

## Failisüsteemid

- Faili mõiste
- Juurdepääsumeedodid
- Kataloogistruktuur
- Failisüsteemide monteerimine
- Failide jagamine
- Kaitse
- Failisüsteemi struktuur
- Failisüsteemi realiseerimine
- Kataloogide realiseerimine
- Ruumi hõivamise meetodid
- Taastumine vigadest
- Logivad failisüsteemid
- Konkreetset näited

## Faili struktuur

- Ei mingit – baitide või sõnade jada
- Lihtne kirjestruktuur
  - \* Read
  - \* Fikseeritud pikkusega kirjed
  - \* Erineva pikkusega kirjed
- Keerukamad struktuurid
  - \* Formaaditud dokumendid (OLE objektivoog näiteks)
  - \* Igasugused hierarhilised andmestruktuurid (ASN.1+DER, ...)
  - \* Segmentidest koosnev käivitata fail
- Keerulisemaid formaate saab lihtsamate abil esitada
- Kelle kontrolli all on formaat?
  - \* Opsüsteemi
  - \* Rakendusprogrammi

## Faili atribuudid

- Nimi – ainus inimloetaval kujul olev metainfo
- Tüüp – vajalik süsteemides, mis toetavad erinevat tüüpi faile (enamus)
- Sisemine identifikaator – opsüsteemi sisemine identifikaator (näiteks i-kirjed UNIXis)
- Asukoht – kuskohas fail kettal asub
- Maht – faili pikkus
- Kaitse – juurdepääsuõigused
- Omanikuinfo – turvalisuse ja arvepidamise tarvis
- Muutmise, lugemise, loomise ajad – monitooringuks
- Metainfot hoitakse enamasti kataloogistruktuuris

## Operatsioonid failidega

- Loomine
- Kirjutamine
- Lugemine
- Positsioneerimine failis (*seek*)
- Kustutamine
- Pikkuse muutmine
- Kõrgema taseme operatsioonid on realiseeritavad nende kaudu
- Avamine ( $\rightarrow$  failipide)
- Sulgemine

## Kataloogistruktuur

- Kuskil peab olema nimekiri failidest
- Lihtsaimal juhul iga seadme alguses kataloog selle seadme failide kohta
- Kataloogis on iga faili kohta
  - \* Nimi
  - \* Tüüp
  - \* Aadress
  - \* Pikkus hetkel
  - \* Maksimaalne pikkus
  - \* Viimase kasutuse kuupäev (arhiveerimiseks)
  - \* Viimase muutmise kuupäev (varundamise jaoks)
  - \* Omaniku ID (kes maksab)
  - \* Kaitseinfo

## Operatsioonid kataloogiga

- Faili otsimine nime järgi
- Faili loomine
- Faili kustutamine
- Kataloogi nimekirja vaatamine
- Faili ümber nimetamine
- Failisüsteemi läbimine

## Kataloogisüsteemi organiseerimine

- Ühest üldisest kataloogist kipub väheks jääma
- Efektiivsus – tahame faili kergesti üles leida
- Nimetamine – kasutajatele mugavuse jaoks
  - \* Mitu kasutajat võivad tahta sama failinime kasutada
  - \* Sama fail võib esineda mite erineva nime all
- Grupeerimine – tahame faile loogiliselt grupeerida mingite omaduste järgi

## Kahetasemeline kataloogistruktuur

- Igale kasutajale oma kataloog
- Tipptasemel on metakataloog, kus on vaid viited kasutajate kataloogidele
- Igal kasutajal oma sõltumatu nimeruum ainult oma failidega
- Rohkem loogilist grupeerimist ei ole
- Kas teiste kasutajate kataloogidesse näeb?
- Kui näeb, siis kuidas sinna pöörduda?  
→ kataloogitee
- Kuskohas on süsteemsed programmid, mis kõigile näha on?

## Puustruktuuriga kataloogid

- Arendame ideed edasi: laseme kataloogipuu suvalise sügavuse peale
- Saavutame loogilise grupeerimise
- Kataloog on eritüübiline fail, mille sees on uued kataloogikirjed
- Kataloogipuu manipuleerimise operatsioonid olid algselt failioperatsioonid, hiljem eraldati nad kataloogioperatsioonideks:
  - \* Kataloogi loomine (mkdir)
  - \* Kataloogi kustutamine (rmdir)
- Jooksva kataloogi mõiste (igal protsessil või globaalne)
  - \* Jooksva kataloogi muutmine (chdir), jooksva kataloogi küsimine
- Kataloogitee (absoluutne ja suhteline)
- Otsimistee, *Desktop File*

## Tsükliteta graaf

- Puustruktuur keelab failide ja kataloogide jagamise erinevate harude vahel
- Kasutajad tahavad vahel jagamist (sama fail või kataloog on nähtav mitmes kataloogist)
- Puu asemel võtame kasutusele suunatud tsükliteta graafi
- Link – meetod alias-nimede tekitamiseks
  - \* Nimeviit (*symbolic link, symlink*)
  - \* Viit (*hard link*)
- Mis saab, kui viidatav objekt kustutatakse?
  - \* Kasutuskordade loendamine
  - \* Tagasiviidad linkide kustutamiseks

## Üldine graaf

- Tsüklid teevad tüli (failisüsteemi läbimisel näiteks)
- Kuidas teha kindlaks, et graafis tsükleid pole?
- Võime näiteks keelata viitamise kataloogidele
- Võime ka tsükleid lubada – paindlikkust jälle natuke juures
- Probleemid tsükli juures:
  - \* Endiselt tahaks failisüsteemi läbida nii, et iga faili näeb ühe korra
  - \* Millal võib faile kustutada?
  - \* Prahi koristamine
  - \* Graafist tsükli leidmise algoritmid kuluvad, eriti kui arvestada igal sammul kettalt lugemist

## Failisüsteemide monteerimine

- Monteerimine, ühendamine, ingl. k. *mounting*
- Failisüsteem tuleb enne kasutamist monteerida, et ta nimeruumis nähtavaks saaks
- Failisüsteem monteeritakse mingisse kindlasse punkti olemasolevas nimeruumis
- Enamasti on monteerimispunktiks tühi kataloog
- Mis saab, kui monteeritakse mittetühja kataloogi peale?
  - \* Viga
  - \* Katab vana sisu kinni kuni lahti monteerimise-  
ni
- Erinevatel protsessidel võib nimeruum (*namespace*) olla erinev – monteeritud on erinevad failisüsteemid
- Ühendmonteerimine (*union mount*)

## Failisüsteemid

- Monteerida saab failisüsteemi
- Failisüsteem võib olla mingil plokkseadmel, võrgus või üldse käigu pealt kokku pandud
- Kokku üks suur virtuaalne failisüsteem (VFS)
- Eriotstarbelised failisüsteemid – procfs, kernfs, usbfs, autofs, ...

## Failide jagamine

- Sama faili tahavad vahel mitu kasutajat/protsessi korraga kasutada
- Kuidagi tuleb faile konsistentsena hoida
- Siin kasutatakse erinevaid semantikaid:
  - \* Lukustamine (kohustuslik ja soovitatav; *opportunistic locking*)
  - \* Eksklusiivne avamine
  - \* Atomaarsus - kas üksikute lugemis- kirjutamisoperatsioonide tasemel või sessiooni (open ja close vahel) tasemel

## Kaitse

- Igal failisüsteemi objektile (failid, kataloogid, ...) on omanik
- Omaniku tähistamiseks mingi numbriline ID (UNIXis UID, Windowsis SID, ...)
- Omanik saab määrata objektile juurdepääsuõigused
- Loabitid UNIXis
- ACL (*Acces Control List*) – üldisem
- Juurdepääs kasutajate, gruppide lõikes ja ülejäänutele
- Alternatiiv: failide või kataloogidega paroolide sidumine

## Failisüsteemi kihiline struktuur

- Rakendusprogrammid
- Loogiline failisüsteem – tegeleb metainfoga (kataloogistruktuur, faili kontrollplokid)
- Failide organiseerimise meetod – teab failide asukohtadest, loogilistest ja füüsilistest plokkidest, vabast ruumist
- Plokkseadmete tase – plokkide I/O haldus, puhverdamine
- Seadmedraiverid – tõlgivad üldisi I/O käskudeks ja suhtlevad seadmega
- Seadmed

## Mida üks failisüsteem kettal hoiab?

- (Partitsioonitabel – väljaspool konkreetset failisüsteemi)
- Algladeplokk (UFS: *boot block*, NTFS: *partition boot sector*) – opsüsteemi buutimiseks vajalik info
- Failisüsteemi juhtplokk – konkreetse failisüsteemi detailne info (plokkide arv, vabade plokkide arv ja asukoht, vabade failikirjete arv ja asukoht, ...). UFS puhul *superblock*, NTFS puhul MFT (*Master File Table*)
- Kataloogistruktuur – kataloogipuu hoidmiseks
- Failide kontrollplokid (FCB – *File Control Block*) – Detailid iga konkreetse faili kohta. UFS puhul i-kirje, NTFS puhul asub MFT sees tabelis

## Mida failisüsteem mälus hoiab

- (Monteerimispunktide tabel)
- Monteeritud failisüsteemide tabel
- Viimati kasutatud kataloogstruktuuri kirjed (ja muu metainfo) puhverdamise mõttes
- Puhverdatud andmeplokid
- Süsteemne avatud failide tabel – avatud failide FCB-de koopiad
- Iga protsessi avatud failide tabel
  - \* UNIX: failideskriptorid (*file descriptor*)
  - \* Win32: failipidemed (*file handle*)

## Virtuaalne failisüsteem (VFS)

- Objekt-orienteeritud lähenemine paljude failisüsteemide realiseerimisele samas opsüsteemis
- VFS laseb sama süsteemifunktsioonide liidest kasutada paljudel erinevatel failisüsteemide tüüpidel
- Rakendusprogrammid liidestuvad VFS-ga, mitte konkreetse failisüsteemiga
- VFS pakub konkreetsetele failisüsteemidele omakorda madalama taseme liidese, mida need kasutada saavad oma info süsteemile pakkumiseks
- Konkreetseid failisüsteemi tüüpe võib vaadelda kui alamklasse, mis realiseerivad sama liidese
- VFS realiseerib ära selle osa funktsionaalsusest, mis on failisüsteemidele ühine ja antud opsüsteemis kohustuslik (näiteks UNIXilaadse semantika)

## Kataloogide realiseerimine

- Lineaarse nimekirjana
  - \* Kataloog on nimekiri failinimedest koos viitadega andmeplokkidele või FCB-dele
  - \* Lihtne programmeerida
  - \* Aeglane suure failide arvu juures
- Paisktabelina
  - \* Otsinguaeg kiirem
  - \* Kollisioonid
  - \* Fikseeritud suurus
- Puuna (tihti näiteks B-puud)
  - \* Otsinguaeg kiire
  - \* Efektivne
  - \* Keeruline programmeerida
- Kombineeritud (puu + paisksalvestus)

## Ruumi hõivamine

- Kuidas konkreetsele failile kettaplokke jagada?
- Kolm põhilist meetodit:
  - \* Pidev tükk
  - \* Lingitud paigutus
  - \* Indekseeritud paigutus

## Pideva tükina hõivamine

- Iga fail katab mingi pideva plokkide jada kettal
- Lihtne – ainult esimese ploki number ja faili pikkus plokkides on vaja meelde jätta
- Otsepöördus suvalise ploki poole
- Raiskab ruumi (dünaamiline ruumihalduse probleem nagu mälu juures)
- Fail ei saa kasvada (või on kasvamine keeruline / aeglane)
- Mitmed uuemad failisüsteemid (Veritas File System näiteks) kasutavad pideva paigutuse modifitseeritud varianti:
  - \* Kettaruumi jagatakse ulatuste (*extent*) kaupa. See on üks sidus ala. Fail koosneb ühest või mitmes ulatusest.

## Lingitud paigutus

- FCB-s on esimese ploki number
- Iga ploki sisse pannakse järgmise ploki number, nii tekib ahel
- Lihtne realiseerida
- Otsepöördust pole – ahel tuleb iga kord algusest läbida
- Klastrid
- Õrn vigade suhtes (ahel läheb kergesti katki)
- FAT (*File Allocation Table*) – viitade tabel on eraldi kettaosas
  - \* FAT on vaja mällu puhverdada liigse seekimise vältimiseks (saame ka otsepöörduse)

## Indekseeritud paigutus

- Tehtud efektiivsema otsepöörduse jaoks
- Kõik plokkide viidad tuuakse kokku ühte kohta
- Iga FCB-ga seotakse viitade plokk – viidad kõigile selle faili plokkidele
- Raiskab ruumi viitadele tervete plokkide kaupa, aga välist fragmenteerumist pole
- Lingitud indekseerimine: ahel indeksiplokkidest
- Mitmetasemeline indeks: indeksiplokkidest tehakse puu
- Kombineeritud skeem: natuke plokkide viidatakse otse, natuke 1-tasemelise puuna, edasi 2-tasemelise puuna ja ülejäänud 3-tasemelise puuna (UFS)
- Aukudega failid

## Taastumine vigadest

- Osasid andmestruktuure puhverdatakse mälus kiiruse huvides
- Aeg-ajalt tuleb seda kettale kirjutada
- Vahel läheb vool enne ära või juhtub muu jama
- Kirjutamisoperatsiooni pooleli jäämisel võivad erinevad struktuurid kettal sünkrost väljas olla
- Vaja parandada (enne järgmist monteerimist näiteks)
- Alati ei saa parandada  $\Rightarrow$  varundamine
- Inkrementaalne varundamine

## Logivad failisüsteemid

- Ingl.k. (*log-structured | log-based | logging | journaling | transaction-oriented*) file systems
- Idee võetud andmebaasidelt: kuidas hoida kettal igal ajahetkel konsistentne seis
- Näited: NTFS, Veritas VxFS, Solarise UFS, Linuxi Ext3
- Logi võib pidada nii metainfo kui kogu info kohta
- Logi võib asuda samal seadmel või hoopis teisel seadmel kiiruse huvides
- *Softupdates* – natuke teine lähenemine

## Logivate failisüsteemide tööpõhimõte

- Transaktsioon – mingi fikseeritud metainfo muutus või fikseeritud hulk andmeplokkide muutusi
- Peetakse logi transaktsioonide kohta – kõik transaktsioonid kirjutatakse järjest logisse
- Taustal mängitakse logi maha päris failisüsteemi peal
- Iga transaktsiooni pärisfailisüsteemi kirjutamise järel kustutatakse see transaktsioon logist
- Crashi järgsel monteerimisel mängitakse logist maha seal olevad terved transaktsioonid ja keritakse tagasi poolikud
- Modifikatsioon: faasipuudel põhinevad failisüsteemid
- Snapshotid

## Konkreetsed näited

- FAT (FAT-12, FAT-16, FAT-32 (28?), pikad nimed)
- HPFS, NTFS (+harud +lühikesed nimed)
- PC partitsioonitabel, LDM, EFI GUID partitsioonitabel
- UFS, FFS, FFS2, disklabel
- Minix, ext2, ext3
- Reiser3, Reiser4
- EFS, XFS, JFS, QFS, VxFS
- HFS, HFS+, Maci partitsioonitabelid
- ISO9660 + laiendused, UDF