Filer

- Data som bevaras över omstart och avstängning
- Implementeras som ett antal diskblock
- Kan ses som
  - ett antal block
  - en sekvens av tecken el. Bytes

Filattribut (exempel)

- namn
- typ (text, program, bild, ...)
- versionsnummer
- storlek (bytes/block)
- accesskydd
- tid för skapande, senaste skrivning, läsning...
- skapare, senaste skrivare, läsare,...

Filattribut sätts/ändras

- Vid skapande:
  - t.ex. namn, typ, skapare, tid för skapande
- Indirekt vid operationer på filer:
  - t.ex. senaste skrivning, storlek
- Explicit genom speciella operationer:
  - t.ex. accesskydd

Filers struktur

Olika struktur för olika ändamål, t.ex.

- Textfiler:
  - sekvens av tecken
- Databasfiler:
  - B–träd
- Programfiler:
  - information för laddaren + relokerbar kod

Struktur kontra operationer

Olika struktur kräver olika operationer, t.ex.

- Programfil:
  - random access (direkt åtkomst på adress X)
- Textfil:
  - sekventiell access (nästa tecken)
- Databasfiler:
  - trädaccess (höger, vänster, vänster, höger...)

Stöd för många strukturer komplicerar OS

- UNIX: en; MacOS två; VMS tre; IBM många

Typ−attributet

- Typen kan markera
  - vilken struktur en fil har
  - olika tolkning av samma struktur (ex. program/text)
- Typen kan vara
  - synlig som del av namnet (MS−DOS, TOPS−20)
  - osynlig (ex. MacOS typ/ägare)
- UNIX har inga filtyper!
  - men filnamnsändelser (extensions)
  - och "magiska tal" för att gissa filinnehåll

Versions/generationsnummer

- Anger olika generationer av samma fil
- Kan anges explicit eller implicit
  - ökande nummer varje gång den skrivs
  - högsta numret (senaste versionen) vid läsning
- Användes t.ex. i VMS, TOPS−20
- Efterliknas under UNIX av Emacs ".~3~" etc

Operationer på filer

- skapa
- ta bort
- existerande filer:
  - läsa
  - skriva över
  - lägga till (append)
  - ändra attribut
  - avkorta (trunkera)
  - flytta läs/skrivpositionen i filen

Att öppna en fil

1. Lokalisera filen i filsystemet
   - returnera ett "handtag" direkt till filen
2. Öppna filen via "handtaget"
   - allokera buffrar, initiera filpekare, etc
3. Utför operationer
4. Stäng filen
   - töm buffrar, uppdatera disk–innehållet
5. Släpp handtaget

När skapas en fil?

- UNIX:
  - filen skapas när den öppnas för skrivning
  - ändringar syns direkt för alla
- TOPS–20, Andrew:
  - filen skapas när den stängs efter skrivning (kan "ångra" skapandet)
  - ändringar syns bara i för de som öppnar filen efter detta
  - underlättas av versionsnummer

Filkataloger (directories)

- En filkatalog är ett register över filnamn och filernas platser på disken
- Struktur:
  - en nivå (tidigt MacOS, MS–DOS)
    - bara en katalog – namnkollisioner och kaos
  - två nivåer (tidiga fleranvändarsystem)
    - t.ex. en katalog per användare – lokaliserade problem
  - trädstruktur
    - en katalog kan innehålla både filer och kataloger

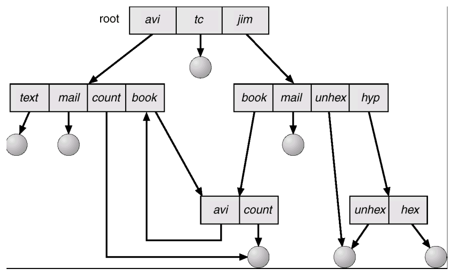
Filkataloger

- Attribut: samma som filer, minus typ och version
- Operationer:
  - skapa/ta bort (katalog)
  - lokalisera fil i katalogen
  - lista filer
  - skapa fil
  - ta bort fil
  - döp om fil
  - ändra attribut (på katalogen)

Länkar

- En fil (eller katalog) kan ha flera namn
  - "hårda" länkar direkt till filen
  - "mjuka" (symboliska) länkar via annat filnamn
- Vad händer när en fil ska tas bort?
  - hårda länkar: referensräknare
  - mjuka länkar: kan peka på intet
- Hantera cykler i trädstrukturen
  - när blir referensräknaren 0?
  - UNIX: användare får inte göra hårda länkar till kataloger

General Graph Directory



Accesskydd i filsystem

- Viktigt inte bara i fleranvändarsystem
- Kopplat till (grupper av) operationer
  - read, write, execute, append, list, delete, ...
- Mest generellt: accesslistor (ACL)
  - lista av <användare,rättigheter> eller <grupp,rättigheter> för varje fil/katalog
  - flexibelt men komplext och långsamt
- Enklare och kompaktare: tre kategorier
  - ägare, grupp, övriga

Filskydd

- UNIX: varje fil har en ägare och en grupp
  - r,w,x för ägare, grupp, övriga (3*3 = 9 bitar)
  - samma bitar men annan semantik för kataloger:
    - r – lista filers namn (läs katalogen)
    - w – skapa/ta bort filer (skriv katalogen)
    - x – läs filattribut och öppna filer (om filen tillåter det)
- TOPS−20: endast katalog har grupp
  - 3*6 = 18 bitar (alla används inte)
  - filer: läs, skriv/ta bort, exekvera, append, list
  - kataloger: använd filer, ändra filskydd, skapa filer

Typer av I/O

- Programmerad I/O (via CPU)
  - enkla former med "busy wait" bara på enkla datorer
  - vanligare: enheten genererar avbrott när I/O klar
  - vanligt för långsamma enheter, t.ex. Terminal
- DMA (Direct Memory Access)
  - enheten skriver direkt i minnet (inte via CPU)
  - ge avbrott när hela överföringen klar istället för byte–för–byte
  - vanligt för diskar, nätverk

Typer av I/O

- Synkron I/O
  - processorn väntar tills I/O är klar
- Asynkron I/O
  - processorn gör annat och får avbrott när I/O klar

I botten är all I/O asynkron (i princip)

På användarnivå oftast synkron

Typer av enheter

- Block–enheter
  - läs/skriv blockvis, t.ex. en sida
  - ex. diskar, bandstationer
- Tecken–enheter
  - läs/skriv en ström av tecken/bytes
  - ex. terminaler, skrivare
- Speciella enheter
  - ex. klockor (avbrott)

I/O–enheter

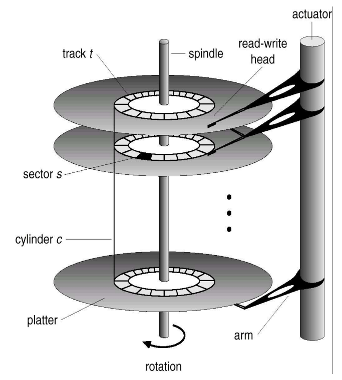
Två delar

- elektromekanik (disk drive, tangentbord)
- device controller
  - kan kontrollera flera enheter
  - ansluten via sladd eller buss
  - kommunicerar med OS
  - abstraherar elektromekaniken
    - "läs spår T, huvud H, sektor S" => flytta huvud etc.

OS–delen av I/O–enheter

- Device driver:
  - enhetsberoende kod
  - kommunicerar med device controller
  - abstraherar ytterligare:
    - "läs block B" => "läs spår T, huvud H, sektor S"
- Device–oberoende kod
  - gemensam för en klass/typ av enheter
  - översätt enhetsnamn till controller, accesskydd, buffring, felrapportering
- User–space–kod
  - biblioteksrutiner

Hårddiskar



Hårddiskar

- Geometri:
  - C: antal spår (cylindrar)
  - H: antal huvuden (spår per cylinder)
  - S: antal sektorer per spår
- Adressering:
  - 3–dimensionellt:
    - spår, huvud, sektor
  - sekventiellt:
    - block = spår*H*S+huvud*S+sektor

Diskar

- Parametrar
  - fasta/utbytbara skivpackar
  - en/fler skivor
  - huvuden svävar/är i kontakt med mediet
  - enkel/dubbelsidiga
  - en/flera armar per sida
  - 20–4096 spår per sida
  - 4–100 sektorer per spår
  - 32–4096 bytes/sektor
  - kapacitet 32k – 50+G byte

Diskar

- Hastighet
  - transfer rate:
    - överföringshastighet mellan disk och minne (Mb/s)
  - positioning time:
    - seek time:
      - tid det tar att flytta huvudet till rätt spår (millisekunder)
    - rotational latency:
      - tid innan rätt sektor passerar under huvudet (millisekunder)

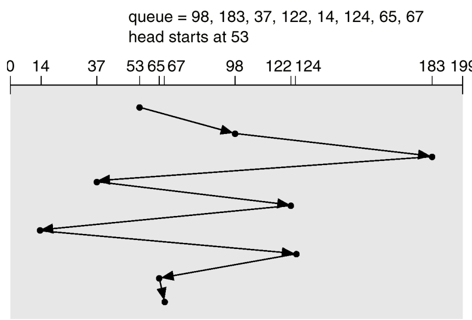
    (jfr minnesaccess: mäts i nanosekunder)

Disk–schedulering

- Hur kan vi minimera disk–I/O–tiden?
  - Tiden beror till stor del på avståndet mellan sektorerna
  - Givet en kö av läs/skriv–operationer med cylindernummer, avgör vilken ordning de ska utföras för att minimera I/O–tiden (och ge rättvisa)

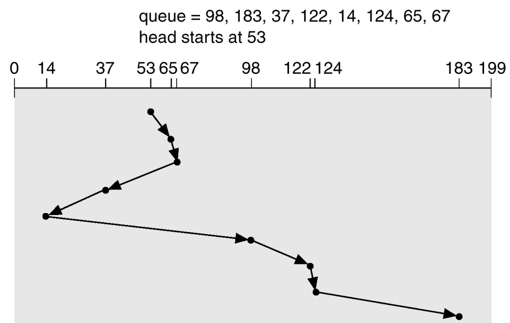
FCFS

- Rättvis, men inte snabbast (ex: 640 cylindrar)



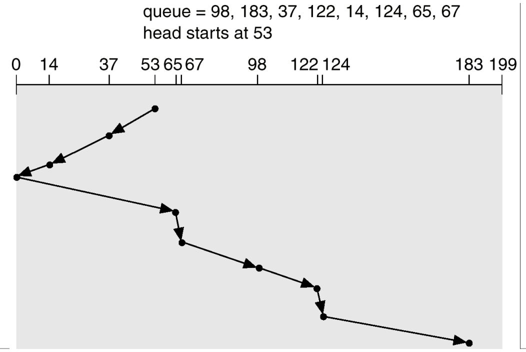
SSTF

- Shortest Seek–Time First
  - välj den i kön som är närmast
  - snabb (ex: 236 cyl), men risk för svält



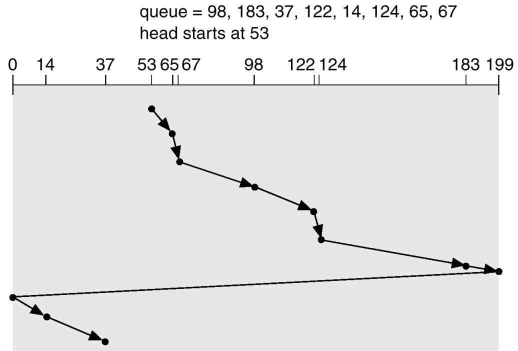
SCAN

- "Hiss−algoritmen"
  - låt huvudet gå från ena änden till andra och tillbaka
  - snabb (ex: 236 cyl), ojämnt fördelad väntetid



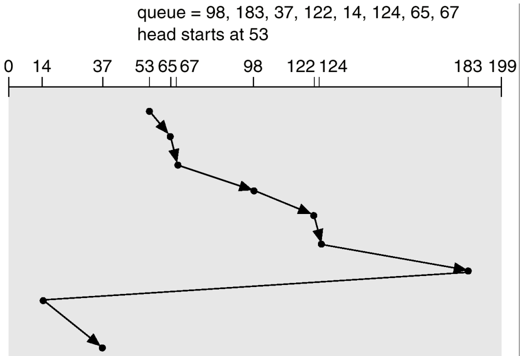
C−SCAN

- Cirkulär SCAN
  - utför inget arbete på tillbakavägen
  - långsammare (ex: 383 cyl), jämnt fördelad väntetid



C−LOOK

- Circular Look−ahead
  - som SCAN, men gå inte ända till kanten i onödan
  - snabbare (ex: 322 cyl), jämnt fördelad väntetid



Hur välja algoritm?

- SSTF: enkel, effektiv
  - men inte bra för hårt belastade system (svältrisk)
- C−LOOK
  - bättre för hårt belastade system
- FCFS
  - duger för lågt belastade system (alla algoritmer blir FCFS om kön bara har ett element)
- Optimal?
  - Inte värt besväret!

Fördelning av diskutrymmet

Vad ska få plats?

- Boot–rutiner för systemet på viss fast plats
- Swap space
  - separat partition på disken, eller speciell fil
- Filsystem
  - filer och kataloger

Organisationen påverkar söktider, och påverkas av diskscheduleringsalgoritmen

Formattering

"On a clear disk you can seek forever"

En tom disk behöver struktur:

- Fysisk formattering:
  - skapa sektorer med ECC (på fabriken)
  - hitta dåliga sektorer
- Logisk formattering:
  - dela upp utrymmet i partitioner för boot, swap, filsystem
  - skapa filsystem

Ledigt utrymme

- Bitvektor (bit map)
  - om bit *n* är satt, så är block *n* ledigt
  - bara praktiskt på disketter (få sektorer)
- Länkad lista
  - varje ledigt block innehåller pekare till nästa
  - grupperad:
    - ledigt block innehåller nr på andra lediga
  - med räknare:
    - första av flera sammanhängande lediga innehåller antalet

Upptaget utrymme

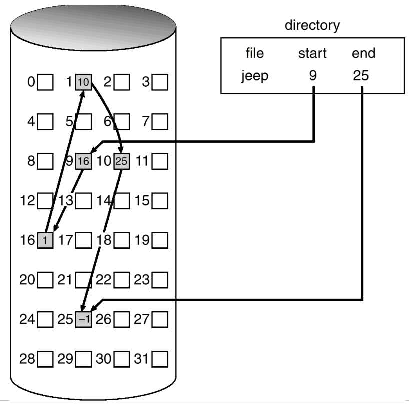
Metoder för att allokera *n* block på disk

- Sammanhängande (contiguous)
  - effektiv åtkomst
  - bör veta storlek i förväg
    - överallokering, re–allokering vid ökning
  - se minnesallokering
    - first/best/worst–fit, extern fragmentering, kompaktering
    - dessutom intern fragmentering pga överallokering

Upptaget utrymme (forts)

- Länkad lista
  - allokerade block har länk till nästa (i filen)
  - sämre för direktadressering
    - många länkar att traversera, många block att läsa
  - behöver utrymme för länkfält
  - ingen extern fragmentering (jfr paging)
  - FAT (File Allocation Table)
    - ha liststrukturen separat i början av partitionen
    - kräver cachning

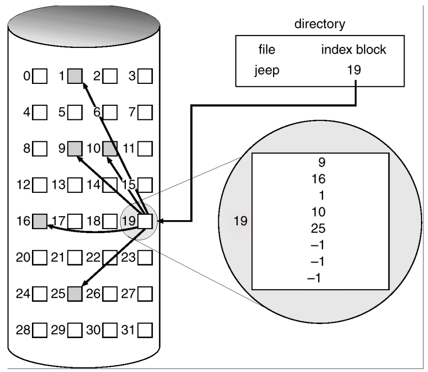
Länkad allokering



Upptaget utrymme (forts)

- Indexerad allokering
  - använd ett block för indextabell till filens block
  - snabb access till alla block (ingen pekartraversering)
  - begränsar storleken
    - antalet index som får plats i ett block
  - ökat overhead jämfört med länkad allokering
    - alltid ett visst antal länkfält

Indexerad allokering



Upptaget utrymme (forts)

- Varianter på indexerad allokering
  - Länkad lista av indexblock
    - första indexblocket pekar ut de första *n* blocken, och har vid behov en pekare till nästa indexblock
    - ingen begränsning i filstorlek
  - Flera nivåer
    - första indexblocket pekar ut *n* indexblock som i sin tur pekar på datablocken (i en eller flera nivåer)
  - Kombination (UNIX)
    - olika index har olika nivåer

UNIX inode