

Meeldetuletus!

- Hinde saamiseks peavad **KÕIK 9** ülesannet olema lahendatud tasemeni **77**
- Tunnikontrollid ja tunni aktiivsus saavad hinnat tõsta aga ainult siis kui teil on hinne!
- Täna on eelviimane loeng -> seega viimased võimalused küsida!

Sisukord

- Failid
- Kataloogid
- Faili paigutus kettal
- Hashing
- Erinevad failisüsteemid
- Hashingu ja Inode ülesanne

Tagasivaade

- Sekundaarmälude evolutsioon
 - Perfokaardid/lindid->magnetlindid->kõvakettad
- Kõvakettad mahutavad palju, odavamad aga aeglased
 - Mehaanilsied takistused
- Liikuva lugemispeaga seade
 - Info lugemine sõltub pea eelmisest asukohast ja andmete paiknemisest kettal

Tagasivaade

■ Ketta haldamine

– Otsimine optimeerimine:

- FCFS (järjekorras teenindamine)
- SSTF (lähim sektor)
- SCAN (ühes suunas liikumine. Äärest äärde. Kui lõppu, vahetab suunda)
- C-SCAN (liigub koguaeg ühes suunas. Äärest äärde)
- LOOK (ühes suunas liikumine, Kui lõppu, vahetab suunda. Vaatab “ette”)
- C-LOOK (ainult ühes suunas, vaatab “ette”)

Tagasivaade

- Pööramiste optimeerimine
 - SLTF (lühim pööramise aeg)
 - SPTF (lühim positsioneerimise aeg)
 - SATF (lühim andmete ligipääsemise aeg)
- Teised kettaste haldamise võimalused:
 - Buffrid
 - Defragmenteerimine
 - Partatsioneerimine
 - Andmete pakkimine

Tagasivaade

■ RAID

- *Redundant Arrays of Independent Disks*
- Tehnoloogia mis kasutab mitut ketast mida organiseeritakse, et anda suur töövõime ja/või kindlus
- Liiasus'
- Turvalisus
- Ketaste vöötimine
- Enamikel juhtudel kiirem lugemine, osadel ka kiirem kirjutamine

Tagasivaade

- RAID 0
 - Ei ole “päris RAID”, sest ei ole liiasust ega turvalisust
 - Saab kiiremini lugeda/kirjutada
- RAID 1
 - Ketta peegeldamine – turvalisus
 - Kiirem lugemine
 - Suur ruumi kulu
- RAID 2
 - Üks kettas paarsus kontrolliga

Tagasivaade

- RAID 3
 - XOR paarsus kood
- RAID 4
 - Vööditud kindla suuruse blokkide järgi
 - Ei ole vaja paarsusinfot enam iga lugemise jaoks
- RAID 5
 - Paarsusinfo hajutatud erinevate ketaste vahel
- RAID 6
 - 2 ketast paarsus infoga

Tunnikontroll nr 5

- isc2.dcc.ttu.ee/OpSys

HARJUTUSVÄLJAK

Learning Court

TK7



Fail

- Fail – universaalne loogiline salvestusühik
- Seotud infokogum, mis on salvestatud välismällu
- Kasutaja seisukohast: väiksem välismällu salvestatav ühik ja infot on võimalik salvestada ainult tema koosseisu.
- Programmi seisukohast: info, mida on võimalik kasutada programmide poolt ka peale antud programmi lõppu
- Low-level: 0 ja 1 jada
- High-level: numbrid, tähemärgid

Fail

- Info – salvestatud hierarhiast sõltuvalt
 - Kõige madalam hierarhia – bitid
 - 2^n võimalust n biti jaoks
 - Baidid (tavaliselt 8 biti)
 - Tähed, vastavalt märgitikule
 - Väli [*field*] (grupp tähti)
 - Kirje [*record*] (grupp välju)
 - Köide [*volume*] (andmehulk.. Näiteks mitu faili)

Fail

- Fail võib koosneda rohkem kui ühest kirjest.
- Füüsiline kirje – reaalselt loetud/kirjutatud info kettastelt
- Loogiline kirje – info kogum, mida tarkvara kohtleb kui ühikut.
- Blokeerimatud kirjed – mille puhul füüsiline kirje vastab loogilisele
- Blokeerivad kirjed – iga füüsiline kirje võib sisalda mitut loogilist

Faili atribuudid

- Faile võib iseloomustada:
 - **Nimi**
 - **Tüüp** – süsteemides, mis toetavad erinevat tüüpi failide kasutamist
 - **Asukoht** – viit asukohale füüsilisel seadmel
 - **Suurus**
 - **Kaitse** – info õiguste kontrolliks
 - **Kellaaeg, kuupäev, kasutajainfo**

Faili atribuudid

- Faili operatsioonid:

- Ava
- Sulge
- Loo
- Kustuta
- Kopeeri
- Nimeta ringi
- Sisu

Faili atribuudid

- Infole faili sees rakendatavad operatsioonid
 - Loe
 - Kirjuta
 - Uuenda
 - Lisa
 - Kustuta

Failidega seotud operatsioonid

- **Faili loomine** – tuleb leida failisüsteemis vaba ruum, luua uus kirje kataloogi
- **Kirjutamine faili** – vaja süsteemne käsk, millele edastatakse faili nimi ja kirjutav info. Süsteem otsib nime järgi faili ja peab teadma positsiooni, kust kirjutamist alustatakse

Failidega seotud operatsioonid

- **Failist lugemine** – vaja faili nimi ja mäluaadress, kuhu järgmine plokk lugeda. Samuti positsioon, kust lugemist alustada. Üldiselt: üks viit lugemisele ja kirjutamisele
- **Faili jooksva positsiooni muutmine**
- **Faili kustutamine** – kirje eemaldatakse kataloogist ja vabastatakse tema poolt hõivatud mälu
- **Faili kärpimine**

Failidega seotud operatsioonid

- Lisaks põhi op-idele:
ümbenimetamine ja lõppu lisamine
(*append*)
- Põhioperatsioonidest saadakse
vajalikud op-id: nagu kopeerimine ja
ümbertõstmine
- Avatud failide tabel

Failide tüübid

- Moodus liigendada faile kategooriatesse lähtudes sellest, mis moodi neid on võimalik kasutada
- Windows: Kasutatakse nimelõppu (nt .exe). Tabel: seob laiendi ja programmi
- Mac OS-is: faili sees info programmi kohta millega loodi
- Unix: otseselt tüüpidega ei arvastata, kaudselt lubade tunnus käivitatava faili märkimiseks

Faili struktuur

- Miks ei peaks struktuur OS-i poolt määratud olema
 - Õige tüüp?
 - Palju koodi
- Faili struktuur sõltub programmist, mis loeb ja kasutab. Elementaarühikuks on kirje
- **Fikseeritud pikkusega kirjed:** kõik faili kirjed ühesuguse pikkusega. Kõige lihtsam: vajalik kirje teades pikkus ja järjekorra numbrid korrutamise tulemusena
- **Muutuva pikkusega kirjed:** pikkus märgitakse kirje algusesse. Kompaktne, kuid optimeerimine keerukas. Leidmiseks vaja eelmised läbi vaadata

Faili struktuur

- **Määramata pikkusega kirje:** kokkulepitud koodiga kirje lõpp. Samuti kompaktne, kuid keeruline.
- MS-DOS ja Unixi OS-id ei arvesta sisemise struktuuriga. Nende jaoks väikseim ühik bait. Sisuga opereerimine rakendusprogrammidele.
- Ketta süsteemis on aga määratud ploki suurus, mida loetakse või kirjutatakse kettale.
- Seega võib fail vaadelda kui plokkide jada

Pöördumise meetodid

- Jadapöördus:
 - Infot failist töödeldakse järjest, üks kirje teise järel.
 - Nt tekstiredaktor ja kompilaatorid
 - Jooksev positsioon meeles
 - ReadNext, WriteNext, rewind
 - Aluseks magnetlindi mudel

Pöördumise meetodid

- Otsepöördus
 - Põhineb ketta mudelil
 - Fail koosneb fikseeritud pikkusega nummerdatud kirjetest
 - Lugemine ja kirjutamine suvalises järjekorras
 - Read n, Write n

Jadapöördus	Otsepöörduses jadapöörduse implementeerimine
reset	Cp=0
read next	Read cp; cp=cp+1;
write next	Write cp; cp=cp+1

Enesekontroll

- Milliste parameetritega saab faili iseloomustada?
- Kuidas erinevad OS-id erinevaid faile eristavad?
- Võrrelge fikseeritud ja muutuva pikkusega kirjeid!

Katalogid

Kataloogid

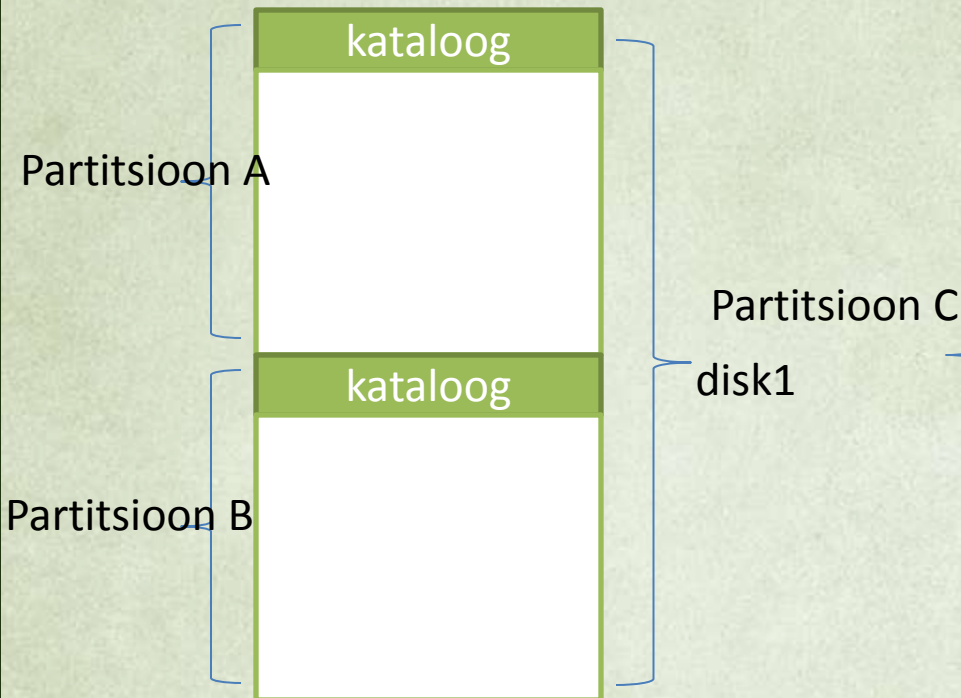
- Palju kasutajaid
- Palju faile
- Faile tuleb siiski kiiresti kasutajatele anda, kiiresti leida
- Organiseerimine -> kataloogid
- Kataloog on “erisorti” fail
- Kataloog ei hoia kasutaja infot

Kataloogi struktuur

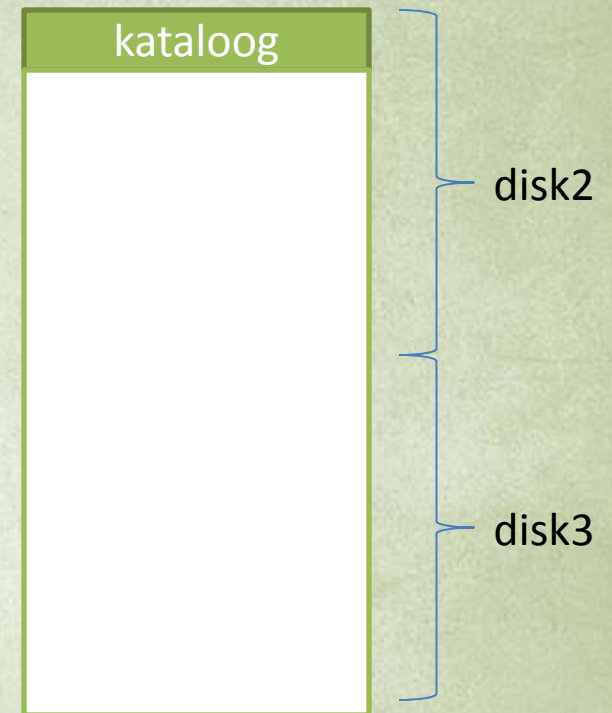
- Paljude failide hoidmine kettal vajab organiseerimist
- Tehakse kahel tasemel: failisüsteemi jagamine partitsioonideks/ kataloogid
- Partitsioonid:
 - Ühel kettal (MS-DOS)
 - Läbi mitme keta (Unix)

Partitsioonid

- Ühel ketal mitu



- Üks üle mitme ketta

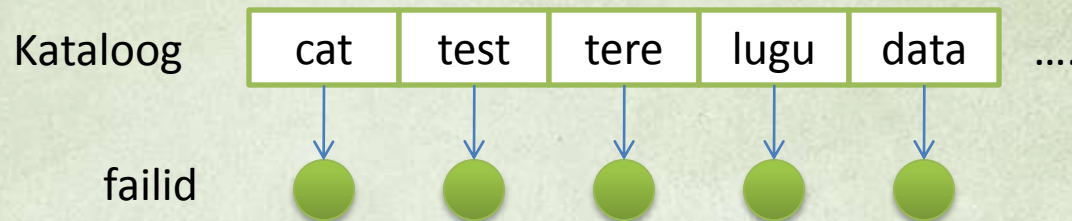


Kataloog

- Kataloogis salvestatakse failide atribuudid: nimi, tüüp, asukoht, pikkus jne..
- Kataloogide funktsioonid:
 - Faili otsimine
 - Faili loomine
 - Faili kustutamine
 - Kataloogi kuvamine
 - Faili ümbernimetamine

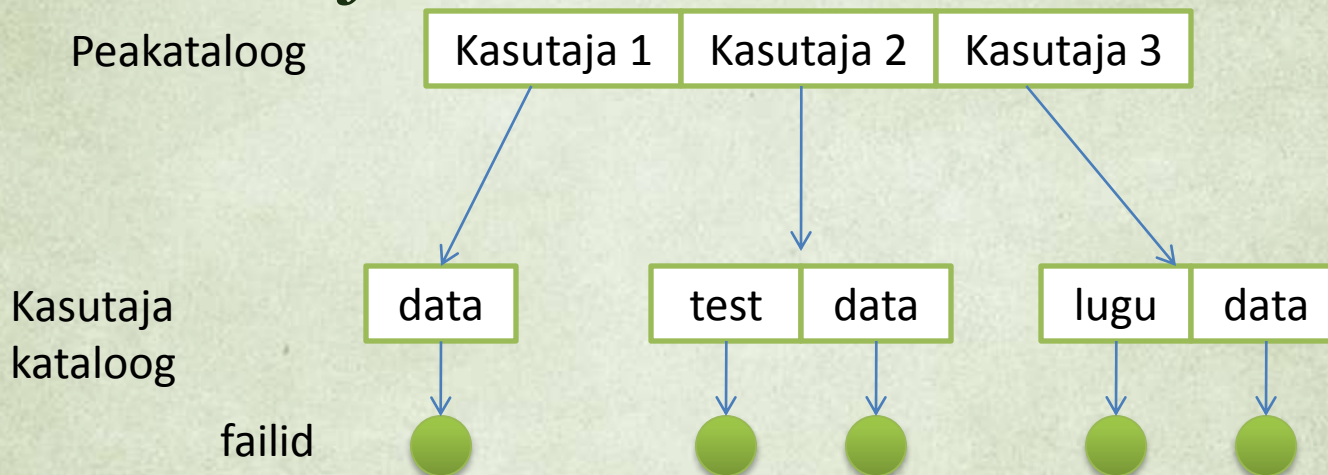
Kataloogide ajalugu

- Vanemates süsteemides: ühetasemeline. Kataloog koosneb failiatribuutide kirjetest



Kataloogide ajalugu

- Järgmine etapp: kahetasemeline
Peakataloog kasutajanimedele ja igal kasutajal oma kataloog
- Kasutajad eraladatud



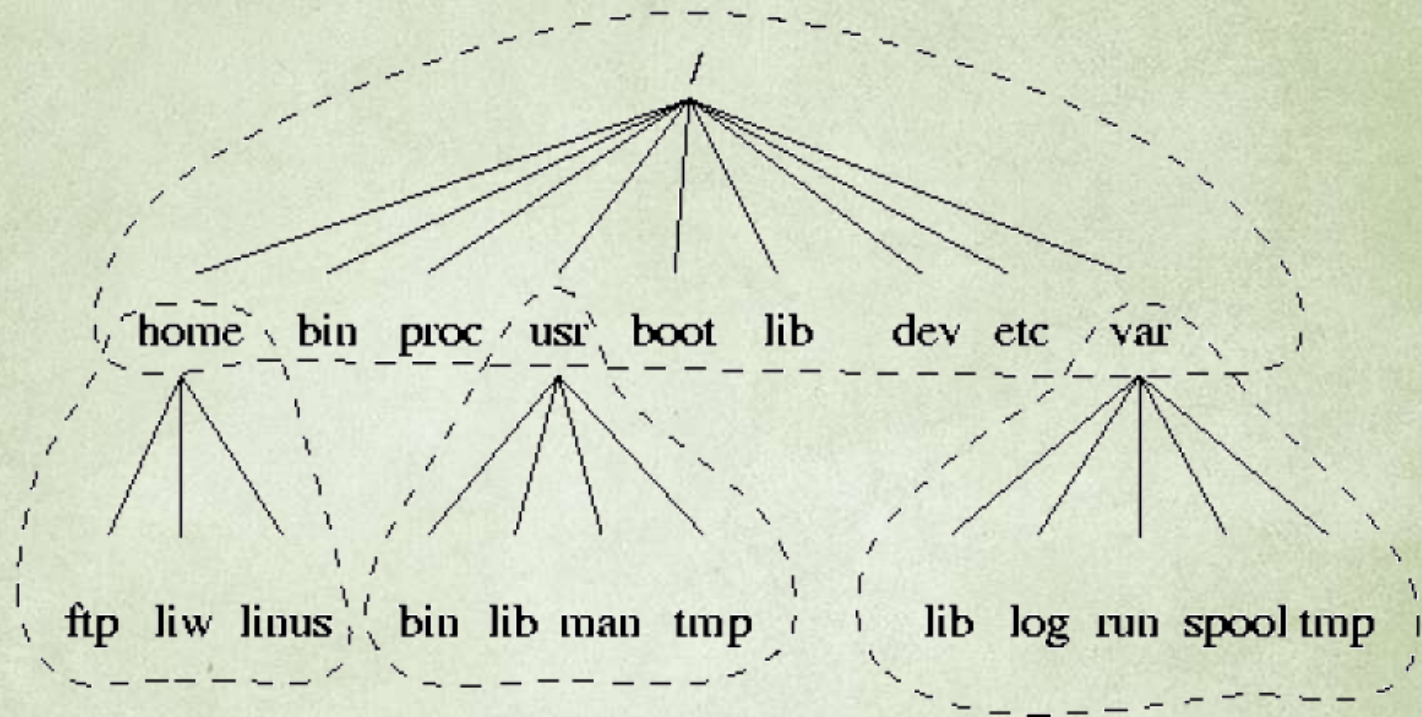
Kataloogide ajalugu

- Tänapäev: keeruline struktuur. Peab olema mugav ja samas efektiivne

Failide paigutus

- Windows
 - Ketta nimed A:, B:, C: jne
 - programmid ja konfiguratsioon samades kataloogides
- UNIX OS-id:
 - Järgivad FHS failisüsteemi paigutamise standardit
 - Failid eri kataloogides otstarbe järgi
 - Eri failisüsteemide ühendamiseks mountimine

Kataloogipuu Unixis



Mis on kasutades Unix-is?

- /boot laetava tuuma asukoht
- /mnt ühenduspunktid kettaseadmetele (floppy, cdrom)
- /bin käivitamiseks vajalikud programmid
- /sbin käivitamiseks vajalikud süsteemprogrammid
- /lib baasteegid
- /usr programmid, teegid
- /etc konf failid
- /home kodukataloogid
- /dev seadme failid
- /var muutuvad ja ajutised andmed
- /tmp ajutised failid
- /proc liides operatsioonisüsteemi tuumaga
- /lost+found leitud andmeblokid, mis ei kuulu ühelegi failile

Mis on kaust Unix-is?

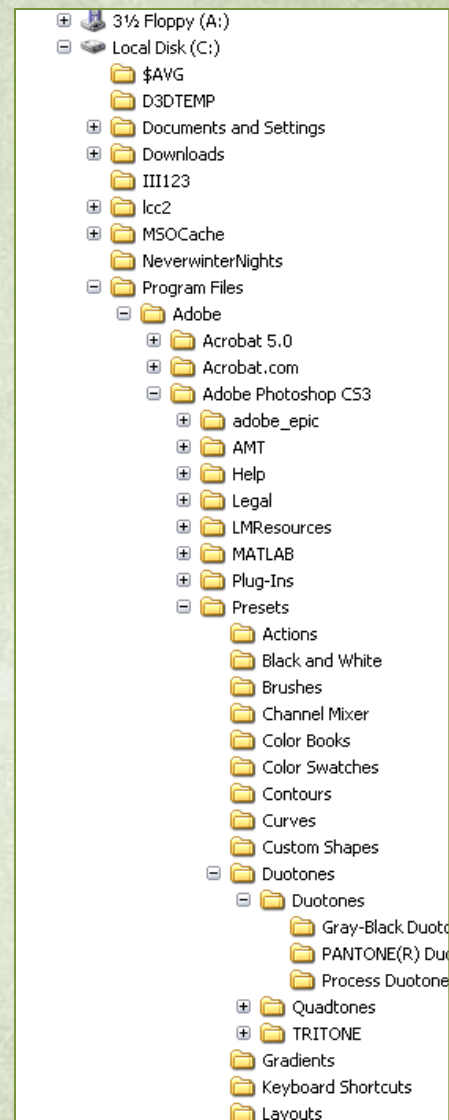
- Eritüüpi fail
- Sisaldab failinime – inode'i paari
- Moodustavad hierarhilise struktuuri
- Igas kataloogis peab olema 2 faili:
 - . Kausta enda inode'i number
 - .. Vanema kausta inode'i number

Inode

- Andmestruktuur Unixi-laadsetes failisüsteemides
- Hoiavad baasinformatsiooni failide, kataloogide ja failisüsteemide kohta
 - UID, GID, faili tüüp, viimase pöördumise aeg, muutmise aeg, sõlme muutmise aeg, suurus, juurdepääsuõigused, viidad andmeosale

Puukujuline kataloogistruktuur

- Peakataloogiks sektsiooni tase
- Iga puu tasemel kirjed, mis viitavad failidele ja alamkataloogidele
- Alamkataloog on samuti fail, millel on eritunnus ja mida käsitletakse erinevalt.
- Faili kättesaamiseks: teekond puu tippust faili
- Absoluutne vs suhteline teekond



Suhtelised aadressid

- Paljud failisüsteemid toetavad suhtelisi aadresse
 - Antud töö kataloogiga võrreldes
 - Praegune kaust: ‘.’
 - Võimalus kasutada aadresse, mis ei alga juurkaustaga
 - Kui süsteem kohtab “suhtelist aadressi”, siis teeb ta sellest enda jaoks absoluutse aadressi
- Samuti viide vanem kaustale: ‘..’

Hard vs soft link

- Hard link:
 - Samale inode-le viitab mitu kataloogi kirjet
 - Inode-s peetakse arvet viitade üle
 - Failisüsteemi piires
 - Ei saa teha kataloogidega
 - Fail kustutakse, kui viimane viit kustutakse
- Soft link (Symbolic link)
 - Seob failinime olemasoleva failinimega
 - Originaal “ei tea” viida olemasolust
 - Saab teha kataloogidesse ja teistesse failisüsteemidesse

Failisüsteemi tasemed

- Realiseeritud kihtidena, kus iga kiht kasutab oma ja alumise kihi vahendeid
- Kihid (ülevalt alla)
 - **Rakendusprogrammid** – tase, kus on märgitud faili sisu ning antakse korraldused faili avamiseks, sulgemiseks...
 - **Loogiline failisüsteem** – teab kataloogi struktuuri. Nt: käsk uue faili loomiseks: loeb kataloogi mällu, lisab uue kirje ja kirjutab kataloogi tagasi ketale ning pöördub failikorraldusmooduli poole

Failisüsteemi tasemed

- **Failikorraldusmoodul** – töötab failide loogilise ja füüsilise plokkide tasemel. Tegeleb ka vaba kettaruumi arvestusega
- **Baasfailisüsteem** – genereerib üldiseid kettalt plokkide lugemise ja kirjutamise käskude, teab kettaplokkide aadresse
nt(driver 1, silinder 23, rada4, sektor 299)
- **Sisend-väljundi juhtimine** – on madalaim tase, koosneb seadme draiveritest ja katkestuste töötlemisest. Nt (otsi välja blokk nr 123)

Enesekontroll

- Kuidas kataloog sarnaneb failiga?
- Miks ühe tasemeline failisüsteem ei ole sobiv?
- Mis vahe on hard ja soft linkil?

Inode ülesanne

HARJUTUSVÄLJAK

Learning Cloud

PAG

MEM

CPU

FSys

DLock

PgTbl

INODE

Hash

DT

Inode ülesanne

- Antud inode tabel ja andmeblokid.
- On antud loetelu faile, mida lugeda
- Vastusesse tuleb anda loetud andmeblokid õiges järjekorras
 - Esimesena vaja minev fail esimesena
 - Väiksema nr andmeblokk enne suurema nr andmeblokke
- Eesmärk:
 - Saada aru kuidas inode töötab ja miks see kasulik on

Failide paigutus kettal

Failide paigutus kettal

- Iga failiga salvestatakse välismällu ka tema juhtplokk, mis sisaldab tema atribuutide väärtusi.
- Eesmärk: kiire kättesaamine
 - Pidev paigutus
 - Lingitud paigutus
 - Indekseeritud paigutus
 - Partitsioneeritud paigutus

Pidev paigutus

- Fail hõivab kettal plokkide järjestikuse jada.
 - Positsiooni nihutamine failis nõuab minimaalset ketapea liigutamist
 - Faili kättesaamine kiire
 - Kataloogis märgitakse algusaadress ja pikkus
 - Probleem: vajaliku mälu leidmine vabade plokkide nimekirjast
 - Välimine fragmentatsioon
 - Pakkimine: nihutatakse nii, et järjestiku hõivatakse plokke - > aega nõudev
 - Fail ei saa kasvada kui tema taga ei ole vabu plokke - > ennetamiseks reserveerida suurema plokkide arvu.
- Sisemine fragmentatsioon

Lingitud paigutus

- Iga fail koosneb plokkide seotud nimistust. Plokid võivad olla kettal laialdi
- Faili pikkust ei ole vaja määrata – plokke saab juurde anda
- Kataloogis määratud faili algus ja lõpp, iga plokk viitab järgmisele
- Välist fragmentatsiooni ei ole
- Puudus:
 - töötab efektiivselt vaid jadapöörduses
 - Viidad nõuavad lisamälu
 - Turvalisus: viitade kadumine hävitab faili

Indekseeritud paigutus

- Viidad faili plokkidest ühte indeksplokki
- Igal failil oma indeksiplokk
- Kataloogis faili atribuutide hulgas on viit indeksile
- Toetab otsepöördust
- Puudus:
 - Ruumi raiskamine. Iga indeksi jaoks terve plokk. Lühikese faili jaoks liialt palju
 - Suurte failide puhul: indeksid ka mitmesse plokki, mis omavahel seotud

Vaba kettaruumi arvestamine

- Tihti kasutatakse bittide vektorit: igale plokile vastab bitt. Kui plokk hõivatud, bitt 1.
- Vaba plokki leidmine lihtne vaid siis kui vektor põhimälus, mis võimalik väikeste ketaste puhul
- Lingitud list. Mälus viit esimesele vabale plokile.
- Võib täiendada vabade plokkide grupeerimisega: esimene plokk viitab $n-1$ plokile, millest $n-1$ plokki on vabad ja n on viit järgmisele n -ile viidale

Kataloogide realisatsioon

- Iga faili poole pöördumisega kaasneb kirje leidmine kataloogist, mille kiirus mõjutab kogu failisüsteemi töökiirust.
- Lineaarne: uue faili loomisega otsitakse läbi kogu kataloog, veendudes, et antud failinime kataloogis ei ole. Uus lisatakse kataloogi lõppu.
- Optimeerimiseks paiskadresseimine (*hashing*)

Enesekontroll

- Mis on lingitud paigutuse ja indekseeritud paigutuse eelised ja puudused?
- Mis on lingitud listi ja bittide vektori kasutamisel vaba mälu haldamiseks eelised ja puudused?

Hashing

- Idee seisneb valemi leidmises, mis arvutaks kataloogis faili kirje aadressi nii, et tema kättesaamiseks piisaks sama valemi kasutamisest.
- Aadress arvutatakse faili nimest
- Kollisioon – valem annab sama tulemuse erinevate sisendite puhul
- Kollisioonid ja nende lahendamine
- Lihtsaim: kas vaba, kui mitte, kontrollitakse järgmist kirjet

Hashing

- 2 olulist algoritmi *hashing* algoritmidel
 - Primaarse paiskfunktsiooni arvutus
 - Kollisioonide lahendamine
- Primaarne funktsioon peaks
 - vältima kollisioonide tekkimist, st püüdma arvestada võtmete iseloomu (raske)
 - Piisava eelinfo puudumisel püütakse lihtsalt võtit moodustavad sümbolite koodid ära segada nii, et iga väike muutus võtmes tooks kaasa funktsiooni suure muutuse.

Näide

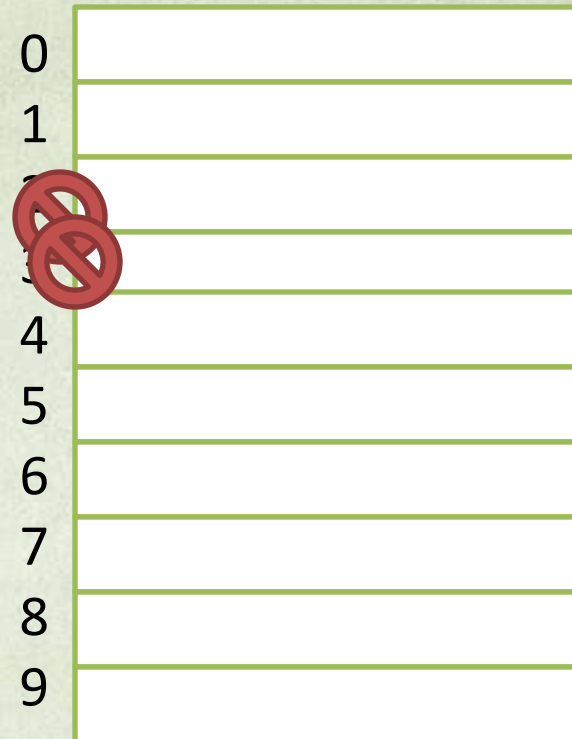
- Jagame võtme 2-baitlisteks rühmadeks, mida käsitleme täisarvudena ning liidame need arvud kokku – saame summa S
- $f = S \bmod N$ ($0..n-1$) See ongi positsioon

Kollisioonide lahendamine

- *Open addressing* vs aheltöötlus
- Palju võtteid järgmise positsiooni leidmiseks:
 1. Liikuda tabelis edasi kindla sammuga (nt samm 1 tähendaks lihtsalt liikumist järgmisele positsioonile)
 2. Leida võtmele individuaalne samm, nt leides $h = S \bmod (N-2)$. See on sisuliselt teisene hashing.
 3. Korduva kollisiooni korral arvutada uus samm.
 4. Jne

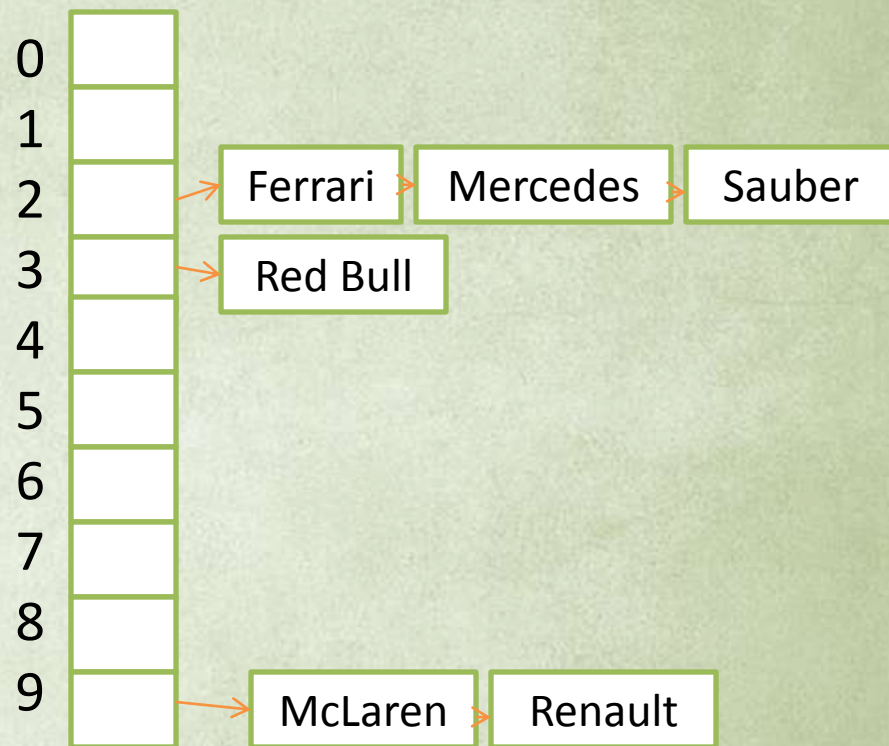
Kollisioonid nt

- Lineaarne $h(id)+1$
- Andmed:
 - Id väärtus
 - 22 **Ferrari**
 - 35 **Red Bull**
 - 95 **McLaren**
 - 24 **Mercedes**
 - 25 **Sauber**
 - 91 **Renault**
- Funktsioon:
 - $H(id)=\text{täisosa}(id/10)$



Kollusioonid nt

- Aheltöötlus
- Andmed:
 - Id väärtus
 - 22 Ferrari
 - 35 Red Bull
 - 95 McLaren
 - 24 Mercedes
 - 25 Sauber
 - 91 Renault
- Funktsioon:
 - $H(id) = \text{täisosa}(id/10)$



Hash funktsioonide näiteid

- Lühendamine:
Võtame ainult mingi osa andmest
– 925371622 -> 622
- Voltimine
– 925371622 -> 925 + 376 + 622 = 1923
- Moodul arvutus
Tabeli suurus nt 1000.
– 1923 mod 1000 = 923

Open addressing

- *Linear Probing*

- $h(k, i) = (h'(k) + i) \bmod m$

- *Quadratic Probing*

- $h(k, i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$ $c_1 \neq c_2$

- *Double hashing*

- $h(k, i) = (h_1(k) + i h_2(k)) \bmod m$

Efektiivsus

- Paiskadresseerimisel sõltub keskmine katsete arv vaid tabeli täidetusest ja ei sõltu objektide arvust.
- Täidetuse kasvades hakkab katsete arv kasvama ~70% täituvuse korral, 90% täituvus on juba kriitiline.

Enesekontroll

- Mis on kollusioon ja mida nende korral teha?
- Milliseid funktsioone võib hash funktsioonidena kasutada?

Hashing ülesanne

HARJUTUSVÄLJAK

Learning Court

PAG

MEM

CPU

FSys

DLock

PgTbl

INODE

Hash

DT

Hashing ülesanne

- 3 erinevat funktsiooni:
 - Voltimine
 - Nt $342=3+4+2=9$
 - Moodul arvutus
 - Nt $24\%20=4$ (ehk jäägi leidmine)
 - Osa võtmine arvust
 - 271 (võtame viimase numbri) $\rightarrow 1$

Hashing ülesanne

Antud

Andmed
52
968
748
144
957
780
48
247
915
908
663
400
923
353

Ülesanne

On antud järgnevad andmed. Need tuleb ära hashida kasutades **voltimist ühe numbri kaupa**. (Nt 24 voltimine: $2+4=6$). Kollisioonide korral minna lihtsalt järgmise vaba kohani.

1	2	3	4	5	6	7	8	9	10
			400			52		144	
11	12	13	14	15	16	17	18	19	20
353	48	247	923	780	915	908	663	748	
21	22	23	24	25	26	27	28		
957		968							

vastan

0 0 0 1 0 1 0 1 0 0 0 0 0 1 0
0 1 1 1 1 1 1 1 1 0 1 0 1 0
0 1 1 1 1 1 1 1 1 0 1 0 1 0

Erinevad failisüsteemid

FAT - File Allocation Table

- Laialt levinud. Vanades OS-ides, floppy-del, nüüd nt fotoka mälukaartidel
- Töötab kehvasti võrreldes teistega, kuid lihtne
 - Failide paigutus: lingitud list
 - Kataloogide sisu: tabel
 - Tabelis info kus failid, millised plokid vabad
 - Klusrtid: tabeli suuruse vähendamiseks.
 - Atribuudid: Read-only, hidden, system, volume label, subdirectory, archive, executable

Contents	Boot Sector	FS Information Sector (FAT32 only)	More reserved sectors (optional)	File Allocation Table #1	File Allocation Table #2	Root Directory (FAT12/16 only)	Data Region (for files and directories) ... (To end of partition or disk)
Size in sectors		(number of reserved sectors)		(number of FATs)*(sectors per FAT)		(number of root entries*32)/Bytes per sector	NumberOfClusters*SectorsPerCluster

FAT

- FATi failisüsteem koosneb 4: sektorist
 - Reserved sektor: kõige alguses. Baas FS informatsioon
 - FAT region: 2 koopiat failide paiknemise tabelist turvalisuse eesmärgil. Milliseid klustreid kasutatakse milliste failide ja kataloogide poolt
 - Root Directory Region: Kataloogi tabel, mis hoiab infot failide ja kataloogide kohta root kataloogis.
 - Data region: võtab enamiku partitsioonist. Tegelikud failid ja kataloogid
- Toetavad OS-id: DR-DOS, FreeDOS, MS-DOS, OS/2 (v1.1) and Microsoft Windows (.-Windows Me).

FAT 32

- Klustri suurus 32 bitti
- 28 neist klustri määramiseks ~268 millionit klustrit
- Esmakordselt Windows 95 OSR2
- FAT ei toeta failide fragmenteerumise ära hoidmist
- Vaba ketaruumi määramine üks raskemaid ülesandeid, sest vaja ketas lineaarselt läbida
- Aadressiteisendus: 2 s

NTFS

- Failisüsteemi standart Windows NT-st
 - Failide paigutus: B+ puud
 - Max failinime pikkus: 255 UTF-16
 - Atribuudid: Read-only, hidden, system, archive, not content indexed, off-line, temporary, compressed
 - USN Journal: jätab meelde kõik muutused failides, kaustades, atribuutides. Hoiab ära probleemid, kui midagi valesti läheb. Võimaldab lihtsalt *rollback*-i.
 - Kokkupakkimine L777 algoritmi järgi

NTFS

- Volume Shadow Copy: varukoopiate tegemine pidevalt.
- Encrypting File System - EFS: pakub tugevat ja kasutajale läbipaistvat krüptimist kõigile failidele ja kaustadele. (Sümmeetriline võti kiiruse pärast)
- Normistik (“Disk quotas”) : piirata kasutajate kettaruumi
- Aadressiteisendus: 100ns
- Toetavad OS-id: Windows NT perekond, Mac OS X, Linux

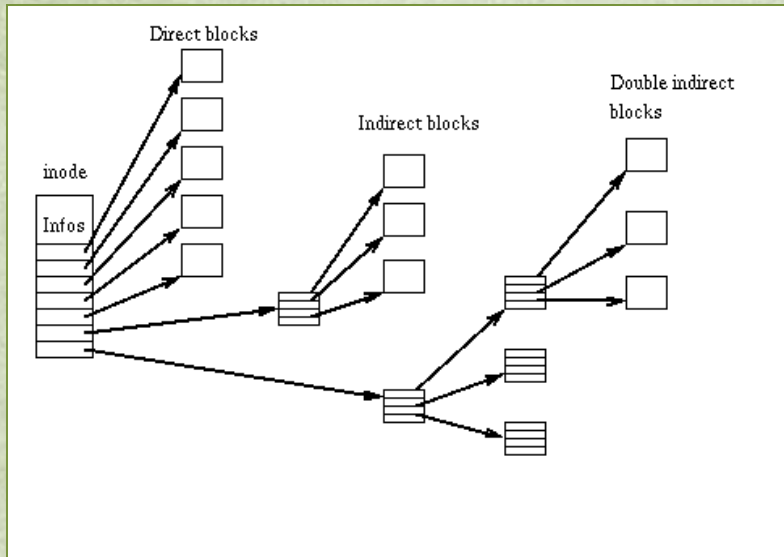
Journaling

- Peab järge muudatustest, mis tuleb failisüsteemi teha, logides, enne kui need muudatused tegelikult tehakse
- Võimaldavad taastada süsteeme kiiremalt peale elektrikatkestusi või süsteemi kokkukukkumisi.
- Suurema tõenäosusega hoitakse ära failide korrumppeerumine
- Meta-data *journaling*: kuna nõuab kaks korda kirjutamist, siis kompromissiks kiiruse ja taastavuse vahel. Võimaldab taastada failisüsteem, kuid failid võivad korrumppeeruda.

Ext2(second extended filesystem)

- Toetavad OS-id: Linux, BSD, Windows*Mac, OS X*
 - Linux'i failisüsteem
 - Failide paigutus: bitmap (vaba ruumi), tabel (metadata)
 - Max failinime pikkus: 255
 - Aadressiteisendus: 1 s
 - Ruum on jagatud plokkideks
 - Plokid kogutud gruppideks: välimise fragmentatsiooni piiramiseks
- * Läbi IFS (Installable File System)

Ext2



- Iga ploki grupp sisaldab superploki ja grupi kirjelduse tabelit. Kõik ploki sisaldavad ploki bitmapi, inode-i bitmapile ja inode-i tabelile ning seejärel tulevad tegelikud ploki andmetega
- Superplokis oluline info OS-i käivitamisel

Ext3

- Toetavad OS-id: Linux, BSD, Windows*
- Ext2 edasi arendus: “Journaled” FS
 - Max failinime pikkus: 254
 - Failide paigutus: bitmap (vaba ruumi), tabel (metadata)
 - Aadressiteisendus: 1 s
- Lubab uuendamist ext2-st ilma backupi vajaduseta
- Eelised
 - Htree indeks suurematele kataloogidele
 - “journal” failisüsteem

Ext 3

■ Puudused

- Puuduvad “modernsete” FS-ide lisavõimalused (nt dünaamiline inode-ide haldus) Vajalik ext2-ga kokku sobimiseks
- Puudub defragmenteerimise võimalus
- Probleemid failide taastamisel: failid kustutatakse inode kustutamise teel ja neid ei saa taastada
- *Journal* jaoks ei ole kontrollsummat

ReiserFS

- Linuxi “*journal*” failisüsteem
 - Failide paigutus: bitmap
 - Aadressiteisendus: 1 s
- Pakkus oma aja kohta võimalusi, mida teistes Linuxi fs-ides polnud
 - Metadata *journaling*
 - ‘Tail packing’ – sisemise fragmentatsiooni vastu
- Väikeste failidega töötas kiirelt
- Puudused
 - Mõned operatsioonid polnud sünkroonsed
 - Puu taastamine rikkus rikunud failisüsteemi veel
 - Alguses peeti isegi “ebastabiilseks”

Reiser4

- Eelised:
 - Efektiivsem *journaling* kasutades logisi
 - Blokkide alam-jaotus. Muudab väikeste failide puhul töö eriti kiireks
 - Parem kiirus kataloogidega
 - Dünaamiline optimeerimine
 - Transaktsioonide toetus
- Toetavad OS-id: Linux (Hetkel paljude Linuxi kernelite poolt aga toetamata)

XFS

- Toetavad OS-id: IRIX, Linux, FreeBSD
 - Max failinime pikkus: 255 biti
 - Failide paigutus: B+ puu (piirangutega)
 - Aadressiteisendus: 1 ns
 - 64-biti failisüsteem
 - *Journaling*: muudatused metadatasse algselt, alles siis tegelikult faili
 - Paigutusgrupid: iga grupp tegeleb omada inode-dega
 - Muutuvad ploki suurused 512 biti – 64 kb
 - Laisk mälu eraldamine (siis kui vaja)

JFS

- Toetavad OS-id: AIX, OS/2, Linux
- 64 bitine “*journaling*” failisüsteem arendatud IBM poolt
- Vabalt kasutatav/saadav GPL litsentsi alusel
 - Max failinime pikkus: 255 biti
 - Max failide arv: pole piiratud
 - Failide paigutus: bitmap(piirangutega)
 - Aadressiteisendus: 1 ns

JFS

- Võimalused:
 - Metadata *journaling* st. metadata säilib alati, aga failid võivad korrumppeeruda ikka
 - Dünaamiline inode mäluhaldus
 - Tihendamine
 - Concurrent I/O
 - Paigutusgrupid: efektiivsem I/O

Failisüsteemide võrdlus

	FAT	FAT32	NTFS	ext2	ext3	Reiser	XFS	JFS
Pääsuõigused	-	-	+	+	+	+	+	+
ACL	-	-	+	+	+	+	+	+
Block Journal	-	-	-	-	+	+	-	-
Metadata Journal	-	-	+	-	+	+	+	+
Kvoodid	-	-	+	+	+	+	+	+
Krüpto	-	-	+	+	-	(+)	-	-
Kompress	-	-	+	-	-	+	-	-
Kiirus						+	+	
Max fail	2 G	4 G	16 E	16 G	16 G	8 T	8 E	4 P
Max FS	2 G	2 T	16 E	2 T	2 T	16 T	8 E	32 P