1. Scheduling a lungo termine = seleziona i processi che devono essere portati dalla memoria secondaria alla memoria centrale
2. Scheduling a breve termine = sceglie quale processo far eseguire sulla CPU
3. Scheduling a medio termine (swapper) = si occupa di trasferire momentaneamente dei processi dalla memoria centrale alla secondaria e viceversa

L'obbiettivo del CPU scheduling è scegliere fra i processi in memoria, che sono ready to execute, quali portare in stato running

Uno scheduling si dice:

- Nonpreemptive: una volta selezionato un processo per l'esecuzione, questo processo rimane in esecuzione fintanto che ha bisogno del processore
- Preemptive: il processo potrebbe perdere il processore in un momento in cui egli vuole rimanere sulla CPU perché ha ancora dei compiti da eseguire

Scheduling Criteria

Criteri che devono essere ottimizzati:

- CPU utilization = quanto viene tenuto attivo il processore
- Response time = periodo di tempo tra una richiesta inviata dall'utente e processore che inizia a prendere in carica la richiesta
- Waiting time = periodo di tempo che un processo aspetta nella ready queue
- Turnaround time = tempo complessivo per eseguire un certo processo
- Throughput = numero di processi che completano la loro esecuzione per unità di tempo

 Minimizzare i time e massimizzare la CPU utilization e il throughput

FCFS (First-Come, First-Served) Scheduling: il primo processo che arriva nell'insieme ready, sarà il primo a essere eseguito

SJF (Shortest-Job First) Scheduling: il processo più breve viene eseguito per primo. Il waiting time medio è il migliore possibile sia in ambito nonpreemptive che preemptive. Il problema è che non si conosce il burst time, è impossibile determinare se un programma terminerà non viene utilizzato

Altro problema è la **starvation**: situazione in cui un processo non riesce ad andare avanti nell'esecuzione (diverso da deadlock, che è quando tutti i processi sono bloccati)

Principio di località: dice se il comportamento di un processo cambia molto lentamente nel tempo. Le istruzioni e i dati che il processo utilizza in un certo momento (working set) saranno molto simili a quelli utilizzati nel periodo precedente o successivo. Viene utilizzato nello SJF, in quanto lo SJF si basa su un comportamento futuro dei nostri processi, quindi si fa riferimento al periodo di CPU burst (quello che succede al processo in futuro)

Priority Scheduling: presenta il problema della *starvation* (se il processo ha priorità bassa), la soluzione si chiama **aging** e consiste nell'aumentare la priorità di un processo quando resta per troppo tempo in cosa

Round Robin (RR): a ogni processo viene assegnato un quanto di tempo e, quando scade, il processo viene tolto dal processore e messo sulla coda dei processi ready (e così via). Se il quanto di tempo è molto grande allora degenera in una FIFO, se invece è molto basso, si avrà un numero elevato di content switch e si rischia che la somma degli overhead causati dai context switch diventino molto significativi

Multilevel Queue: i sistemi operativi moderni cercano di combinare le varie politiche precedenti per massimizzare i benefici e ridurre gli inconvenienti si usa una coda su più livelli

Processor Affinity (per macchine con più processori): consente il binding e l'unbinding di un processo o thread ad una CPU, in modo che venga eseguito solo da una CPU

{..} = non determinismo

Per un sistema time sharing è necessario usare un algoritmo preemptive (non SJF ma RR, che non produce starvation) in modo da garantire che qualsiasi processo di qualsiasi utente possa prima o poi usare la CPU

• Proprietà di distributività sulla somma: **[(a mod n) + (b mod n)] mod n = (a+b) mod n**;
• Proprietà di distributività sulla sottrazione: **[(a mod n) - (b mod n)] mod n = (a-b) mod n**;
• Proprietà di distributività sulla moltiplicazione: **[(a mod n) * (b mod n)] mod n = (a* b) mod n**;
• Proprietà di distributività sull'esponente: **(a mod n)d mod n = ad mod n**;
Teorema di Fermat: Dato un numero primo p e a con a < p, allora possiamo dire che
a^(p-1) mod p = 1. Una conseguenza di questo teorema è che se si prende
(a^(p-1) x a) mod p = (a^(p-1) mod p x a mod p) mod p = 1 x a = a
Teorema di Eulero: se p,q sono numeri primi, F(p x q) = (p - 1) x (q - 1)

CAPITOLO 6
Il semaforo è una variabile, che può essere incrementata e decrementata, e che gestisce ..
Codice operazioni semaforo

P(S): **while** S <= 0 **do** no op
S--;
V(S): S++;

Semaforo binario = può assumere solo valori 0 e 1
Variabile booleana = può assumere solo 'vero' o 'falso'

Il semaforo empty indica quante sono le caselle libere sul buffer, mentre il semaforo
full indica quante sono le caselle piene.
il semaforo mutex() serve per assicurarsi che le manipolazioni sul buffer avvengano in maniera mutual exclusion
Per entrare nella mutua esclusione bisogna verificare che non sia pieno perché si potrebbe finire in deadlock

Critical section = zona di codice di un processo che prevede l'utilizzo di una variabile/risorsa condivisa
Mutua esclusione = quando un processo opera sulla variabile condivisa, nessun altro può fare altrettanto
Progress = condizione di progresso dice che se nessun processo è in esecuzione nella sezione critica e ci sono dei processi che vorrebbero entrarci, la sezione del processo che dovrà entrarci non può essere rimandata all'infinito

CAPITOLO 7 Memory Management
Swapping
Le operazioni che le variabili di un programma possono fare sul processo sono 'load' e 'write'
L'associazione di indirizzi di istruzioni e dati agli indirizzi di memoria può avvenire in tre fasi:

- Compile time = se in fase di compilazione si conosce dove risiederà il processo in memoria, allora è possibile generare il codice assoluto (se la posizione di partenza cambia allora sarà necessario ricompilare il codice)
- Load time = se non si sa dove risiederà il processo in memoria allora il compilatore deve generare un codice trasferibile; l'associazione finale viene ritardata fino al momento del caricamento, se l'indirizzo di partenza cambia allora è sufficiente ricaricare il codice utente
- Execute time = quando il processo viene caricato sulla memoria centrale, i riferimenti rimangono quelli iniziali. Quando viene eseguito il load(k), la CPU chiede di accedere alla cella k ed è necessaria un operazione r, per accedere alla cella m di RAM, dove m = k+r

Indirizzo fisico = indirizzo di una cella di memoria sul quale si vuole operare (esiste in memoria)

Indirizzo logico = indirizzo prodotto dalla CPU, è virtuale e serve da riferimento per accedere alla posizione di memoria fisica
L'MMU (Memory Management Unit) calcola l'indirizzo fisico per l'indirizzo logico corrispondente
Swapping = spostamento di un processo da memoria centrale a memoria secondaria

Contiguos Memory Allocation
I processi vengono messi su memoria centrale in dei blocchi contigui di memoria (uno per processo)
Dopo che la CPU produce un indirizzo logico viene cambiato in fisico attraverso il re location register e viene effettuato un controllo di protezione.
Dynamic Storage-Allocation Problem, come salvare i buchi di memoria:

- First-fit = processo allocato nel primo hole abbastanza grande da contenerlo
- Best-fit = si sceglie in base alla taglia del processo dato in modo da cercare l'hole che meglio approssima il processo
- Worst-fit = allocare il processo nel buco più grande possibile

Il problema principale dell'allocazione contigua è la **frammentazione**: dopo un po' di utilizzo del sistema (terminazioni di alcuni processi) ci si ritrova nella situazione in cui abbiamo tanti piccoli buchi all'interno della memoria centrale che non sono utilizzati la soluzione è **compattare**: spostare i processi in questione in modo da lasciare un unico che abbia come taglia la somma degli altri buchi di partenza (durante questo periodo la macchina è inutilizzabile). In questo caso si chiama **frammentazione esterna**.

Paging
Nell'allocazione contigua, si assegna un area contigua di memoria ai processi, dove non è possibile spezzettarli. Con la paginazione è possibile spezzettare i processi, si suddivide la memoria centrale in dei blocchi chiamati frame e un processo in blocchi chiamati pagine.
Vantaggi: si evita la frammentazione esterna
Svantaggi: la page table occupa spazio
La taglia di frame e pagine è uguale, le pagine possono essere posizionate sui frame che non sono necessariamente contigui ma bisogna tenere conto degli indirizzi logici e fisici, tramite 3 punti:

1. Numero della pagina
2. Numero del frame collegato alla pagina
3. Offset che indica lo scostamento

Ciò porta alla creazione di una page table che indica come i blocchi di un processo siano stati allocati in memoria
Come effettuare l'impaginazione:

1. Determinare lo spazio degli indirizzi logici di un processo, in modo non contiguo
2. Dividere la memoria fisica in blocchi di dimensioni fisse chiamati frame
3. Dividere la memoria logica in blocchi della stessa dimensione, chiamati pagine
4. Per eseguire un processo di dimensioni n pagine, è necessario trovare n frame liberi e caricare il processo
5. Impostare una tabelle delle pagine per tradurre gli indirizzi logici in indirizzi fisici (page table)
6. Frammentazione interna

L'indirizzo logico si divide in: numero di pagina (prima cifra, usato come indice nella page table che contiene il numero del frame contenente la pagina relativa) e offset (due cifre finali, taglia frame = taglia pagina)
Mentre nell'indirizzo fisico si ottiene: il numero di frame (prima cifra ottenuta consultando la page table che mappa la cella da logica a fisica) e offset (ultime due cifre ottenute trasportandole dall'indirizzo logico)
Memoria associativa (TLB)
È una memoria tampone che l'MMU (Unità di gestione della memoria interna) utilizza per velocizzare la traduzione degli indirizzi virtuali. Il TLB possiede un numero fisso di elementi della tabella delle pagine, la quale viene usata per mappare gli indirizzi virtuali in indirizzi fisici. La TLB serve a caricare dei pezzi della page table e la memoria complessivamente coperta è data da un numero di entry, moltiplicato per la taglia di pagina (kb, mb,..)
EAT = hit ratio of TLB * (access time of TLB + access time of main memory) + (1-hit ratio of TLB) * (access time of TLB + 2 access time of main memory)

La **memoria virtuale** è lo spazio visto da un processo che può essere più grande della memoria fisica reale. L'idea è quella di poter eseguire un processo avendo caricato sulla memoria centrale solo un sottoinsieme delle sue pagine, quindi non serve caricarlo fisicamente tutto sulla memoria aumento del livello di multiprogrammazione e un processo può essere eseguito senza dover attendere che l'intero processo sia caricato su RAM. Svantaggi: richiede molte risorse e se il numero delle pagine è molto grande, il meccanismo diventa complesso e rallenta l'accesso alla memoria
Page fault = andare a cercare sulla memoria secondaria una parte di un processo che non è stata caricata sulla RAM durante la produzione di un indirizzo logico. Viene prodotto da un processo che cerca di accedere ad una pagina che in quel momento non è presente su memoria centrale. Se non ci sono frame liberi in cui inserire la pagina recuperata, bisogna trovare una pagina vittima (da eliminare per fare spazio all'altra). Ci sono due algoritmi importanti per il page fault:
Page replacement = usa il dirty bit per ridurre l'overhead del trasferimento delle pagine (ha a che fare con la scelta della pagina vittima). Così si completa la separazione tra memoria logica e fisica. L'algoritmo di page replacement ha il compito di fare in modo che la percentuale di page fault sia più bassa possibile
Frame allocation = determina quanti frame associare ai singoli processi

- Equal allocation = a tutti i processi si da lo stesso numero di frame, è eccessivo perché alcuni processi hanno esigenze maggiori rispetto ad altri
- Proportional allocation = frame allocati in modo proporzionale alla taglia del processo stesso
- Allocazione per priorità = assegnare una quantità di frame ai vari processi che è proporzionale alla priorità dei processi stessi

L'allocazione può essere:

- Globale: se permette ad un processo di selezionare un frame di sostituzione a partire dall'insieme di tutti i frame (vantaggioso per favorire processi a priorità alta)
- Locale: se ogni processo effettua la scelta solo nel proprio insieme di frame allocati

Algoritmi di page replacement:
First-In-First-Out (FIFO)
Optimal Page Replacement Algorithm: algoritmo ottimale ma impossibile da implementare, la pagina vittima è quella che verrà utilizzata più in là nel tempo
Last Recently Used (LRU): la scelta della pagina vittima cade sulla pagina usata meno recentemente (si guarda al passato)
Reference Bit: associare ad ogni pagina un bit che indica se la pagina è stata utilizzata nell'ultimo intervallo di tempo (in quel caso viene settato). L'idea è quella di selezionare una pagina che abbia il reference bit a 0
Second Chance: unisce reference bit e FIFO, l'idea è di scegliere la pagina vittima guardando il reference bit della pagina stessa, se è a 0 allora diventa vittima, se è a 1 gli viene data una seconda possibilità e il bit viene settato a 0. Quindi la prima pagina nella lista con il reference bit a 0 diventa la pagina vittima

Thrashing
È un rischio della memoria virtuale. Quando il livello di multiprogrammazione diventa troppo elevato rispetto alle esigenze dei singoli processi e ciò potrebbe causare un elevato numero di page fault. Quando ci sono troppi page fault, il processore potrebbe essere poco utilizzato. Per ovviare a questo problema è necessario ridurre il livello di multiprogrammazione (a livello software) oppure ridurre il numero di page (hardware)

CAPITOLO 9 FILE SYSTEM
È un metodo del sistema operativo usato per immagazzinare, catalogare e recuperare file. La sua struttura si divide in logica e fisica
Nella struttura logica il SO deve definire le operazioni per manipolare il file system, usando come unità di trasferimento le informazioni tra un file ed un processo (record logico). Nel caso standard viene preso come record logico il Byte e i metodi di accesso sono di due tipi:

- Sequenziale: il record logico successivo da utilizzare è il record logico precedente o successivo (ci si può spostare sequenzialmente nel file

- Casuale/diretto: ci si muove liberamente nel record logico

Nella struttura fisica i record logici sono allocati nella memoria di allocazione secondaria (disco rigido). Lo spazio disponibile viene diviso in blocchi fisici. Dato un certo file costituito da un insieme di record logici, bisogna capire dove sono posizionati e come si riesce a tener traccia degli stessi. Tre metodi di allocazione:

- Contiguo: ogni file occupa blocchi contigui sul disco. Si potrebbero avere problemi di frammentazione esterna perché si ha della memoria inutilizzabile esterna rispetto a quella assegnata per i file. Si utilizza la deframmentazione per ricavare da tanti blocchi piccoli liberi, un blocco più grande
- Concatenato: trasmissione dei file utilizzando un puntatore per riconoscere il blocco successivo del file system (evita la frammentazione esterna ma non considera il vincolo di contiguità). Non è vantaggiosa perché se ti serve l'ultimo elemento del blocco devi scorrerlo tutto e perché se si perde un puntatore si perdono anche tutti i blocchi che seguono (per evitare questo ultimo problema si possono mettere puntatori nella direzione opposta ma ciò raddoppierebbe la dimensione per i puntatori). Si può usare la File Allocation Table che contiene un elemento per ogni blocco e indica se il blocco è libero oppure no
  Un blocco grande riduce il numero di operazioni IO richieste per ritrovare un certo record, ma al costo di maggiore frammentazione interna; riducendo la dimensione del blocco avviene il contrario

- Indicizzato: per ogni file system si usa un blocco separato per scrivere la sequenza in cui il file è stato memorizzato. All'interno si trovano i puntatori dei blocchi del file
  Si evita la frammentazione esterna
  Bisogna utilizzare un blocco indice per ogni file e quindi viene sprecato tanto spazio
  L'accesso ai file è più lento rispetto alle altre 2 allocazioni, bisogna sempre leggere prima il blocco indice e poi il file (due letture). Se i file sono molto piccoli, c'è un grande spreco di spazio su disco perché il blocco indice rimane quasi completamente inutilizzato

Si indica con TBF la taglia del blocco fisico e con TRL la taglia del record logico il numero di record logici dentro un blocco fisico è dato dalla divisione tra TBF e TRL

Crittografia a chiave simmetrica/privata

La chiave di cifratura è la stessa di decrittazione

- Deve rimanere segreta ma deve essere nota ad entrambi i partecipanti (sender, recevier)
- La chiave deve essere recapitata in modo sicuro ad entrambi i partecipanti
- Il numero di chiavi necessarie aumenta in base al quadrato del numero delle persone che vogliono scambiarsi i messaggi: se ci sono n persone è c'è la possibilità di effettuare scambi segreti tra coppie di persone, allora servono circa n*(n-1)/2
- Gli attaccanti che intercettano il messaggio non avranno la chiave e dovranno usare una decrittazione che richiede elevate capacità di calcolo

Crittografia a chiave asimmetrica/pubblica

Con n persone servono n coppie di chiavi

Ogni persona coinvolta genera una chiave pubblica e una privata, chiunque abbia la chiave pubblica può inviare dati crittografati al proprietario della chiave privata, controllarne la firma digitale o autenticarlo

- Si evitano necessità di uno scambio in modo sicuro dell'unica chiave utile alla cifratura/decifratura (come nella simmetrica)
- Viene utilizzata nelle firme digitali

Algoritmo RSA

Un messaggio viene cifrato, con la chiave pubblica del destinatario (che chiameremo X), dal mittente (che chiameremo Y). Quindi Y cercherà sull'elenco la chiave pubblica di X per cifrare il messaggio, dopodiché solo X con la sua chiave privata potrà decifrare il messaggio e neanche Y sarà più in grado di leggerlo. È necessario che solo X possegga la sua chiave privata, altrimenti non avrebbe senso.

Come funziona l'algoritmo?

È basato sull'elevata complessità computazionale della fattorizzazione in numeri primi:

1. Si scelgono due numeri primi p e q abbastanza grandi
2. Si calcola il loro prodotto n = p x q, chiamato modulo, e la funzione toziente f(n) = (p-1) x (q-1)
3. La fattorizzazione di n è segreta e solo chi sceglie i due numeri p e q la conosce
4. Si sceglie un numero e tale che non esiste un numero diverso da 1 che sia divisibile per e e per f(n); quindi esiste un inverso moltiplicativo d tale che (e x d)mod f(n) = 1
5. La chiave pubblica è costituita dalla coppia [n,e] mentre la chiave privata dalla coppia [n,d] e l'informazione segreta è d

//Semaforo e pseudocodice delle operazioni sui semafori

Formule da sapere

Quanti sono i blocchi nell'HD: taglia HD / taglia singolo blocco = numero di puntatori (in bit) quindi diviso 8
Numero di blocchi in cui il file è diviso: dimensione file / taglia singolo blocco
EAT = hit ratio of TLB * (access time of TLB + access time of main memory) + (1-hit ratio of TLB) * (access time of TLB + access time of main memory * number of pages + access time of main memory)
EAT = Rt * (Tt + Tm) + (1-Rt) * (Tt + Tm * n + Tm)
Percorso assoluto è il percorso completo (tutta la path)
Percorso relativo è solo una directory con la sua struttura
Symbolic link = tipo di file che è il collegamento ad un altro file o ad una directory
Hard link = copiare un file, quindi è presente il file completo e non un collegamento

1 Mbyte = 2^20 byte
1 Kbyte = 2^10 byte
1 byte = 8 bit

Spazio di indirizzamento fisico = grandezza memoria centrale
Grandezza memoria centrale = numero di frame * taglia pagina
Taglia frame = taglia pagina
Se grandezza memoria centrale = 2^x allora il numero di bits dell'indirizzo fisico è x
Se taglia pagina = 2^x allora il numero di bits dell'offset è x
Se numero di frame = 2^x allora il numero di bits del frame è x
Spazio di indirizzamento logico = 2^x allora il numero di bits dell'indirizzo logico è x
Spazio di indirizzamento logico = taglia processo
Spazio di indirizzamento logico = numero pagine * taglia pagina
Numero di pagine del processo = taglia processo / taglia pagina
Se taglia processo = 2^x allora il numero di bits dello spazio di indirizzamento logico è x
Taglia page table = numero di entry page table * taglia entry page table
Numero di entry page table = numero di pagine del processo
Taglia entry page table = numero di bits del frame + numero bits per altri campi
Se taglia page table > taglia frame allora bisogna usare la paginazione
Se spazio di indirizzamento logico > grandezza memoria centrale allora bisogna usare la memoria virtuale

Grandezza di un frame = bit in cui è scritto l'indirizzo fisico del sistema – bit in cui è suddivisa la ram
Entry page table più grande = indirizzo logico / frame (offset)
Frame page table più grande = numero entry page table * byte per scrivere il numero di un frame (esponente del numero di frame / 8) / frame(offset)

Quando la dimensione dell'indirizzo logico è superiore alla RAM bisogna implementare la memoria virtuale

Se la cpu è bassa e il disco alto allora bisogna ridurre il livello di multiprogrammazione (Thrashing)

Se la cpu è alta e il disco è basso allora è una situazione ottimale
Se la cpu è bassa e il disco è basso allora bisogna aumentare il livello di multiprogrammazione

Domande più comuni (non finito)

Il **bit di validità** permette di sapere se la pagina di cui abbiamo il riferimento virtuale risiede nella memoria fisica o no. In caso affermativo, la page table contiene l' indirizzo fisico.

Principio di località + esempio di applicazione

Durante l'esecuzione di una data istruzione presente in memoria, con molta probabilità le successive istruzioni saranno ubicate nelle vicinanze di quella in corso. Nell'arco di esecuzione di un programma si tende a fare riferimenti continui alle stesse istruzioni
A livello hardware: memoria cache(come TLB) e della memoria virtuale
A livello software: SJF o LRU (page replacement)

Page Table

Una tabella di pagine è una struttura che indica come i blocchi di un processo siano
stati allocati in memoria centrale, associando blocchi di RAM alle pagine del processo.
Inoltre è importante per la traduzione da indirizzi virtuali a indirizzi fisici, necessaria per accedere ai dati in memoria.

Inverted Page Table

invece di avere una page table per ogni processo, si ha un'unica page table per tutta la memoria centrale.
Da informazioni relative ai frame in cui è divisa la RAM e non alle pagine dei processi.

Paging

È la tecnica attraverso la quale il sistema operativo del computer, per mezzo dei rispettivi algoritmi di paging, suddivide la memoria in parti di dimensioni minori, e la alloca al programma da eseguire usando pagine come blocco minimo di lavoro. Si suddivide la memoria centrale in dei blocchi chiamati frame e un processo in blocchi chiamati pagine.
Vantaggi: Poiché i programmi raramente utilizzano contemporaneamente tutte le parti del proprio codice e dei propri dati, è conveniente implementare il meccanismo della memoria virtuale. Si evita la frammentazione esterna
Svantaggi: perdita di prestazioni; codice complesso da realizzare, maggior tempo di traduzione degli indirizzi da logici a fisici

Page table su più livelli?

PCB

è una struttura dati (interna al SO) che traccia delle informazioni principali associate ad un processo (caratteristiche e risorse utilizzate da esso). Il PCB è molto importante nel context switch (cambio di contesto), ovvero nella fase in cui il sistema operativo fa passare sul processore un nuovo processo, rimuovendo quello precedente perché tiene traccia dello stato del programma (come una fotografia)
Contiene: program counter, livello di priorità, informazioni per lo scheduling del processo, ecc..

Indicare le azioni che devono essere svolte per costruire l'indirizzo fisico finale.

L'indirizzo logico si divide in: numero di pagina (prima cifra) e offset (due cifre finali)
Mentre nell'indirizzo fisico si ottiene: il numero di frame (prima cifra ottenuta consultando la page table che mappa la cella da logica a fisica) e offset (ultime due cifre ottenute trasportandole dall'indirizzo logico)

Memoria virtuale

È un'architettura capace di simulare uno spazio maggiore rispetto a quello fisicamente disponibile
L'idea è quella di poter eseguire un processo avendo caricato sulla memoria centrale solo un sottoinsieme delle sue pagine, quindi non serve caricarlo fisicamente tutto sulla memoria □ aumento del livello di multiprogrammazione e un processo può essere eseguito senza dover attendere che l'intero processo sia caricato su RAM. Svantaggi: richiede molte risorse e se il numero delle pagine è molto grande, il meccanismo diventa complesso e rallenta l'accesso alla memoria

Multilevel queue

i sistemi operativi moderni cercano di combinare le varie politiche precedenti per massimizzare i benefici e ridurre gli inconvenienti □ si usa una coda su più livelli

Process affinity

(per macchine con più processori): consente il binding e l'unbinding di un processo o thread ad una CPU, in modo che venga eseguito solo da una CPU

Pipelining

tecnologia per incrementare il throughput, ovvero la quantità di istruzioni eseguite in un certo tempo, parallelizzando i flussi di elaborazione di più istruzioni. Tecnica che permette di partire da un processo e dividerlo in sotto operazioni per sfruttare meglio la CPU

Superscalarità

strumento per risolvere situazioni in cui nel pipelining ci siano delle fasi che necessitano di un tempo maggiore rispetto ad altre.

Multicore

aumentare le componenti che si occupano di processare i dati (all'interno dei processori) in modo da eseguire più operazioni nello stesso momento

Controller

è un piccolo circuito che controlla l'hardware puro, contiene alcuni registri che vengono usati dal processore per inviare istruzioni a quella specifica periferica. Fa accedere al bus una o più unità periferiche

Driver

è la parte del sistema operativo che dialoga con i controller

Thrashing

È un rischio della memoria virtuale. Quando il livello di multiprogrammazione diventa troppo elevato rispetto alle esigenze dei singoli processi e ciò potrebbe causare un elevato numero di page fault. Quando ci sono troppi page fault, il processore potrebbe essere poco utilizzato. Per ovviare a questo problema è necessario ridurre il livello di multiprogrammazione (a livello software) oppure aumentare la memoria RAM (hardware)

Frame allocation

Determinare quanti frame vanno associati ai singoli processi, tramite 3 metodi:
Equal allocation: a tutti i processi si da lo stesso numero di frame
Proportional allocation = i frame vengono allocati in modo proporzionale alla taglia del processo stesso
Priority allocation = viene assegnata una quantità di frame ai vari processi, proporzionale alla priorità dei processi stessi
L'allocazione può essere *globale*, se permette ad un processo di selezionare un frame di sostituzione a partire dall'insieme di tutti i frame, oppure *locale*, se ogni processo effettua la scelta solo nel proprio insieme di frame allocati

Indirizzo fisico e logico

l'indirizzo logico è generato dalla CPU nella prospettiva di un programma mentre l'indirizzo fisico è una posizione che esiste nell'unità di memoria.
L'indirizzo logico si divide in: numero di pagina (prima cifra, usato come indice nella page table che contiene il numero del frame contenente la pagina relativa) e offset (due cifre finali, taglia frame = taglia pagina)
Nell'indirizzo fisico si ottiene: il numero di frame (prima cifra ottenuta consultando la page table che mappa la cella da logica a fisica) e offset (ultime due cifre ottenute trasportandole dall'indirizzo logico)

Frammentazione interna e esterna

Quando la memoria viene allocata in blocchi di dimensione fissata, con l'espressione frammentazione interna si intende la differenza fra la memoria assegnata ad un processo e quella effettivamente richiesta da quest'ultimo. Quando si alloca la memoria in blocchi di dimensione variabile e si caricano e si rimuovono da quest'ultima dei processi, lo spazio libero si frammenta in piccole parti; si parla di frammentazione esterna quando lo spazio libero complessivo nella memoria è sufficiente per soddisfare una richiesta, ma non è contiguo

La starvation di un processo è quando il processo non riesce a proseguire nella sua esecuzione.
Nel caso Round Robin non ci può essere starvation essendo ogni quanto di tempo avviene uno switch tra i processi.

Nel caso SJF è possibile essendo che potrebbe arrivare un processo troppo lungo che non venga mai eseguito, la soluzione sarebbe usare l'aging, ovvero aumentare la priorità dei processi che restano per troppo tempo in coda

Il **controller** è un piccolo circuito che controlla l'hardware puro, contiene alcuni registri che vengono usati dal processore per inviare istruzioni a quella specifica periferica
Il **driver** è la parte del sistema operativo che dialoga con i controller
Solo il driver fa parte del SO

Frame allocation è quanti frame devo dare ad un singolo processo. Non è possibile è effettuare una scelta arbitraria del numero di pagine da associare a un certo processo, è comunque necessario un numero minimo.
Equal allocation = a tutti i processi si da lo stesso numero di frame, è eccessivo perché alcuni processi hanno esigenze maggiori rispetto ad altri
Proportional allocation = frame allocati in modo proporzionale alla taglia del processo stesso
Allocazione per priorità = assegnare una quantità di frame ai vari processi che è proporzionale alla priorità dei processi stessi

Il problema della crittografia a chiave pubblica è che non si può verificare la firma e quindi il mittente può non essere chi dice di essere.