

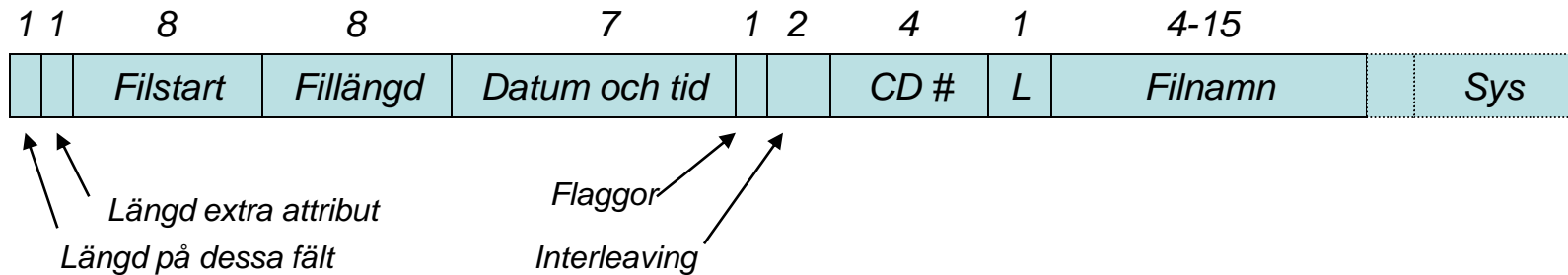
Operativsystem

Filsystem II
(kap 6 i boken)

Exempel på filsystem

- CD-ROM filsystem (ISO 9660 filsystem)
 - CD-ROM består av en linjär sekvens av logiska block på 2352 bytes, efter felkorrigering och annat 2048 bytes
 - ISO9660
 - 16 block ej definierade
 - 1 block – primary volume descriptor
 - root katalog
 - filer konsekutivt

ISO 9660 katalog



ISO9660 nivå:

- Level 1: 8+3 tecken
- Level 2: 31 tecken
- Level 3: Filer kan bestå av flere sektioner

Rock Ridge extensioner

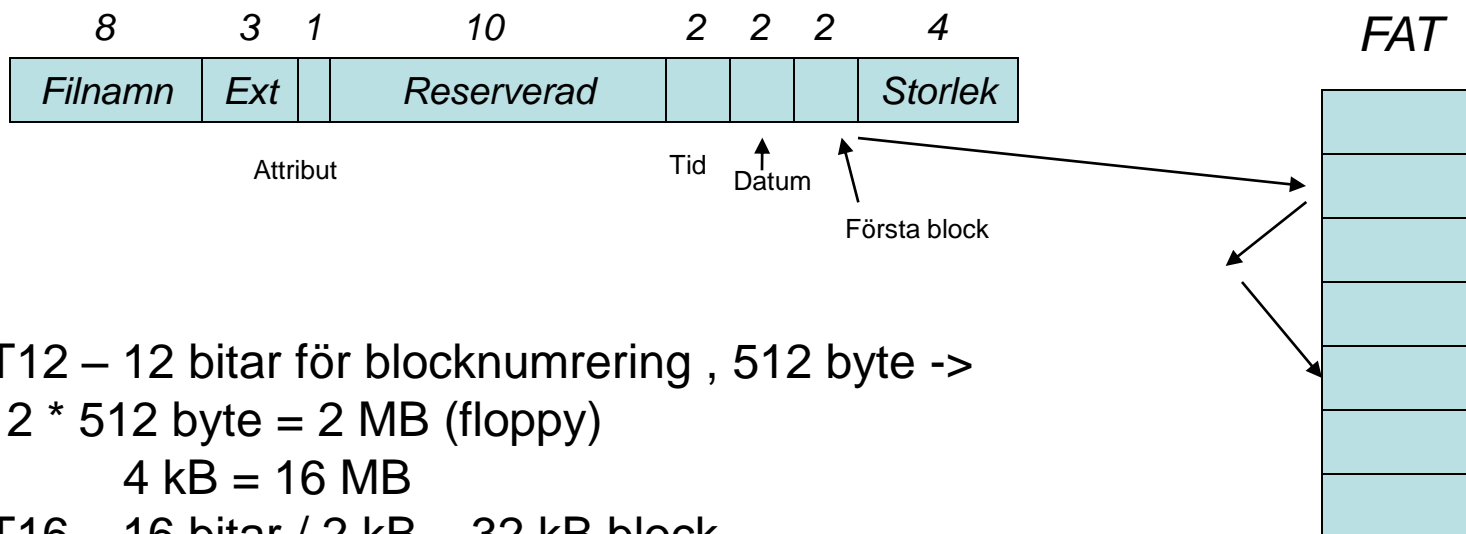
- Använder "sys"-fältet för följande "hjälp"-fält
 - PX – POSIX attribut (rwxrwxrwx)
 - PN – Major och minor enhetsnummer
 - SL – Symboliska länkar
 - NM – Alternativt namn
 - CL – Plats för underkatalog
 - PL – Plats för moderkatalog
 - RE – Förflyttning
 - TF - Tidsmärkning

Joliet extensioner

- Microsofts version av Rock Ridge:
 - Långa filnamn (64 tecken)
 - Unicode teckenuppsättning
 - Hierarkiska kataloger med djup större än 8 nivåer
 - Katalognamn med extensioner

MS-DOS

Katalog:



FAT12 – 12 bitar för blocknumrering , 512 byte ->

$2^{12} * 512 \text{ byte} = 2 \text{ MB (floppy)}$

4 kB = 16 MB

FAT16 – 16 bitar / 2 kB – 32 kB block

-> Max: $2^{16} * 2^{15} = 2^{31} = 2 \text{ GB}$

FAT32 – egentligen 28 bitar, 4 kb-32 kB block

-> Max: $2^{28} * 2^{15} = 2^{43} \text{ B} = 8 \text{ TB}$

(Alt $2^{32} * 2^{15} \text{ B} = 2^{47} = 128 \text{ TB}$)

Win 98 filsystem

- Såsom MS-DOS, men vissa förändringar i katalogstrukturen
 - långa filnamn
 - unicode filnamn
 - filnamn bakåt kompatibla, genom namnkonstruktion såsom
 - "long_file_name.txt" ->"long_f~1.txt"

Utveckling av filsystem

- Journaling filesystem
- Log-baserade filsystem
- Action control lists (ACL)
- Särskilja attribut från filkatalogen
- Atomära filsystem
- Flera strömmar per fil
- Filsystem för flash-minnen

Journaling filsystem

- Nackdel med många filsystem: Tar länge att restarta filsystem efter krasch (fschk())
- För att undvika, information om varje filaccess sparas först i en Journal (dvs. vad man tänker göra)
- Först efter detta uppdateras metadatan i filsystemet
- Vid fel måste man nu endast gå igenom journalen, inte hela filsystemet
- Oftast stöd för journal endast för Meta-data
- T.ex. Ext3 är Ext2-filsystemet utvidgat med journaling

Några filsystem med journal-stöd

A comparison of journaling file systems

| <i>Kernel support</i> | <i>Ext4</i> | <i>ReiserFS</i> | <i>XFS</i> | <i>JFS</i> |
|--------------------------------------|-------------|-----------------|------------|------------|
| Kernel prerequisites | No | No | Yes | No |
| In kernel.org source tree 2.4.lx | 2.4.15 | 2.4.1 | - | - |
| In kernel.org source tree 2.5.lx | 2.5.0 | 2.5.0 | - | 2.5.6 |
| License | GPL | GPL | GPL | GPL |
| <i>Features</i> | | | | |
| Largest block size supported on ia32 | 4 KB | 4 KB | 4 KB | 4 KB |
| File system size maximum | 1 EB | 16 TB | 8 EB | 4 PB |
| File size maximum | 16 TB | 1 EB | 16 EB | 32 PB |
| Growing the file system size | Patch | Yes | Yes | Yes |
| Access Control Lists | Patch | No | Yes | WIP |
| Dynamic disk inode allocation | No | Yes | Yes | Yes |
| Data logging | Yes | No | No | No |
| Place log on an external device | Yes | Yes | Yes | Yes |

Log-baserade filsystem

- De flesta skrivningar till filer är små (< 1 kb)
 - För varje skrivning måste data + meta-data uppdateras (ofta på olika ställen på skivan) → låg medelprestanda
- Samla istället en mängd små uppdateringar, skriv dessa i en följd till skivan (t.ex. 1 MB) → prestanda enligt vad hårdvara tillåter
- Läsning blir nu i princip att gå igenom loggen tills man hittar det man söker
 - Krävs system för att försnabba läsningen (t.ex. för random access läsning)
- **Exempel: JFFS - JFFS2**
 - Data skrivs i noder: inoder / dirent noder
 - Noder samlas i block
 - Noder är antingen giltiga eller föråldrade
 - Behövs en garbage collector, som konverterar block innehållande föråldrade noder till rena block

Action Control Lists

- MS-DOS: I praktiken inget skydd av filer
- Ext2/Ext3: rwxrwxrwx-typ skydd (dvs. rättigheter för ägare / grupp / övriga)
- Men typiskt: Du vill ge rätt åt någon viss person att ändra t.ex. en webbsida, hur då???
- Access Control Lists: Kan lägga till rättigheter på personbasis / gruppbasis för fil:
- minfil.txt:
 - jbjorkqv: rwx, aeklund: rx
- Stöd i t.ex.: NTFS, UFS, XFS, ext2*, ext3*, ReiserFS

Attribut skilda från själva filen

- Traditionellt: Fil är en namngiven räkka av bytes
- Olika applikationer behöver olika attribut för filer: t.ex. datummärkning, accesstider, rättigheter (ACL), samt alltmer metadata
- Tillägg av attribut har traditionellt betytt att själva filsystemet måste modifieras
- Istället vill man göra det möjligt att attribut kan läggas till filer efter behov
- Möjliggör också att en fil består av flere data”strömmar” (streams) (t.ex. NTFS)

Atomära filsystem

- Typisk exempel
 - 1. Tag ut 10€ från konto A
 - 2. Sätt in 10€ på konto B
- Om systemet kraschar mellan 1. och 2., är det bättre om också 1. makuleras
- Vanligt i databaser, men kraven på sådant i filsystem ökar också
- Journaler hjälper här till
- Implementationer idag brukar erbjuda atomär uppdatering av meta-datan i filsystemet, dock ej atomär data-uppdatering

Flera strömmar per fil

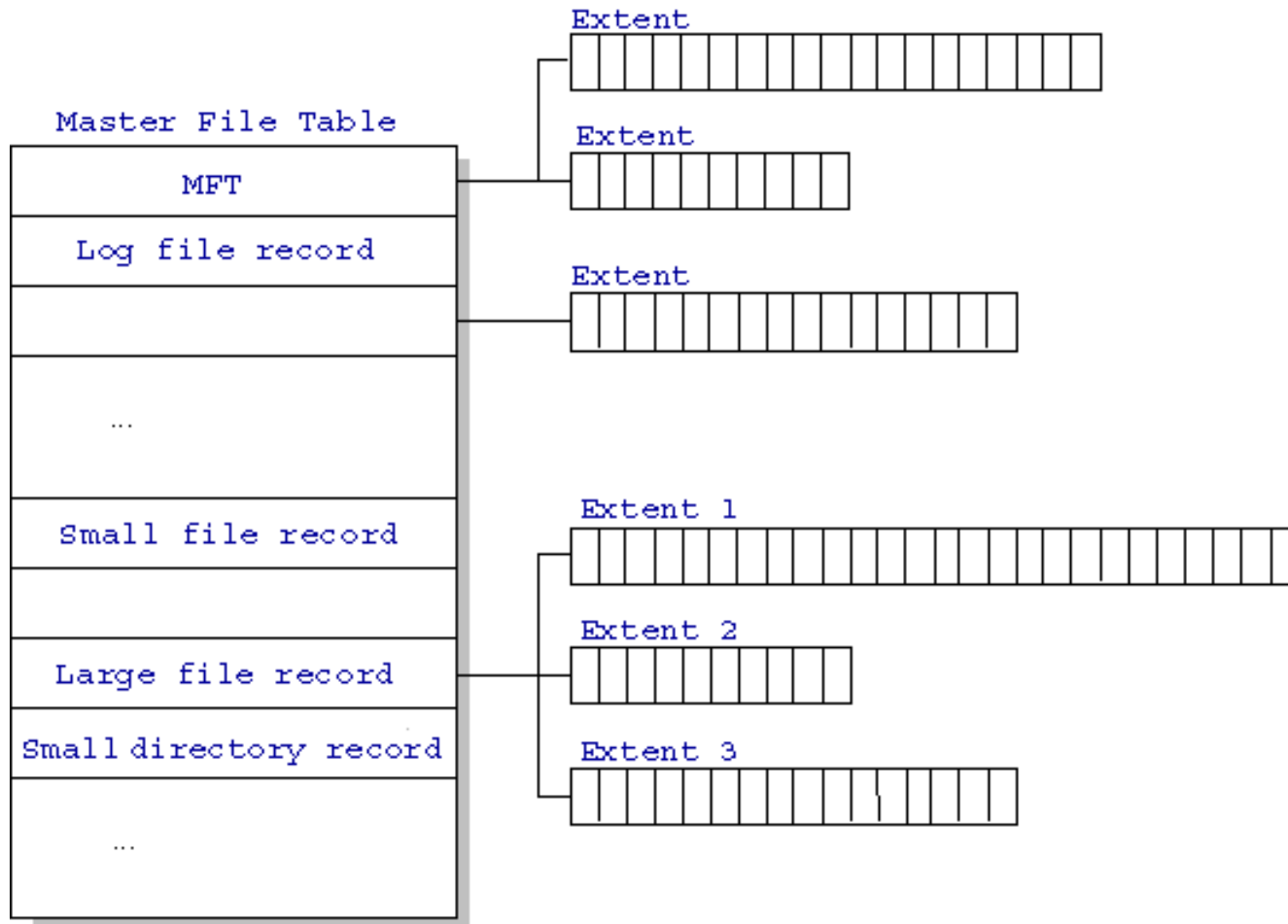
- En logisk fil kan ha flere fysiska representationer (strömmar), t.ex.
 - 1. MS Word-format
 - 2. Text-format
 - 3. HTML-format

NTFS

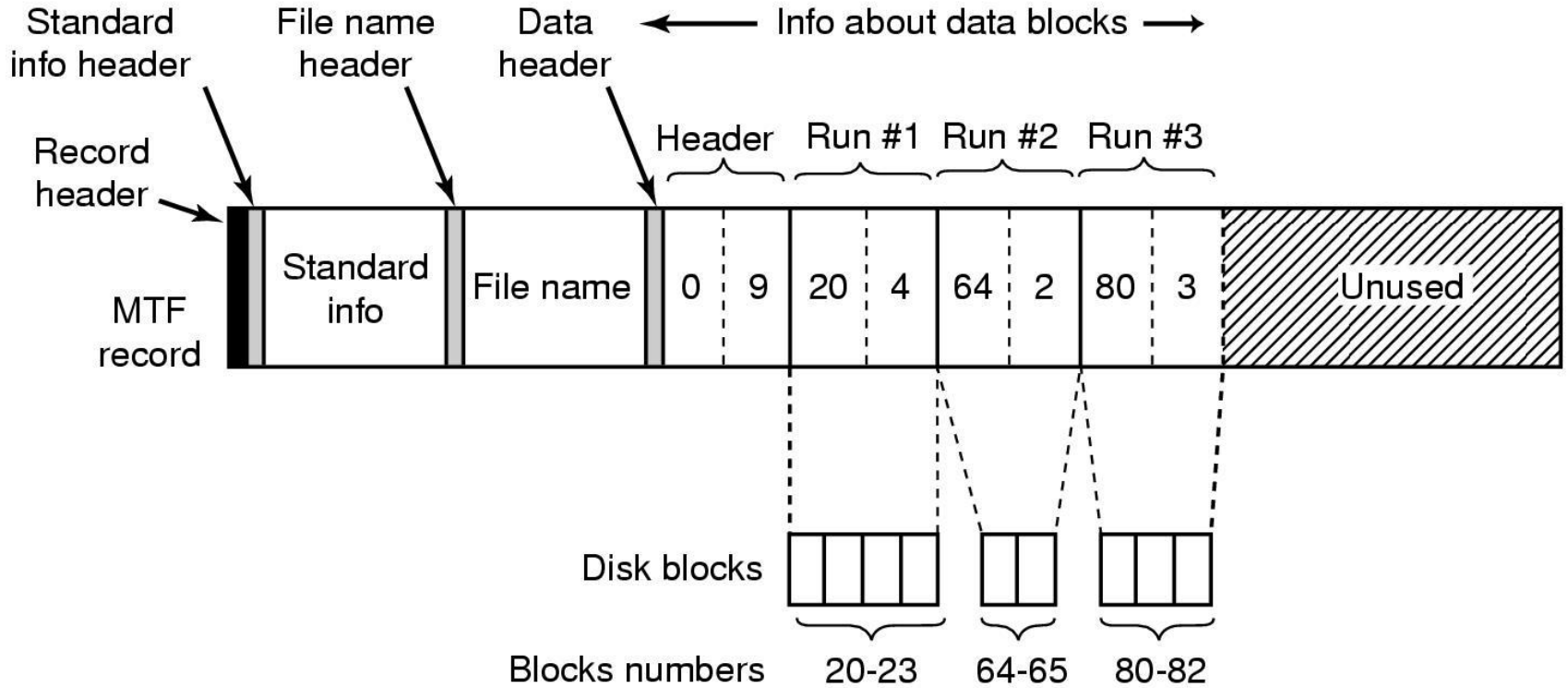
- Master File Table (MFT)
 - En räkka av 1 kB strukturer
 - All data i NTFS är i sig filer
 - Filsystemets meta-data är speciella filer:

| | |
|-----------|---|
| \$MFT | Itself MFT |
| \$MFTmirr | copy of the first 16 MFT records placed in the middle of the disk |
| \$LogFile | journaling support file (see below) |
| \$Volume | housekeeping information - volume label, file system version, etc. |
| \$AttrDef | list of standard files attributes on the volume |
| \$. | root directory |
| \$Bitmap | volume free space bitmap |
| \$Boot | boot sector (bootable partition) |
| \$Quota | file where the users rights on disk space usage are recorded (began to work only in NT5) |
| \$Upcase | File - the table of accordance between capital and small letters in files names on current volume. It is necessary because in NTFS file names are stored in Unicode that makes 65 thousand various characters and it is not easy to search for their large and small equivalents. |

NTFS



MFT File structure



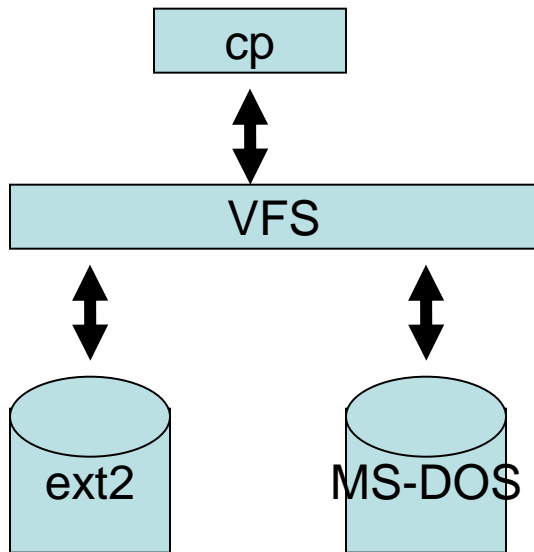
NTFS

- Symboliska länkar??
 - Ja, finns i NTFS, men inte tillgängligt direkt i Win32
 - Hjälpprogram "junction.exe" från www.sysinternals.com skapar sådan, t.ex.
 - junction "c:\Documents and Settings\jbjorkqv\My Documents" c:\docs
- Hard links??
 - Ja, finns i NTFS

Implementation av filsystem

- Hittills talat om hur filsystem logisk byggs upp
- Hur implementeras filsystem i ett operativsystem?
- Ofta används något typ av virtuellt filsystem (Virtual File System)
- Exempel: Linux VFS-arkitektur

Virtuellt filsystem



```
inf = open("/floppy/TEXT",  
           O_RDONLY, 0);  
outf=open("/tmp/test",  
          O_WRONLY|O_CREATE|O_TRUNC,  
          0600)  
do {  
    l=read(inf, buf, 4096);  
    write(outf, buf, l);  
} while (l);  
close(outf);  
close(inf);
```

VFS – Filsystemimplementation - rollfördelning

VFS

- Filabstraktion för all filsystem
- Handhar systemanrop från användarnivå (open, write, read...)
- Interaktion med filsystemet via traversering av mount-punkter
- Handhar även anrop från andra delar av kernelen

Filsystemet

- Administrerar fil & katalogdata
- Administrerar filsystemets meta-data (ägare, tidstämplar etc.)
- Överför data mellan
 - Specifikt FS
 - VFS data

Linux VFS

- Allmän filmodell – alla filsystem bör implementera följande objekt
 - Superblock
 - Motsvarar det fysiska superblocket på hårddisken (eller i fall av FAT en virtuell modell för superblocket)
 - I-node
 - Motsvarar den I-nod som finns på disken (eller en virtuell version av den)
 - Filobjekt
 - Finns i kernel-minnet så länge filen är öppen
 - Dentry
 - Information om hur ett stignamn mappas till en fil

Linux VFS

- Filsystem som inte har fysiska objekt såsom i-noder, så måste motsvarande implementation "simulera" dessa: procfs, fat, nätverk
- Filsystem registreras via kernel-funktionen `register_filesystem()`

Specialfilssystem

- I linux används specialfilssystem för att t.ex. Enkelt manipulera datastrukturer i kernelen
- Har inte en fysisk motsvarigheter
 - devfs - /dev – virtuella enhetsfiler
 - proc - /proc – accessering av datastrukturer i kernelen
 - sockfs – sockets – för IPC mellan processer / datorer

Exempel på specialfilssystem

```
[jbjorkqv@borken jbjorkqv]# more /proc/filesystems
nodev      rootfs
nodev      bdev
nodev      proc
nodev      sockfs
nodev      tmpfs
nodev      shm
nodev      pipefs
nodev      binfmt_misc
nodev      ext2
nodev      ramfs
nodev      iso9660
nodev      nfs
nodev      autofs
nodev      devpts
nodev      usbdevfs
nodev      usbfs
```

Linux /proc-filsystem

- Proc-filsystemet kan användas t.ex. för
 - Listning av statistisk information
 - Hitta information om hårdvara i maskinen
 - Modifiera parametrar under körning
 - Titta på och modifiera nätverksparametrar
 - Minnes- och prestandainformation
- Läsning / skrivning enligt normala accessrättigheter
 - Normal kan endast root-användaren göra förändringar
- Exempel: ändra max antal trådar tillåtna i systemet
 - `echo "2048" > /proc/kernel/threads-max`

Linux kernel /proc file system

- loadavg Average of system load for the last 1, 5 and 15 minutes
- uptime Time in seconds since boot-up and total time used by processes
- meminfo The number of total, used and free bytes of memory and swap area(s)
- kmsg Kernel messages that have yet to be read in by the kernel
- version Current rev of the kernel and/or distribution (read from linux_banner)
- cpuinfo Recognized processor parameters
- pci Current occupation of pci slots.
- self/ Information about processes currently accessing /proc
- net/ Descriptions about the network layer(s)
- scsi/ Contains files with information on individual scsi devices
- malloc Monitoring provisions for kmalloc and kfree operations
- kcore A core dump for the kernel (memory snapshot)
- modules Information regarding single loaded modules

Linux kernel /proc file system

- stat General Linux Statistics
- devices Information about kernel registered devices on the system
- interrupts Interrupt assignment information
- filesystems Existing filesystem implementations
- ksyms Symbols exported by the kernel
- dma Occupied DMA channels
- ioports Currently occupied IO ports
- smp Individual information about CPU's if SMP is enabled
- cmdline Command line parameters passed to the kernel at boot time
- sys/ Important kernel and network information
- mtab Currently mounted filesystems
- md Multiple device driver information (if enabled)
- rc Enhanced real time clock (if enabled)
- locks Currently locked files

Proc filesystem

```
more /proc/cpuinfo
processor      : 0
vendor_id    : GenuineIntel
cpu family   : 6
model        : 7
model name    : Pentium III
              (Katmai)
stepping     : 3
cpu MHz      : 551.262
cache size   : 512 KB
fdiv bug     : no
hlt bug      : no
f00f bug     : no
coma_bug     : no
fpu          : yes
fpu_exception : yes
cpu_id level : 2
wp           : yes
flags        : fpu vme de pse
              tsc msr pae mce cx8 apic sep
              mtrr pge mca cmov
pat pse36 mmx fxsr sse
bogomips     : 1101.00
```

```
$ more /proc/devices
```

```
Character devices:
```

```
1 mem
2 pty
3 tty
4 ttyS
5 cua
7 vcs
10 misc
14 sound
128 ptm
136 pts
162 raw
180 usb
226 drm
```

```
Block devices:
```

```
2 fd
3 ide0
9 md
22 ide1
```

struct flock

```
#define LOCK_SH          1      /* shared lock */
#define LOCK_EX          2      /* exclusive lock */
#define LOCK_NB          4      /* or'd with one of the above to prevent
    blocking */
#define LOCK_UN          8      /* remove lock */
#define LOCK_MAND       32      /* This is a mandatory flock */
#define LOCK_READ       64      /* ... Which allows concurrent read
    operations */
/
#define LOCK_WRITE      128     /* ... Which allows concurrent write
    operations */
#define LOCK_RW         192     /* ... Which allows concurrent read & write
    ops
*/
struct flock {
    short l_type;
    short l_whence;
    off_t l_start;
    off_t l_len;
    pid_t l_pid;
};
```


Linux file locking

- Mandatory locks can be enabled per file-system during `mount()` (`flag MS_MANDLOCK`)
- Leases:
 - If process tries to open file protected by a lease, it is blocked as usual
 - A signal is sent to the process owning the lease
 - It should update the file, and release the lock (lease)
 - If owner does not to this, the lease is automatically removed by the kernel

Networking

OSI model

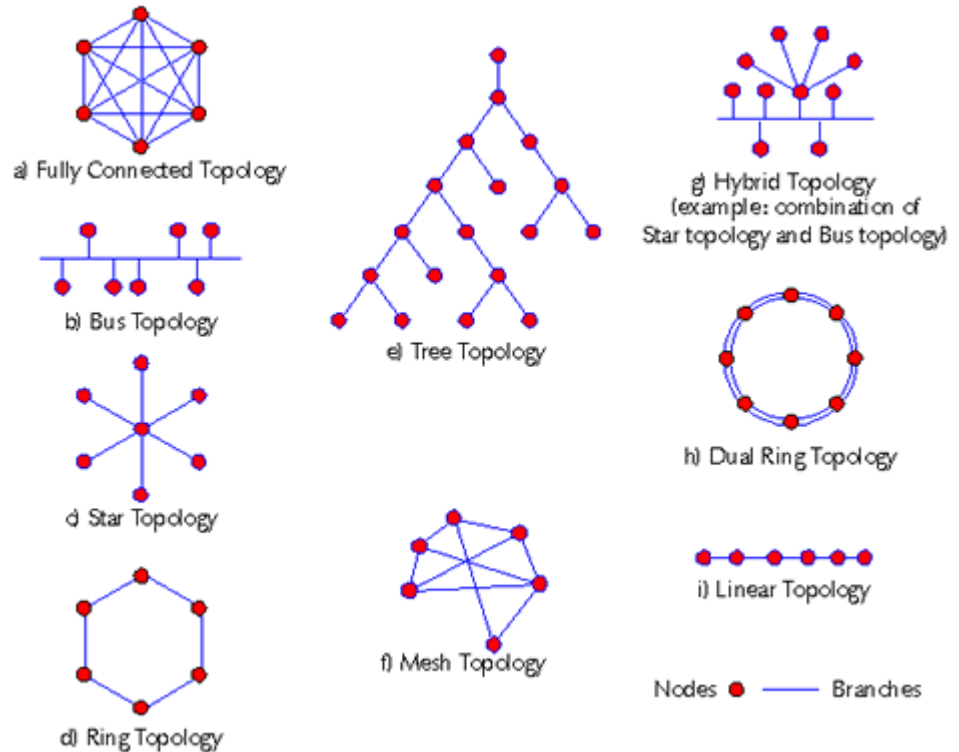
| | Osi model | TCP/IP | Microsoft |
|---|------------------|---|---|
| 7 | Application | Application programs like mail, web browsers etc. | |
| 6 | Presentation | Telnet, FTP, SMTP, HTTP | Server Message Block (SMB) |
| 5 | Session | | Network Basic Input / Output System (NetBIOS) |
| 4 | Transport | Transmission Control Protocol (TCP), Unacknowledged Datagram Protocol (UDP) | Network Basic Extended User Interface (NetBEUI) |
| 3 | Network | Internet protocol (IP) | |
| 2 | Data link | Network card interfaces (NIC) | |
| 1 | Physical | Transmission media: Twisted pair, coax, etc... | |

Network architectures

- The operation system may support several networking architectures
 - TCP/IP v4 / v6, Appletalk, Bluetooth, DecNet, Novell IPX, Unix Domain Sockets, X25
- In Linux, Network implementation referred as Net-4
 - Similar to VFS, common interface to a large number of architectures

Nätverkstopologier

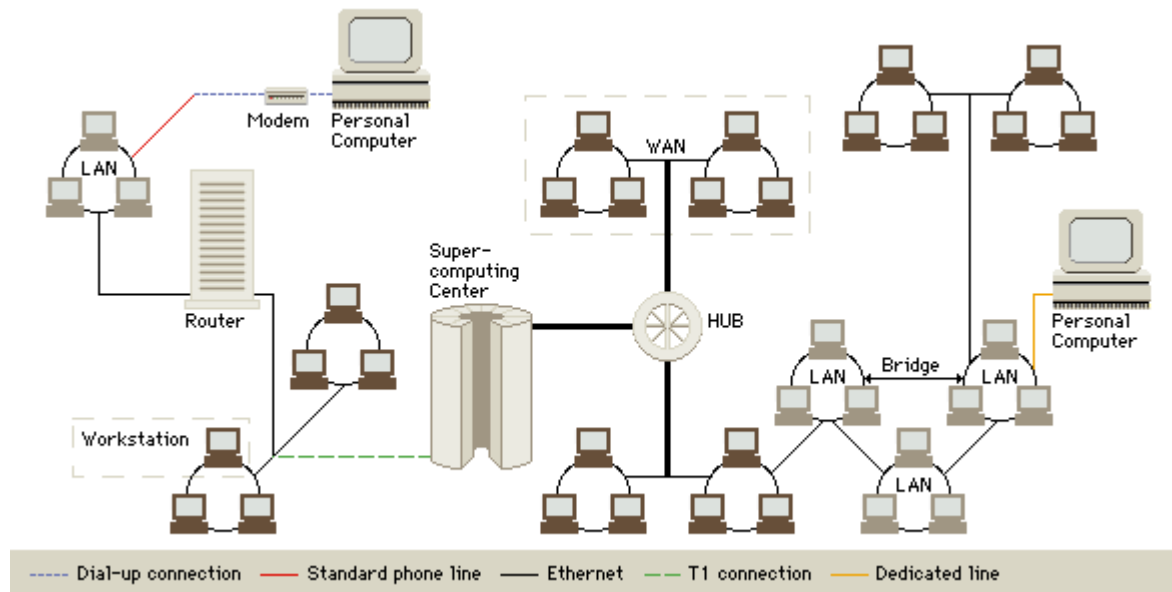
- Linjär
- Ring
- Mesh
- Fullt kopplad mesh
- Stjärna
- Träd



Källa: <http://www.networkdictionary.com/>

Internet

- Sammakoppling av lokalnät
 - Routers – ruttar mellan olika nät

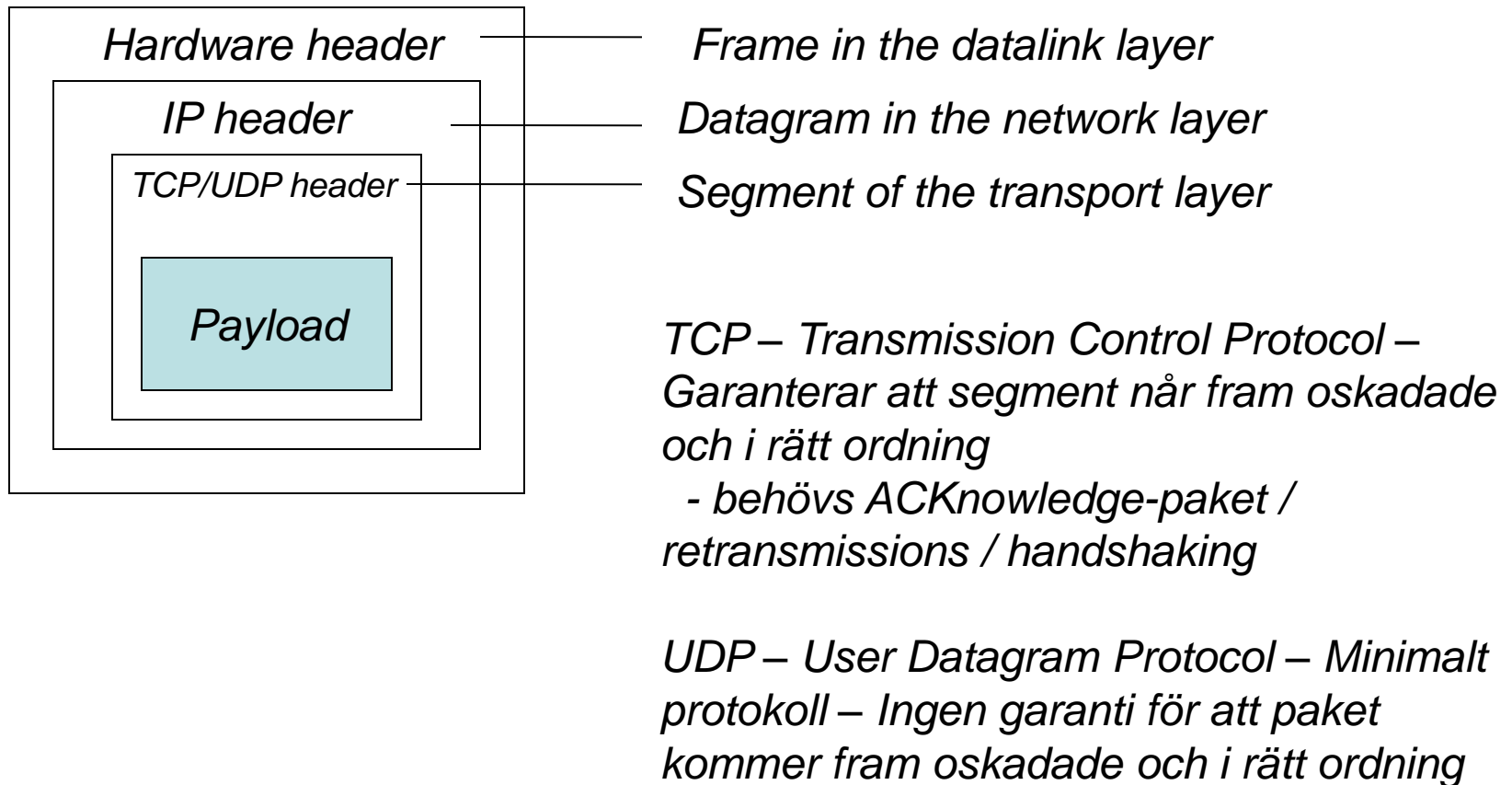


Källa: <http://images.encarta.msn.com/xrefmedia/aencmed/targets/illus/ilt/T300545A.gif>

Ethernet

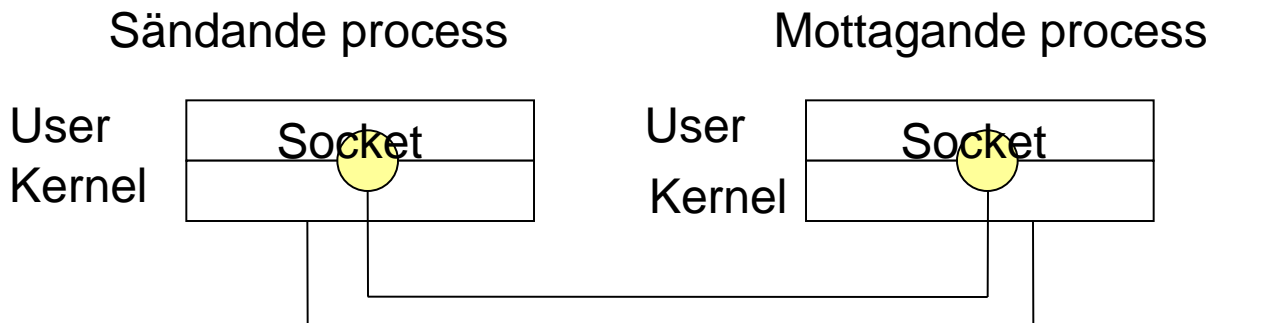
- IEEE802.3 – standard (CSMA / CD)
 - Kollisioner – kollisionsdetektionsmekanismer
 - Kabelformat
 - 10Base2 (thin Ethernet)
 - 10 MHz, tunn koaxialkabel, alla hör alla på samma kabel
 - 10Base5
 - 10 MHz, tjock koaxialkabel
 - 10BaseF
 - 10 MHz över fiber-optisk
 - 10BaseT
 - 10 MHz över tvinnad parkabel
 - 10Broad36
 - 10 MHz över bredbandskabel
 - 100BaseT, 1000BaseT etc...
 - 100/1000 MHz över tvinnad parkabel
 - Switched / not switched ethernet
- IEEE802.11 –standard (Wireless, CSMA / CA)
 - Request to send / Clear to Send

TCP/IP



BSD Sockets

- För att effektivt kunna använda nätverk, måste det finnas ett applikationsgränssnitt (API) mellan användarkod och den nätverkskod OS erbjuder
- BSD socket används ofta som bas för sådan API
- En socket
 - Är en kommunikationsändpunkt – förbindelsepunkt för kanal som förenar kommunikationen mellan två processer
 - Data som "skuffas" in i den ena förbindelsepunkten "kommer ut" i den andra
 - Nätverkskoden i OS kärnan ser till att kommunikationen äger rum

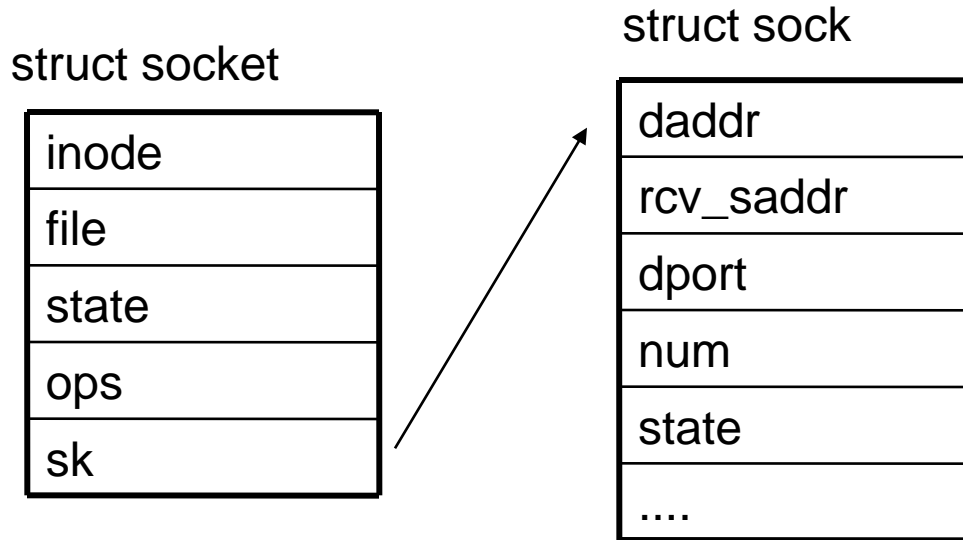


Implementation av *sockets* – *sockfs* filsystem

- I Linux implementeras *sockets* via ett speciellt filsystem, kallat *sockfs* filsystem
- Grunderna i detta: "The socket data structure"
 - *inode* – pointer to the inode object
 - *file* – pointer to the file object
 - *state* – connection status of the socket
 - *ops* – methods of the socket object
 - *sk* – low level struct sock descriptor

INET socket

- INET sockets (struct sock) hör till nätverksarkitekturen IPS: INET socket address in sk field



INET sockets - operationer (ops)

- close()
- connect()
- disconnect()
- accept()
- ioctl()
- init()
- destroy()
- shutdown()
- setsockopt()
- getsockopt()
- sendmsg()
- recmsg()
- bind()
- backlog_rcv()
- hash()
- unhash()
- get_port()

Exempel I – läsning från web-server

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

char msg[] = "GET / HTTP/1.0\n\n";
char retbuf[4096];

int main() {
    int sockfd;
    struct sockaddr_in raddr;
    int count;

    sockfd = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);

    raddr.sin_family = AF_INET;
    raddr.sin_port = htons(80);
    /* www.abo.fi = 130.232.213.59 */
    inet_pton(AF_INET, "130.232.213.59", &raddr.sin_addr);

    if (-1 == connect(sockfd, (struct sockaddr *)&raddr, sizeof (struct
sockaddr_in))) {
        perror();
        exit(-1);
    }

    write(sockfd, msg, strlen(msg)+1);
    count = read(sockfd, retbuf, 4096);
    retbuf[count] = 0;
    printf(retbuf);
}
```

Exempel II – en enkel server

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

char msg[] = "GET / HTTP/1.0\n\n";
char retbuf[4096];

int main() {
    int sockfd, c_sock;

    struct sockaddr_in saddr;
    struct sockaddr_in caddr;
    int count;

    sockfd = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);

    saddr.sin_family = AF_INET;
    saddr.sin_port = htons(9876);
    saddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    bind(sockfd, (struct sockaddr *)&saddr, sizeof(saddr));

    listen(sockfd, 5);
    while (1) {
        char ch;
        socklen_t clen = sizeof(caddr);
        printf("Server waiting\n");

        c_sock = accept(sockfd, (struct sockaddr *)&caddr,
&clen);
        printf("Connection from '%s'\n",
inet_ntoa(&caddr.sin_addr.s_addr));

        read(c_sock, &ch, 1);
        ch++;
        write(c_sock, &ch, 1);
        close(c_sock);
    }
}
```

Ruttning (routing)

- IP-lagret ser till att packets skickas framåt mot sin slutdestination
- IP-adresser (32-bit) indelas av
 - Nätverksadress
 - Värddadress
 - Nätverksmask (network mask) ger vilka bitar som anger nätverksadress
 - Resten av bitarna anger värddator på undernätet
- Ruttningstabeller är listor över de nätverk som värddatorn känner till
 - Om ett visst nätverk inte finns i ruttningstabellen, skickas paketet till en förgiven dator (*default gateway*)

Ruttning

- VP v.4
nätverksadresser i
format x.x.x.x (varje x
motsvarar en oktett)
- Destination AND
netmask →
destinationsnät
- Om destinationsnät
motsvarar nät i
ruttningstabellen, skicka
iväg på denna interface

```
C:\WINDOWS\system32\cmd.exe
er - Packet Scheduler Miniport
0x10004 ...00 0a e4 26 ec a2 ..... Intel(R) PRO/1000 MT Mobile Connection - Det
erministic Network Enhancer Miniport
=====
Active Routes:
Network Destination        Netmask          Gateway          Interface        Metric
0.0.0.0                    0.0.0.0          130.232.210.238  130.232.210.226  10
127.0.0.0                  255.0.0.0        127.0.0.1       127.0.0.1        1
130.232.210.224            255.255.255.240  130.232.210.226  130.232.210.226  10
130.232.210.226            255.255.255.255  127.0.0.1       127.0.0.1        10
130.232.255.255            255.255.255.255  130.232.210.226  130.232.210.226  10
224.0.0.0                 240.0.0.0        130.232.210.226  130.232.210.226  10
255.255.255.255           255.255.255.255  130.232.210.226  2                 1
255.255.255.255           255.255.255.255  130.232.210.226  130.232.210.226  1
Default Gateway:          130.232.210.238
=====
Persistent Routes:
None
C:\Documents and Settings\jbjorkqv>ping 192.168.0.226
Pinging 192.168.0.226 with 32 bytes of data:
Control-C
^C
C:\Documents and Settings\jbjorkqv>ping 130.232.210.226
Pinging 130.232.210.226 with 32 bytes of data:
Reply from 130.232.210.226: bytes=32 time<1ms TTL=128
Reply from 130.232.210.226: bytes=32 time<1ms TTL=128
Ping statistics for 130.232.210.226:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
Control-C
^C
C:\Documents and Settings\jbjorkqv>
```