LUENTO 10

Käyttöjärjestelmät II

TIEDOSTOJEN HALLINTA
Käytännön esimerkit

Ch 12.8-9 [Stal 05]

Ch 10.6.4, 11.6-7 [Tane 01]

Ch 20.7 [DDC 04]

Mitä KJ-I:ssä / KJ-II:ssa?

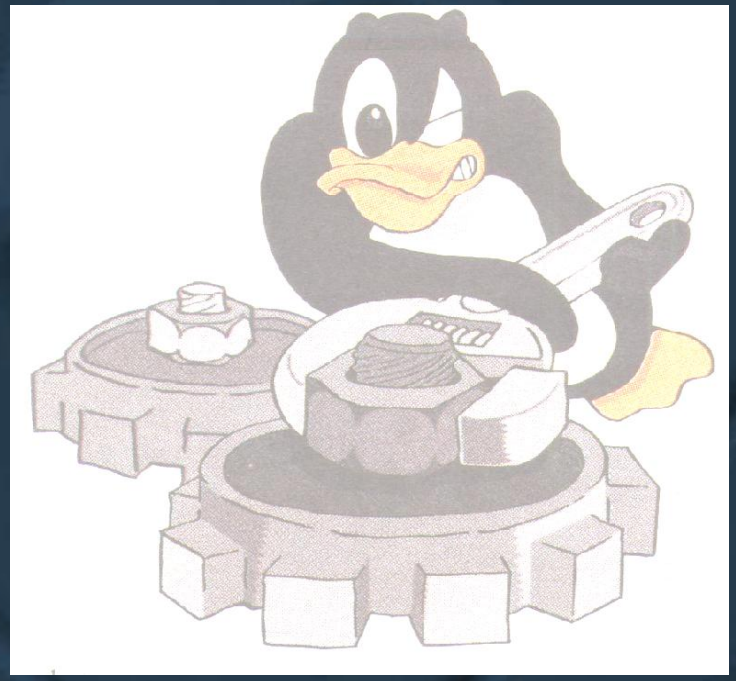
KJ-I

- Tiedostojen organisointi, hakemistot
- Tiedostojen yhteiskäyttö, tietueet ja lohkot
- Levytilan hallinta
- UNIX: tiedostojärjestelmä

Seuraavaksi KJ-II:ssa [Stal 05] [Tane 01] [DDS 04]

- Linux
  - Virtual File System (Ch. 12.8 [Stal 05])
  - ext2fs (Ch 6, Ch 11.6 [Tane 01], Ch 2.7 [DDS 04])
  - NFS, Network File System (Ch 10.6 [Tane 01])
- Windows
  - Journaling File System
  - NTFS - W2K File System (Ch 12.9 [Stal 05], Ch 11.7 [Tane 01])

Käyttöjärjestelmät II



LINUX

Tiedostojärjestelmät

Linux

- **Tiedosto = tavujono**
  - ei tietueita, ei jaksoja
  - organisointi sovelluksissa
- **Tiedostonimi ja attribuutit erillään**
  - attribuutit = i-solmu (i-node, index node)
- **Hakemisto**
  - tiedosto, jossa pareja (tiedostonimi, i-solmunumero)
- **Symbolinen linkki (soft)**
  - tiedosto, jossa tiedoston polkunimi

Fig. 6-16 [Tane 01]

Bovet D.P., Cesati M.:
Understanding the LINUX KERNEL. O'Reilly, 2001.

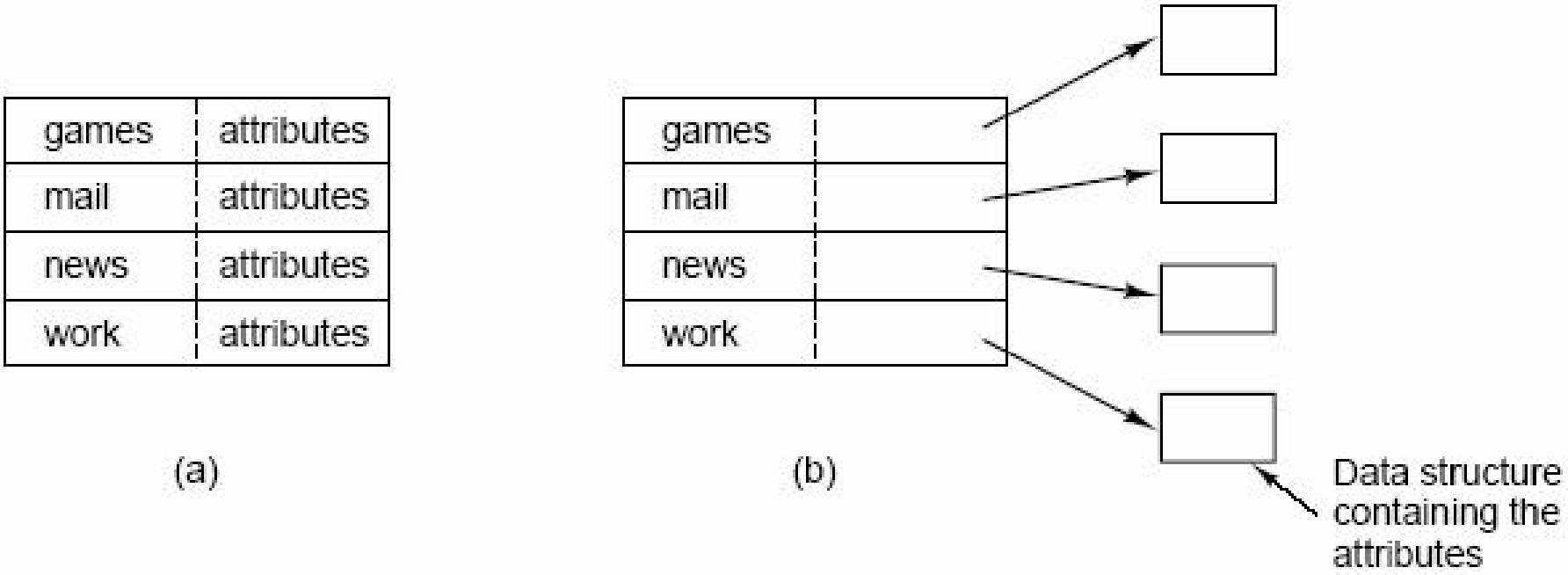


Fig. 6-16. (a) A simple directory containing fixed-size entries with the disk addresses and attributes in the directory entry. (b) A directory in which each entry just refers to an i-node.

[Tane01]

Tiedostojen yhteiskäyttö

Fig. 6-18 [Tane 01]

Fig. 6-19 [Tane 01]

- **Sama tiedosto käytössä monta kertaa**
- **Hard link**
  - monta omistajaa (vai monella omistajan oikeudet?)
  - kaikilla samat oikeudet
    - omistaja poistaa ⟶ muilla silti käytössä normaalisti
- **Soft link eli symbolinen linkki**
  - tiedoston tyyppi: symbolinen linkki
  - tiedoston sisältö: merkkijono, joka indikoi varsinaisen tiedoston
  - vain yksi omistaja
    - omistaja poistaa ⟶ muiden linkit epäkelpoja

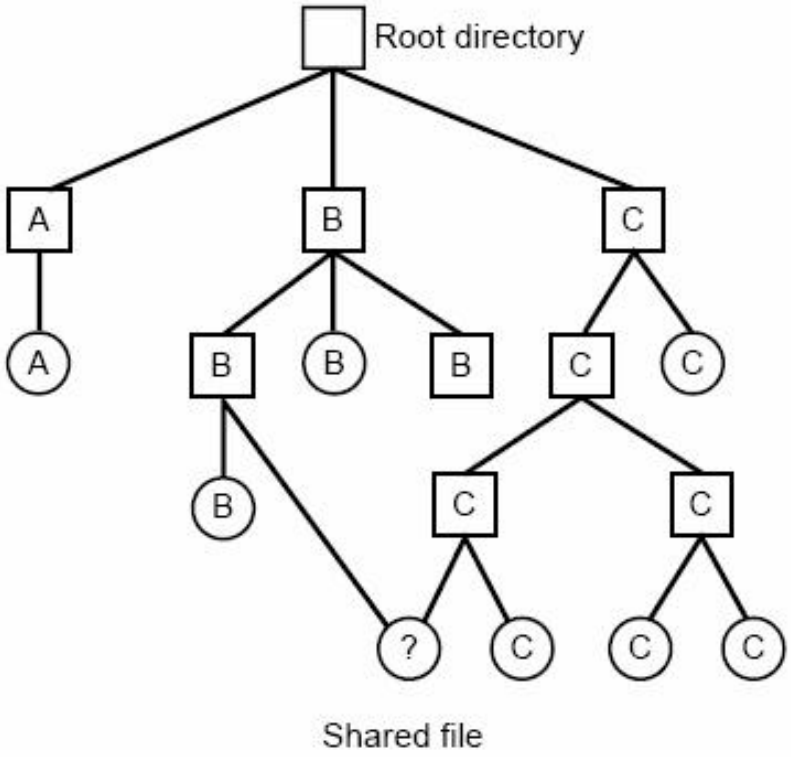


Fig. 6-18. File system containing a shared file.

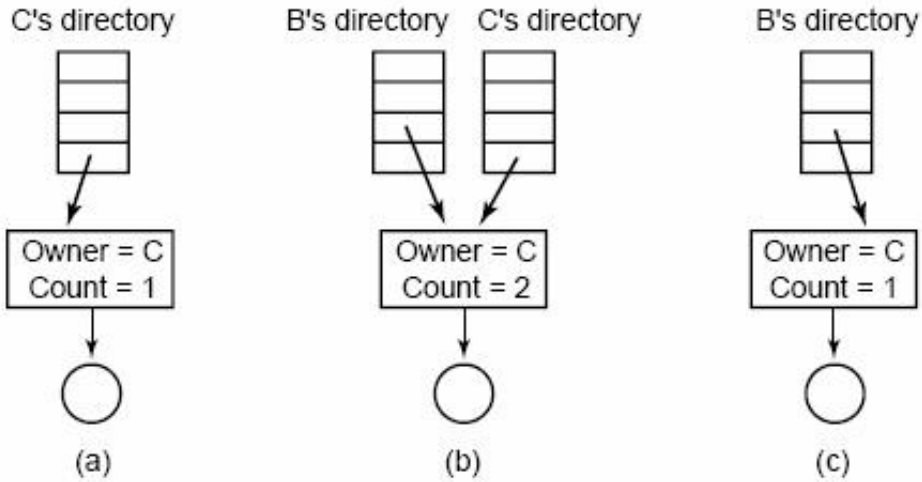


Fig. 6-19. (a) Situation prior to linking. (b) After the link is created. (c) After the original owner removes the file.

[Tane 01]

Linux VFS (virtual file system)

Fig. 12.15 [Stal 05]

Fig. 12.16 [Stal 05]

- **Tuki useille tiedostojärjestelmille**
  - register_filesystem(), unregister_filesystem()
  - ext2fs, procfs, FAT, NTFS, minix, NFS, smb, ...
  - superlohko
    - määrittelee tiedostojärjestelmän
    - oma paikka levyllä
- **Tarjoaa sovelluksille yhtenäisen rajapinnan**
  - open(), read(), write(), seek(), close(), ...
  - kaikki VFS'n kautta viitattu tieto **ei ole** levylle talletettuja tiedostoja
    - i-node
    - KJ-oliot, laitteet, yhteiset muistialueet
    - tiedon suojaus silti tiedostojen tapaan

Linux VFS rakenne

Ch. 20.7 [DDC 04]

- **VFS fd – file descriptor (file object)**
  - aukiolevalle tiedostolle, missä kohtaa lukemassa/kirjoittamassa
  - oikeudet (user ID, group ID)
  - linkki tiedoston *dentry'*yn
  - sallitut operaatiot
- **VFS dentry – directory entry**
  - osoittaa hakemistopuussa kaikkiin lähisukulaisiin (niiden *dentry*)
    - isä-hakemisto
    - lapsi-hakemistot tai –tiedostot
    - sisarus-hakemistot tai –tiedostot
  - linkki tiedoston i-node:en
- **VFS i-node (varsinainen tiedoston metatieto)**
  - tiedostojärjestelmän tunniste ja superlohko (*superblock*)
  - tiedostojärjestelmän sisäinen *i-node*

VFS metatiedon välimuistit

- **VFS ja hakemistohierarkia hidastavat käyttöä**
  - tiedot pitää yleensä hakea levyltä hakemisto kerrallaan
  - tiedostojärjestelmäkohtainen *lookup()*
- **dcache (dentry cache)**
  - viimeksi viitattujen tiedostojen dentry
  - nopea kuvaus *filename* ⟶ *i-node*
  - tiedoston X *dentry* välimuistissa ⟶ myös kaikki tiedoston X esivanhempien *dentry* välimuistissa
- **i-node cache**
  - viimeksi käytössä olleiden tiedostojen *VFS i-node*t
    - näistä löytyy tiedostojärjestelmän *i-node*

Linux tiedostojärjestelmät

- ext2fs (second extended file system)
  - Linuxia varten kehitetty tiedostojärjestelmä
  - esikuvana BSD Fast File System (FFS)
    - lohkoryhmät
  - tehokkuus, luotettavuus
- /proc
  - erikoistiedostot, luodaan 'lennosta'
  - esim. ytimen parametrien kysely/asettaminen
  - KJ-palvelut piilotettu tiedostojärjestelmän käytöksi
    - käytön valvonta tiedostojärjestelmän suojauksen avulla
- ext3fs
  - journaling file system, log-structured file system (LFS)
  - Red Hat Linux'issa

Looginen levy – yleinen tapaus

Fig. 6-11 [Tane 01]

- MBR (master boot record)
  - fyysinen sektori 0, jonka BIOS lukee
    - Basic Input Output System
    - mitä tehdään ennen alustusta tai miten alustetaan
  - partitiotaulu
    - kunkin partition alku ja loppu
    - tiedostojärjestelmän tyyppi
    - yksi partitioista aktiivinen → bootti
      - voidaan valita alustuksen yhteydessä?

Flash BIOS

Linux ext2fs levy

General: Fig. 6-11 [Tane 01]

Linux: Fig. 10-35 [Tane 01]

- Lohkoryhmät (block groups)
  - yhtenäisesti levyltä allokoitu alue
    - datalohkot ja i-nodet fyysisesti lähellä toisiaan
    - säästä hakuvarren siirroissa
- Kaikki lohkot samankokoisia (1 KB)
- Kaikki i-nodet 128B (tavallinen UNIX 64B)

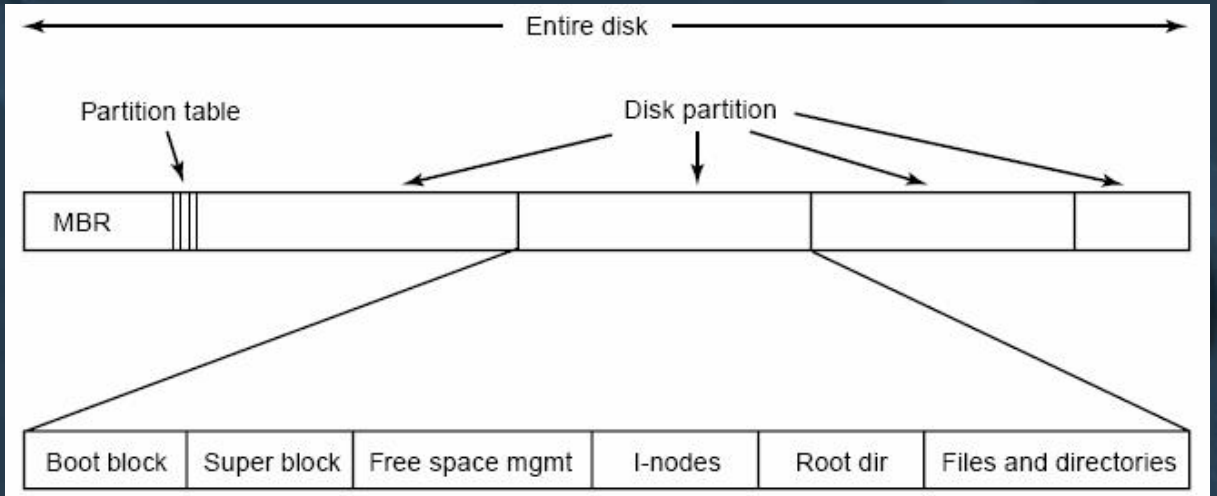


Fig. 6-11. A possible file system layout.

[Tane01]

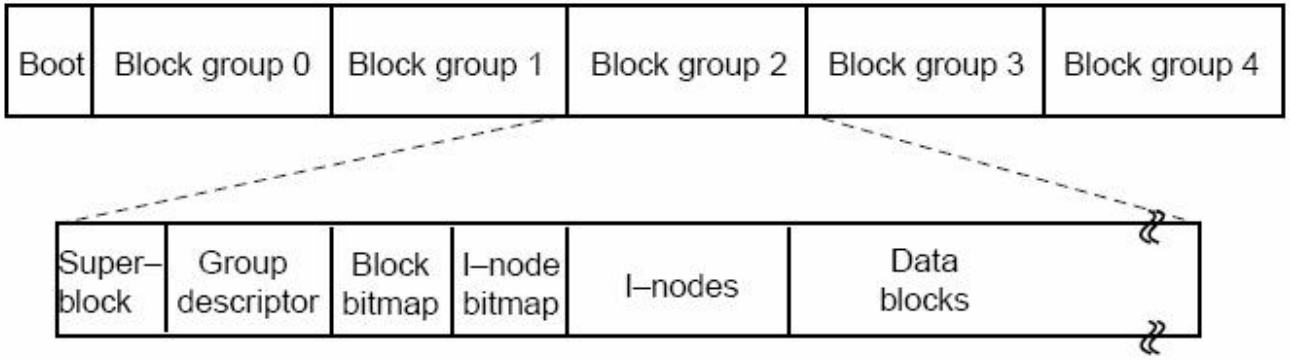


Fig. 10-35. Layout of the Linux Ext2 file system.

ext2fs superlohko (superblock)

Fig. 10-35 [Tane 01]

- **1 lohko**
- **Kuvaa koko ext2fs-partition rakenteen**
- **Kopio jokaisen lohkoryhmän alussa**
  - luotettavuus, virheestä toipuminen
- **Ydin operoi vain lohkoryhmän 0 superblokilla ja ryhmäkuvaajilla**
  - muille käyttöä, jos superblock 0 'rikki'
  - */sbin/e2fsck* kopioi aika-ajoin muualle

ext2fs superlohko (superblock)

	0	1	2	3	4	5	6	7
0	Number of i-nodes				Number of blocks			
8	Number of reserved blocks				Number of free blocks			
16	Number of free i-nodes				First data block			
24	Block size				Fragment size			
32	Blocks per group				Fragments per group			
40	i-nodes per group				Time of mounting			
48	Time of last write				Status		Max. mnt cnt	
56	Ext2signat.		Status		Error behav.		Pad word	
64	Time of last test				Max test interval			
72	Operating system				File system revision			
80	RESUID		RESGID		Pad word			
blocksize	Pad words							

ext2fs ryhmäkuvaaja (group descriptor)

Fig. 10-35 [Tane 01]

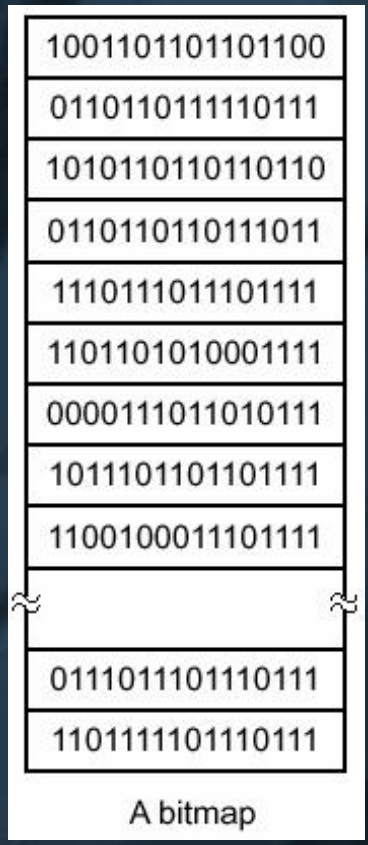
- n lohkoa
  - tietoa kaikista lohkoryhmistä
  - käyttöä valittaessa ryhmää varaukselle
  - ei perustu sylintereihin
- kopio jokaisen lohkoryhmän alussa
- yksi kuvaaja 24 B

bg_block_bitmap	lohkobittikartan lohkonro (4 B)
bg_inode_bitmap	i-solmubittikartan lohkonro (4 B)
bg_inode_table	i-solmutaulun lohkonro (4 B)
bg_free_blocks_count	vapaiden lohkojen lkm (2 B)
bg_free_inodes_count	vapaiden i-solmujen lkm (2 B)
bg_used_dirs_count	hakemistojen lkm ryhmässä (2 B)
bg_pad, bg_reserved	tyhjää (6 B)

ext2fs lohkoryhmän bittikartat

- 2 bittikarttaa
  - vapaat lohkot
  - vapaat i-nodet
- Molemmissa 8192 bittiä (1 KB lohko)
- 0 = vapaa, 1 = varattu

Fig. 10-35 [Tane 01]



ext2fs lohkoryhmän i-node taulu

Fig. 10-35 [Tane 01]

- n lohkoa
- *i-node*t á 128 B
  - tiedoston attribuutit
  - tiedoston lohkojen numerot
    - 12 suoraa viitettä lohkoihin, viite 4B
    - 3 epäsuoraa viitettä lohkoihin
- Esimerkki
  - 1 KB bittikartta ⟶ ryhmässä 8192 lohkoa tai i-nodea
  - i-noden lohkoviite 13225
    - lohkoryhmä 1, siirtymä 5033 (=13225-8192)
- root-hakemiston i-node = *i-node #2*

ext2fs i-node

Fig. 10-35 [Tane 01]

	0	1	2	3	4	5	6	7
0	Mode		Uid		File size			
8	Access time				Time of creation			
16	Time of modification				Time of deletion			
24	Gid		Link counter		No. of blocks			
32	File attributes				Reserved (OS-dependent)			
40	12 direct blocks							
88	One-stage indirect block				Two-stage indirect block			
96	Three-stage indirect block				File version			
104	File ACL				Directory ACL			
112	Fragment address				Reserved (OS-dependent)			
120	Reserved (OS-dependent)							

Access Control List

ext2fs datalohkot (data blocks)

- Lohkon koko 1 KB
  - 2 KB, 4 KB tai 8 KB?? ei.
- Suurin tiedostokoko 2 GB
  - kenttä "file size" rajoittaa (ylin bitti ei käytössä!)
  - jos 64-b kone, max 4TB
- Tiedosto voi jakautua useamman ryhmän alueelle

Fig. 10-35 [Tane 01]

ext2fs hakemistoalkio

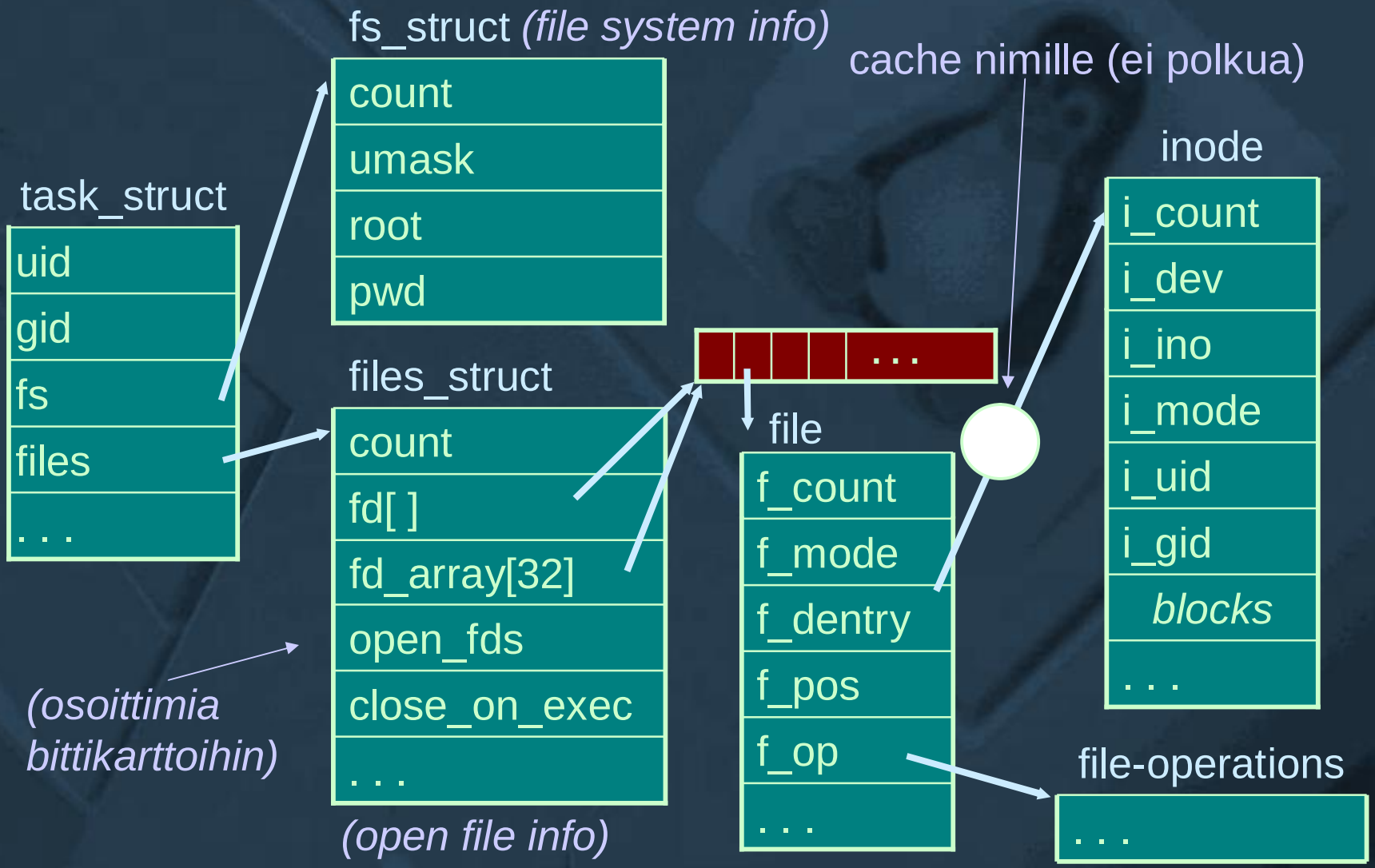
4 B	2 B	1 B	1 B	1 - 255 B (1 - EXT2_NAME_LEN)
i-node number	**entry len**	**name len**	**type**	**file name**

- Hakemisto on tiedosto, joka kuvaa tiedostonimet *i-node*iksi
  - peräkkäinen lista hakemistoalkioita (directory entry)
  - voi olla myös B-puu, jos paljon tiedostoja
- Hakemistoalkio on vaihtelevanpituinen
  - pituus aina 4:n monikerta (lopussa /0-merkkejä)
- Tyyppi
  - 0 = tuntematon, 1 = tavallinen tiedosto, 2 = hakemisto
  - 3 = merkkilaite, 4 = lohkolaite, 5 = nimetty putki
  - 6 = pistoke, 7 = symbolinen linkki

Esim: TKTL mail server

Polku tässä tiedostossa tai hakemistoalkiossa (fast symbolic link)

Linux: tiedoston käyttö (vfs)



Linux: tiedostojen käyttö (vfs)

file_operations

llseek(file, offset, whence)
read(file, buf, count, offset)
write(file, buf, count, offset)
readdir(dir, dirent, filldir)
poll(file, poll_table)
ioctl(inode, file, cmd, arg)
map(file, vma)
open(inode, file)
flush(file)
release(inode, file)
fsync(file, dentry)
fasync(file, on)
check_media_change(dev)
revalidate(dev)
lock(file, cmd, file_lock)

- **Jokaisella tiedostojärjestelmällä omat funktiot**
- **File_operations rakenteessa funktion osoite**
- **Jos ei toteuta kyseistä operaatiota, osoitin NULL**

Fig. 6-5 [Tane 01]

Linux procfs tiedostojärjestelmä

- **Process file system**
- **Ei todellinen (fyysinen) tiedostojärjestelmä**
  - kaikki keskusmuistissa, levyllä ei tiedostoja
- **Käyttöliittymä prosessikuvaajiin**
  - hakemistossa /proc
  - jokainen /proc'in alihakemisto määrittelee omat read() ja write() operaationsa
    - /proc/4321 on prosessin 4321 hakemisto
  - KJ-tietojen lukeminen ja kirjoittaminen
  - read() ja write() toteuttavat suojatun tietorakenteen
    - käytön valvonta tiedostojärjestelmän avulla
    - samanaikaisuuden hallinta

Linux sysfs

- **hakemisto /sys**
- **käyttöliittymä laitekuvaajiin**
  - unified device model
- **väylät hakemistossa /sys/bus**
  - pci laitekuvaaja hakemistossa /sys/bus/pci
- **I/O laitteet laitetyypin mukaan**
  - /sys/class/input
  - laitetyypin nimi, numero, laitteet, ajurit
- **pidetään kirjaa kaikista laitteista, jotka käytössä ja missä ne ovat**
- **pollataan aika ajoin väyliä, jos uusia laitteita tulisi tai vanhoja poistuisi**
  - hot swappable devices

Käyttöjärjestelmät II

NFS

Network File System

Ks. esim. Ch 10.6.4 [Tane 01]

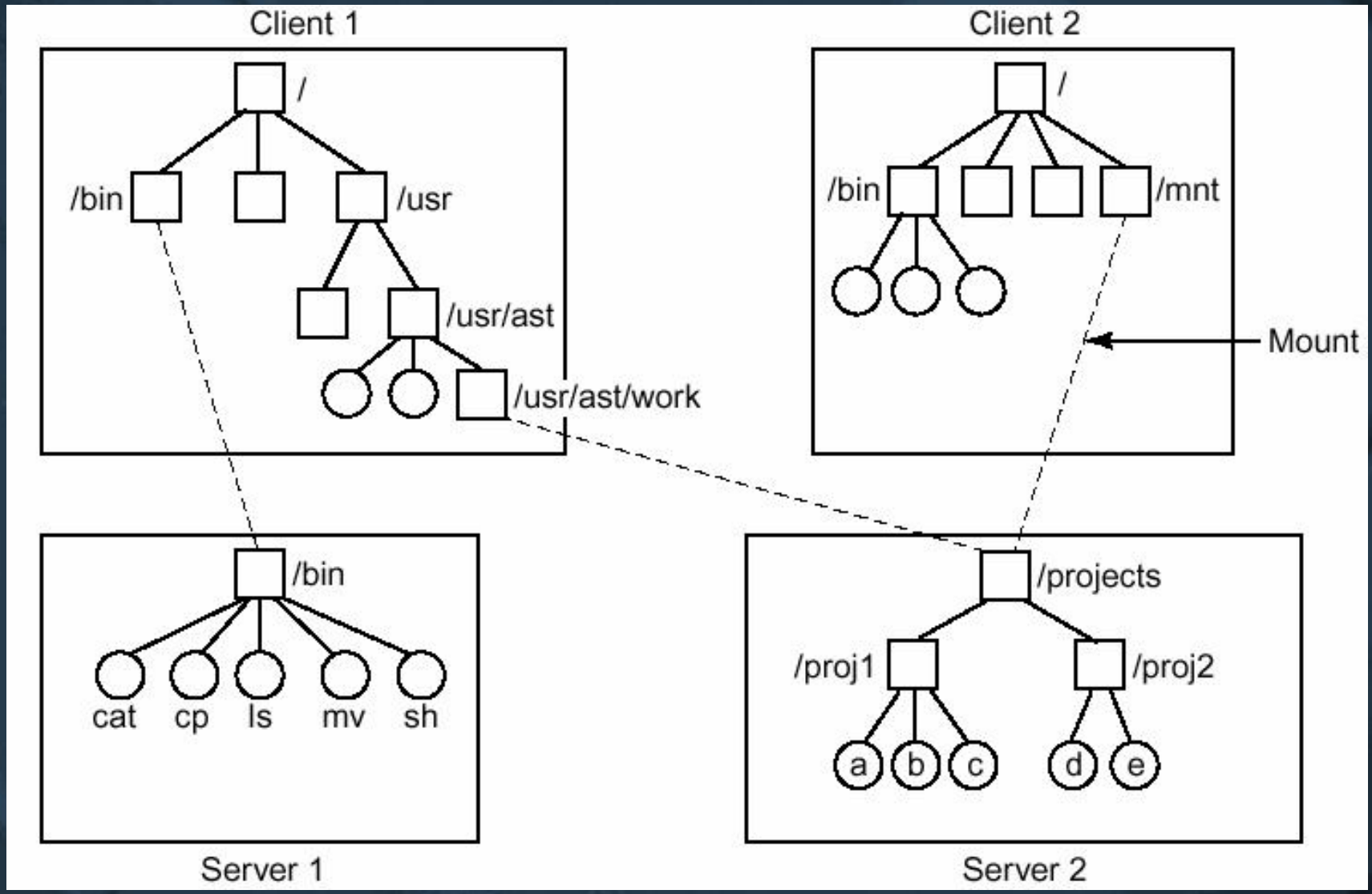
NFS

- **Etäkoneiden hakemistojen liittäminen omaan hakemistopuuhun**
  - kehittäjä Sun Microsystems
- **NFS-protokolla**
  - pyyntö-vastaus protokolla
  - ei ota kantaa siihen kuinka toteutetaan
    - NFS-palvelija, NFS-asiakas
- **Windowsin vastine SMB-protokolla**
  - Server Message Block

NFS-arkkitehtuuri

- **Etäkone (Palvelija)**
  - suorittaa NFS-palvelijaa
  - määrittelee hakemiston julkiseksi hakemistossa **`/etc/exports`**
    - mm. käyttöoikeuksien rajaaminen
- **Asiakas**
  - suorittaa NFS-asiakasprosessia
  - asemoi (”mounttaa”) hakemiston omaan hakemistopuuhun
  - mount-point määritelty tiedostossa **`/etc/fstab`**
- **VFS huomaa milloin viitataan toisessa koneessa olevaan (mountattuun) tiedostoon**
  - välitä pyyntö palvelijalle
  - palvelijan tiedostojärjestelmä ei ole tärkeä

Fig. 10-36 [Tane 01]

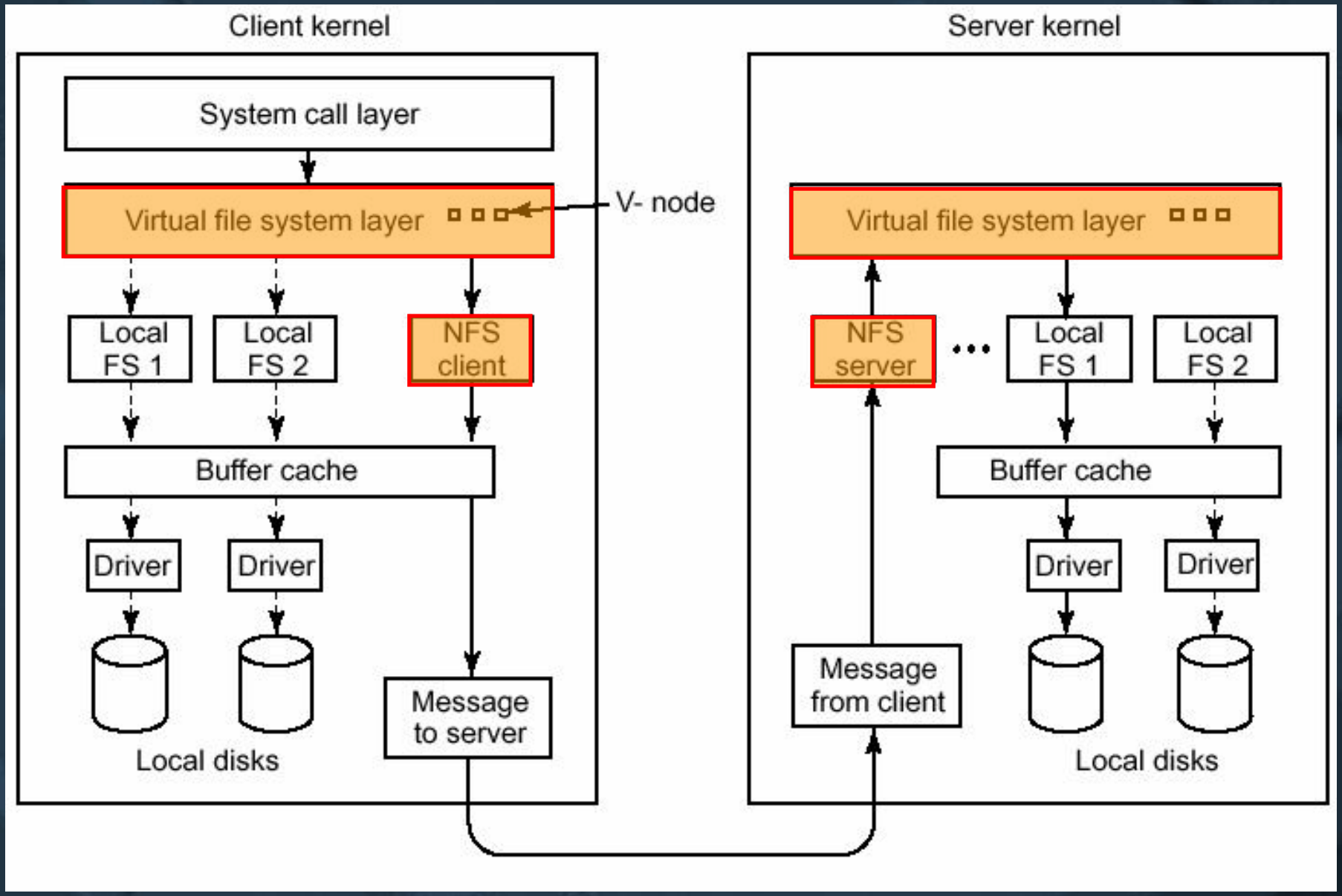


(Fig 10-36, [Tane 01])

NFS-protokolla

Fig. 10-37 [Tane 01]

- **mount**
  - asiakas lähettää polkunimen palvelijalle
  - palvelija palauttaa kahvan (file handle)
    - tiedostojärj. tyyppi, laite#, inode#, oikeudet
    - käytetään jatkossa kaikissa pyynnöissä
  - voidaan tehdä alustusskripteissä (boot)
- **automounting**
  - mountataan, kun viitataan ens. kertaa
- **pyynnöt**
  - normaalit palvelupyynnöt sanomina
    - read(), write(), ...
- **tilaton protokolla**
  - kaikki tarvittava tieto mukana pyynnössä
    - kahva, lukupositio, paljonko, …



(Fig 10-37, [Tane 01])

Käyttöjärjestelmät II

Windows 2000

Tiedostojärjestelmä

(NTFS)

LFS – Log-Structured File System

(Ch 6.3.8 [Tane 01])

- **”Uusi, parempi, fiksumpi, luotettavampi” – ihan totta!**
- **Usein tilanne**
  - paljon päivityksiä, useimmat todelliset levyviitteet kirjoituksia
    - luvut levyvälimuistista
  - useimmat kirjoitukset pieniä päivityksiä
    - levyn hakuvarsi liikkuu paljon, vähän dataa siirtyy
- **Ongelma tavallisen tiedostojärjestelmän uuden tiedoston X luomisessa:**
  - kirjoita hakemiston i-node, hakemisto, tiedoston i-node ja lopulta tiedosto
  - virta poikki (tms vika) kesken kaiken? Oooops.
- **Ratkaisu: tapahtumaloki, joka takaa tiedostojärjestelmän konsistenssisuuden**
  - pidä lokia sekä metatiedosta (esim. inode) että itse datasta (sektorit)
- **Journaling File System**
  - pidä lokia vain metatiedosta – järjestelmä säilyy konsistenssina (data ei)
- **Esim: Microsoft NTFS, Red Hat Linux ext3fs**

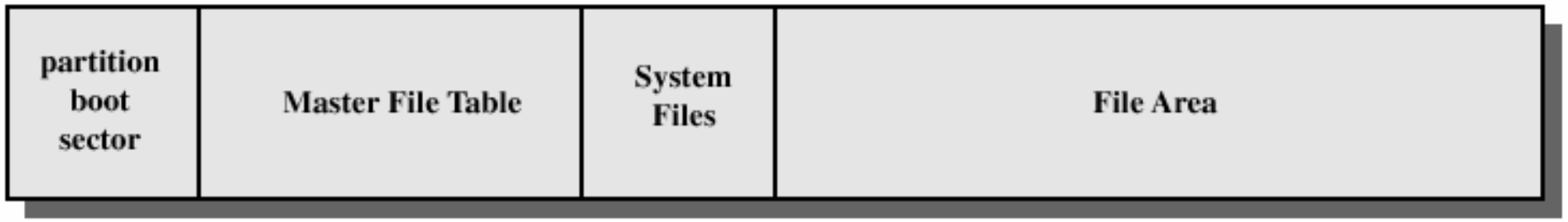
LFS – alkuperäinen idea

- **Koko levy on loki tapahtumista**
  - uudet tapahtumat kirjoitetaan ”loppuun” vapaaseen tilaan, peräkkäisiin lohkoihin
  - nopeata, levyn täysi kapasiteetti hyödynnettävissä
- **Uusi tiedosto X hakemistoon D**
  - kirjoita X:n data-tapahtuma
  - kirjoita X:n metadata (i-node?) -tapahtuma
  - kirjoita X:n hakemistoalkio D:ssa –tapahtuma
- **systeemi koko ajan konsistensissa tilassa**
- **tiedon haku hidasta**
  - ei niin paha, kun useimmat levytapahtumat kirjoittaa
  - metadata (i-node) välimuistit: **i-node map**, **superblock**
  - ***cleaner*** säie etsii tyhjää tilaa ja tiivistää lokeja

NTFS: Piirteitä

- **Kaatumisista ja levyvirheistä toipuminen**
  - LFS lokitiedoston avulla
- **Käyttöoikeudet**
  - pääsylistat (security descriptor)
- **Sallii suuret levyt ja tiedostot**
  - FAT32'ssa vain 2^{32} lohkoa, suuri allokointitaulu
- **Tiedosto-oliot ovat *(arvo, attribuutti)* -pareja**
- **Mahdollisuus indeksointiin tiedoston käsittelyn nopeuttamiseksi**
- **Lohko, cluster**
  - yksi tai useampi peräkkäinen sektori (esim. 512 B - 4 KB)
    - 32 GB levyllä 128 sektoria/lohko (→ lohko 64-512 KB)
  - varauksen ja kirjanpidon perusyksikkö
- **Partitio, volume**
  - fyysinen levyn looginen osa, jolla oma tiedostojärjestelmä

NTFS-partitio



(Fig. 12.17 [Stal 05])

- **Boottilohko**
  - partition ja tiedostojärj. rakenne, boottitietue ja -koodi
  - MFT:n sijainti
- **MFT**
  - tietoa tiedostoista, hakemistoista (folders) ja vapaasta tilasta
- **System Files (~ 1MB)**
  - kopio MFT:n alkuosasta
  - virheistätoipumisloki, bittikartta vapaat/varatut lohkot, attribuuttien kuvaustaulu
- **File Area -** tiedostojen lohkoille

NTFS – MFT

Fig. 11-36 [Tane 01]

- 1 KB:n kokoisia MFT-tietueita
  - jokainen kuvaa yhden taltiolla olevan tiedoston
    - myös hakemisto on tiedosto
  - vaihtelevanmittainen osa käytössä

Fig. 11-35 [Tane 01]

    - (attribuutti, arvo) pareja (ei paikkasidonnainen!)
    - data attribuutti, 'arvo' = lohkojen sijainti
- 16 ensimmäistä tietuetta varattu ns. metadatalle
  - 16 $-alkuista tiedostoa

Fig. 11-34 [Tane 01]

- Jos pieni tiedosto, tietue sisältää myös datan
- Jos iso tiedosto, data erillisellä tallealueella
  - MFT-tietuessa lohkonumeroita
  - kuvaus voi jatkua useampaan MFT-tietueseen

Fig. 11-37 [Tane 01]

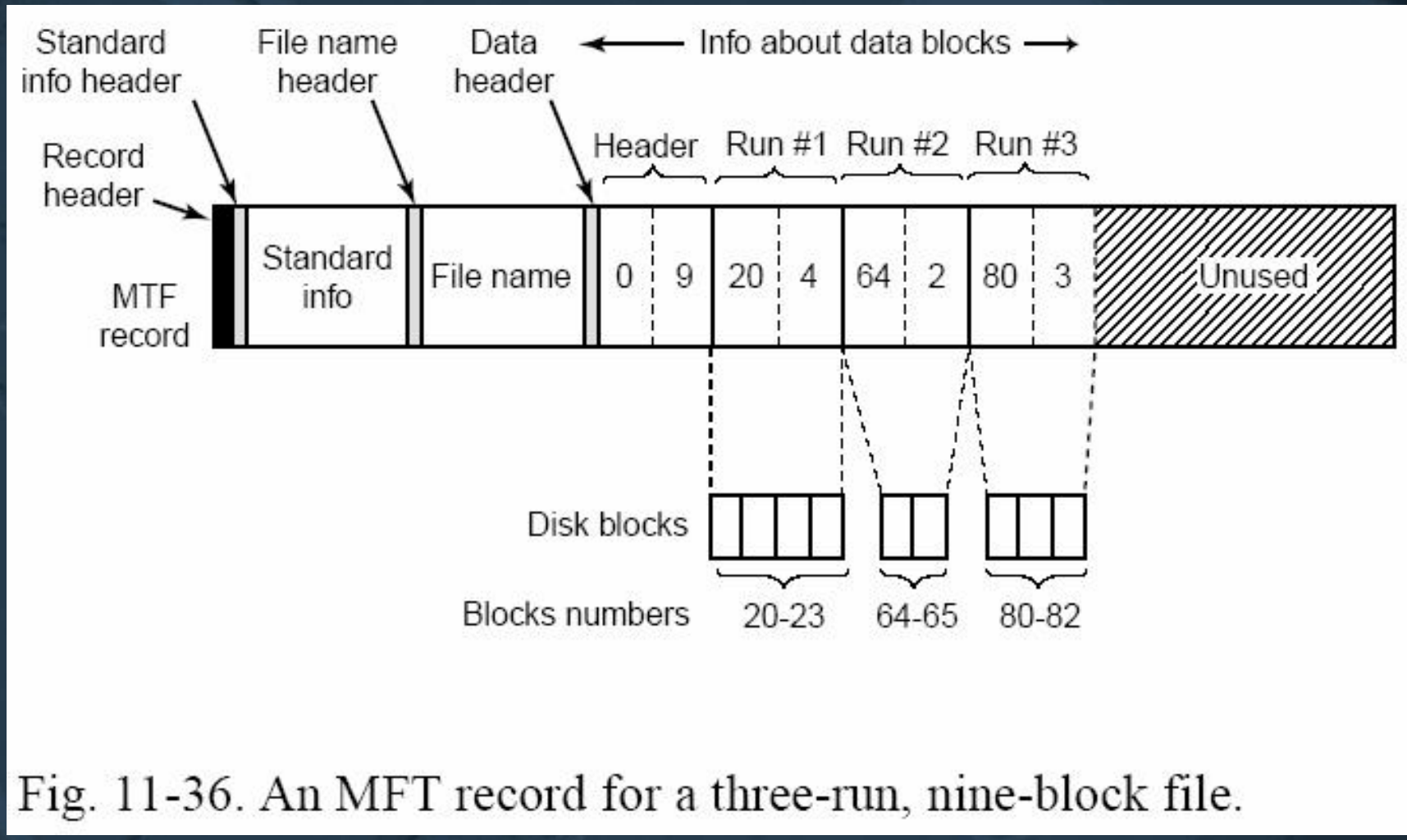


Fig. 11-36. An MFT record for a three-run, nine-block file.

[Tane 01]

Attribute	Description
Standard information	Flag bits, timestamps, etc.
File name	File name in Unicode; may be repeated for MS-DOS name
Security descriptor	Obsolete. Security information is now in $Extend$Secure
Attribute list	Location of additional MFT records, if needed
Object ID	64-bit file identifier unique to this volume
Reparse point	Used for mounting and symbolic links
Volume name	Name of this volume (used only in $Volume)
Volume information	Volume version (used only in $Volume)
Index root	Used for directories
Index allocation	Used for very large directories
Bitmap	Used for very large directories
Logged utility stream	Controls logging to $LogFile
Data	Stream data; may be repeated

Fig. 11-35. The attributes used in MFT records.

[Tane 01]

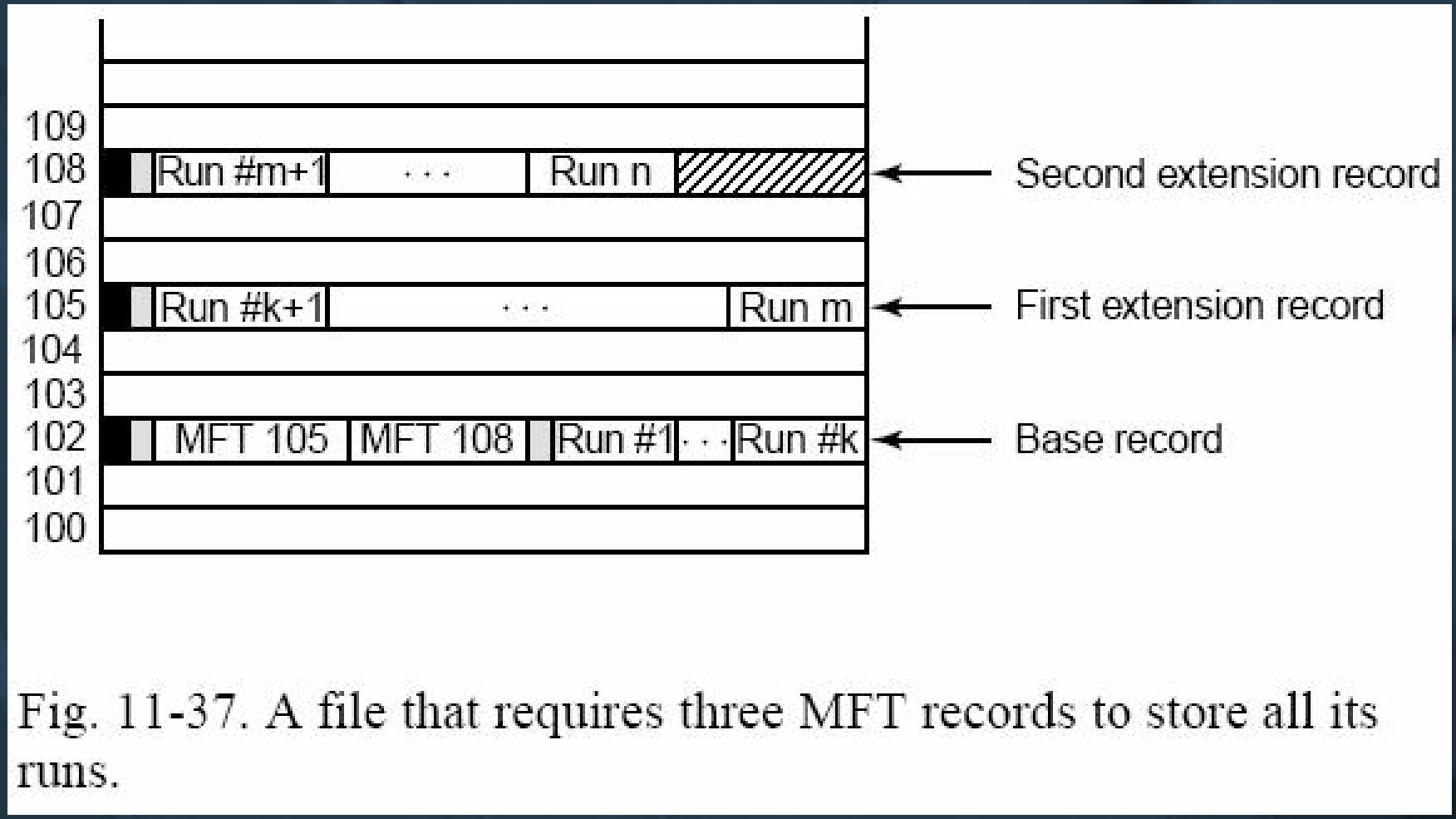
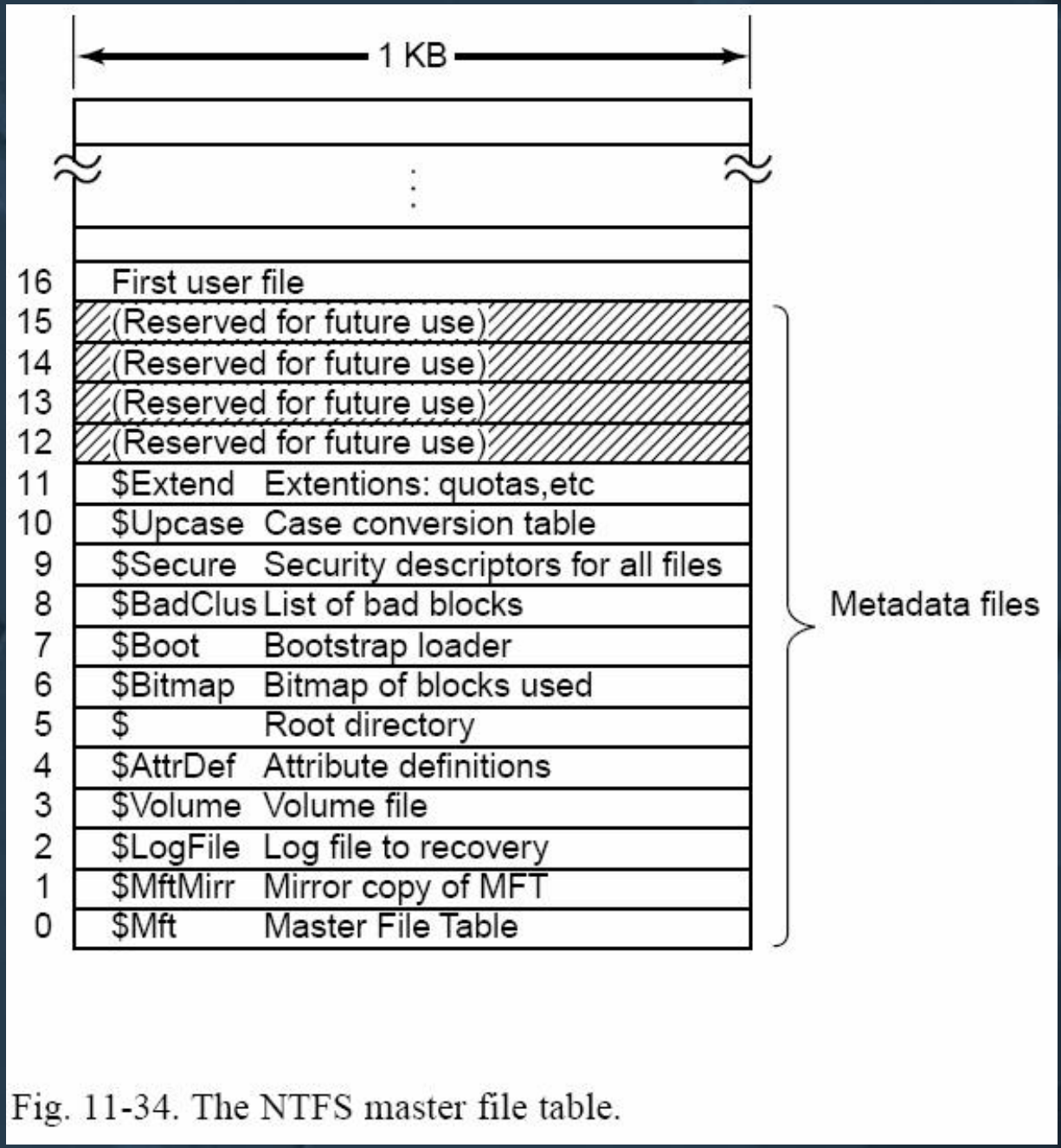


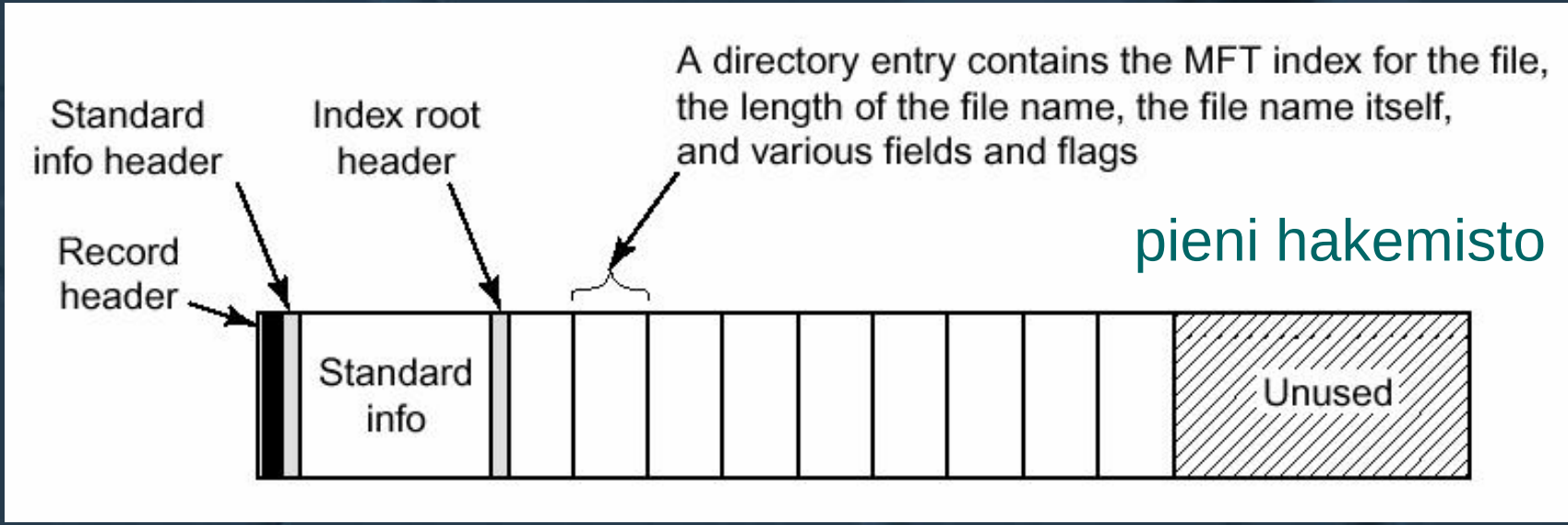
Fig. 11-37. A file that requires three MFT records to store all its runs.

[Tane 01]



[Tane 01] Fig. 11-34. The NTFS master file table.

Hakemiston MFT-tietue



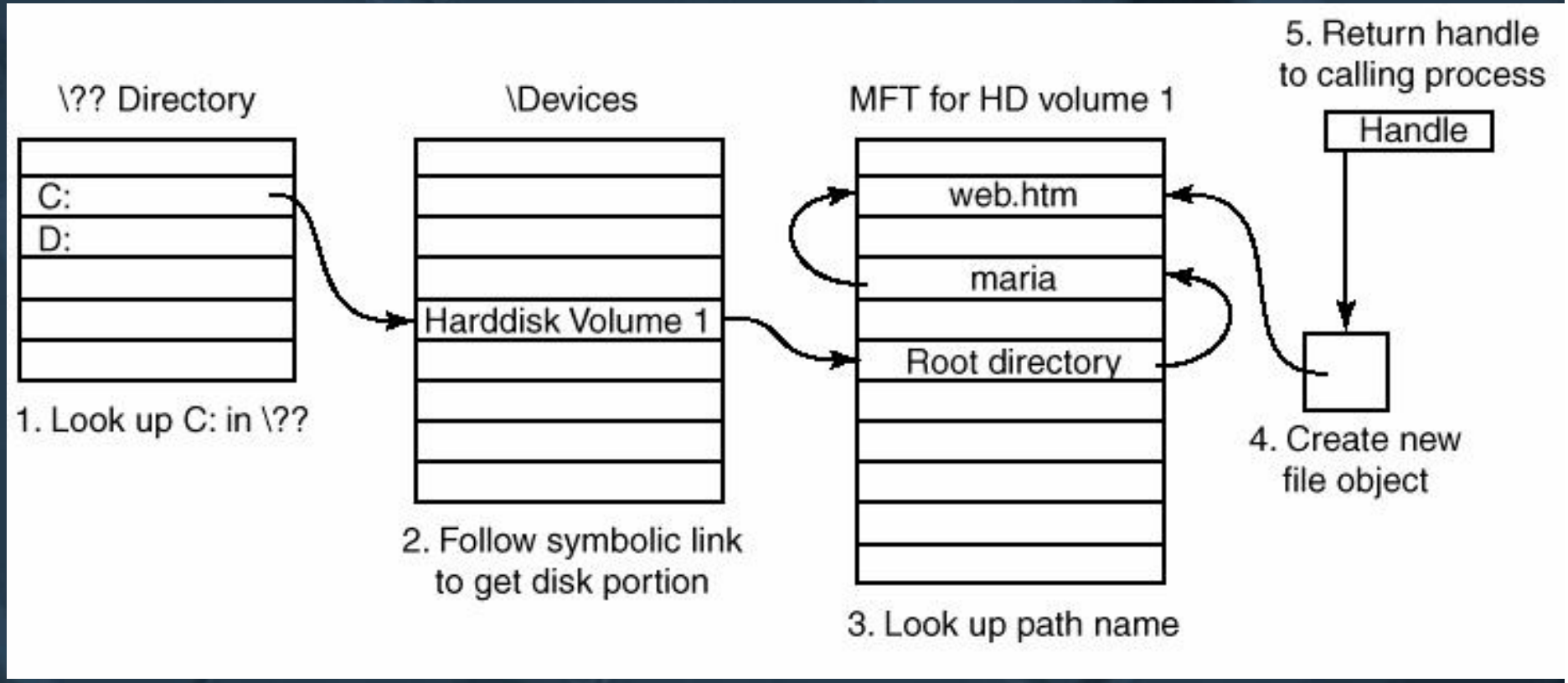
(Fig. 11-38 [Tane01])

- Pienissä hakemistoissa MFT-tietueet peräkkäisjärjestyksessä
- Isoissa hakemistoissa MFT-tietuessa B-puun (B-tree) indeksirakenne
  - nimen etsintä ei ole peräkkäishakua

Tiedoston käyttö

Fig. 11-39 [Tane 01]

- CreateFile(”C:\maria\web.htm”, ...)
  - etsi ensin oikea taltio
  - juurihakemiston tietue on MFT:ssä, etsi juurihakemistosta alihakemiston tietue
  - oliomanageri: luo uusi tiedosto-olio ja palauta kahva siihen
  - prosessi käyttää kahvaa seuraavissa kutsuissa



(Fig. 10-39 [Tane 01])

NTFS API

- **NTFS WIN32 API vs Unix API**

Fig 11-31 [Tane 01]

- **Tiedoston kopiointi Win32 APIn avulla**

Fig 11-32 [Tane 01]

Fig 6.5 [Tane 01]

Win32 API function	UNIX	Description
CreateFile	open	Create a file or open an existing file; return a handle
DeleteFile	unlink	Destroy an existing file
CloseHandle	close	Close a file
ReadFile	read	Read data from a file
WriteFile	write	Write data to a file
SetFilePointer	lseek	Set the file pointer to a specific place in the file
GetFileAttributes	stat	Return the file properties
LockFile	fcntl	Lock a region of the file to provide mutual exclusion
UnlockFile	fcntl	Unlock a previously locked region of the file

Fig. 11-31. The principal Win32 API functions for file I/O. The second column gives the nearest UNIX equivalent.

[Tane 01]

```
/* Open files for input and output. */
inhandle = CreateFile("data", GENERIC_READ, 0, NULL, OPEN_EXISTING, 0, NULL);
outhandle = CreateFile("newf", GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
   FILE_ATTRIBUTE_NORMAL, NULL);

/* Copy the file. */
do {
   s = ReadFile(inhandle, buffer, BUF_SIZE, &count, NULL);
   if (s && count > 0) WriteFile(outhandle, buffer, count, &ocnt, NULL);
} while (s > 0 && count > 0);

/* Close the files. */
CloseHandle(inhandle);
CloseHandle(outhandle);
```

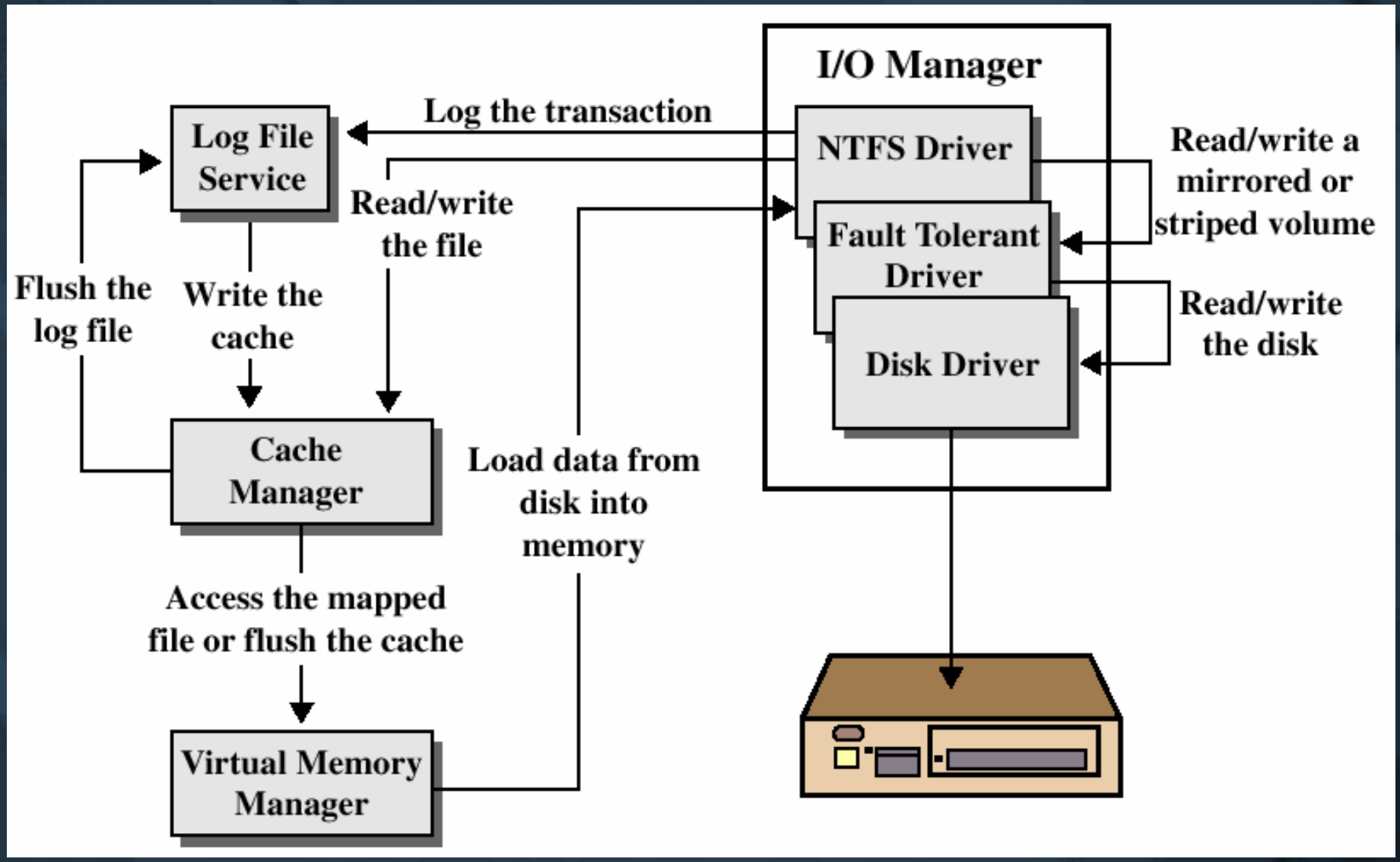
Fig. 11-32. A program fragment for copying a file using the Windows 2000 API functions.

[Tane 01]

NTFS: Virheistä toipuminen

Fig 12.18 [Stal 05]

- **Log FS**
  - kirjaa lokiin kaikki taltiota muuttavat transaktiot
  - loki aluksi välimuistissa (vain kirjanpidon rakenteista)
- **Muuta taltiota**
  i. talleta lokitapahtuma tiedostovälimuistiin (file cache)
  ii. tiedostomuutos välimuistiin
  iii. talleta lokitapahtuma levylle välimuistista } ”tapahtuma”
  iv. talleta muutokset levylle välimuistista } ”tapahtuma”
  v. kommitoidu (**commit**)
- Jos köllähtää ennenkuin muutokset levyllä, bootti voi palauttaa edeltävän tilanteen lokin avulla (**rollback**)
  - ei takaa etteikö tiedostojen tietoa katoaisi
  - järjestelmä säilyy eheänä (koherenttina)



(Fig 12.18 [Stal05])

NTFS virheistä toipuminen

- **Huono levysektori?**
  - kirjoittamassa? kirjoita muualle OK
  - lukemassa? *too bad*, data menetetty
  - lukemassa master boot recordia tai boot sector'ia
    - *really too bad*, taltio ehkä menetetty …
    - … ellei ehjää kopiota löydy
  - NTFS Log FS pitää tiedostojärjestelmän muuten eheänä
- **Yleinen lääke: Cluster Remapping**
  - sektori otetaan pois käytöstä ja kyseinen looginen sektori mapataan muualle levylle
    - voi hidastaa peräkkäiskäyttöä jatkossa

Käyttöjärjestelmät II

Linux ext3fs

Linux ext3fs

[Tweedie talk, 20.7.2000] click

(http://olstrans.sourceforge.net/release/OLS2000-ext3/OLS2000-ext3.html)

- **ext3fs = ext2fs with journaling**
  - journal in special file, or in special device
- **Problem: time spend in recovering file system after a crash**
  - fsck (e2fsck) takes too long – hours for big systems
  - must have better availability
- **Complete compatibility with ext2fs (ext2fs ↔ ext3fs)**
  - clean, unmounted ext3fs has no journal, can mount as ext2fs

TKTL: ext3fs kaikissa tiedostopalvelimissa

Linux ext3fs

- **Extra layer on top of ext2fs: JFS (journaling FS)**
  - independent on actual file system (ext2fs)
    - ext2fs does not know about journaling!
  - arbitrary modifications in buffer cache
  - transactional semantics
    - ”do all these 5 updates, or none of them”
  - API to add transactions onto a block device
- **File update**
  - no data to disk until transaction commit
    - no quarantee when it will be written (**write behind**)
    - written to disk from JFS cache of updates
  - write to log first, commit log, then do file update
    - disks can guarantee one sector write with power failure
    - use special sector updates as commit blocks for log

Kertauskysymyksiä

- Mihin tarvitaan VFS:ää?
- Kuinka ext2 poikkeaa ”iciwanhasta” UNIXin tdstojärjestelmästä?
- Mistä ext2:n tehokkuus / luotettavuus?
- Mitä tietoja superlohkossa/indeksisolmussa?
- Miksi i-solmuja? Miksi ei attribuutit ja nimi samassa paikassa?
- Mihin tarvitaan NFS-protokollaa?
- Mitä hyötyä on NFS:n tilattomuudesta?
- Mitä on ”journaling” ja ”logging”? Windows vs. Linux?