Operativsystem

Filsystem I

(kap 6 i boken)

Filer

- Namngivning
  - Alla filer namnges, reglerna för namngivning är systemberoende
    - Kan vara case-sensitive (såsom i UNIX)
  - Kan vara två-delade filnamn (men numera mera konvention än regel)
- Struktur
  - Filer är ur OS synvinkel en räcka av tecken
- Filtyper
  - Reguljära filer eller katalogfiler
- Access och attribut
  - Filer kan accesseras sekventiellt eller valfritt ställe (random access)

Fil-operationer

- Create
- Open
- Close
- Read
- Write
- Delete
- Attributes get / set
- Rename

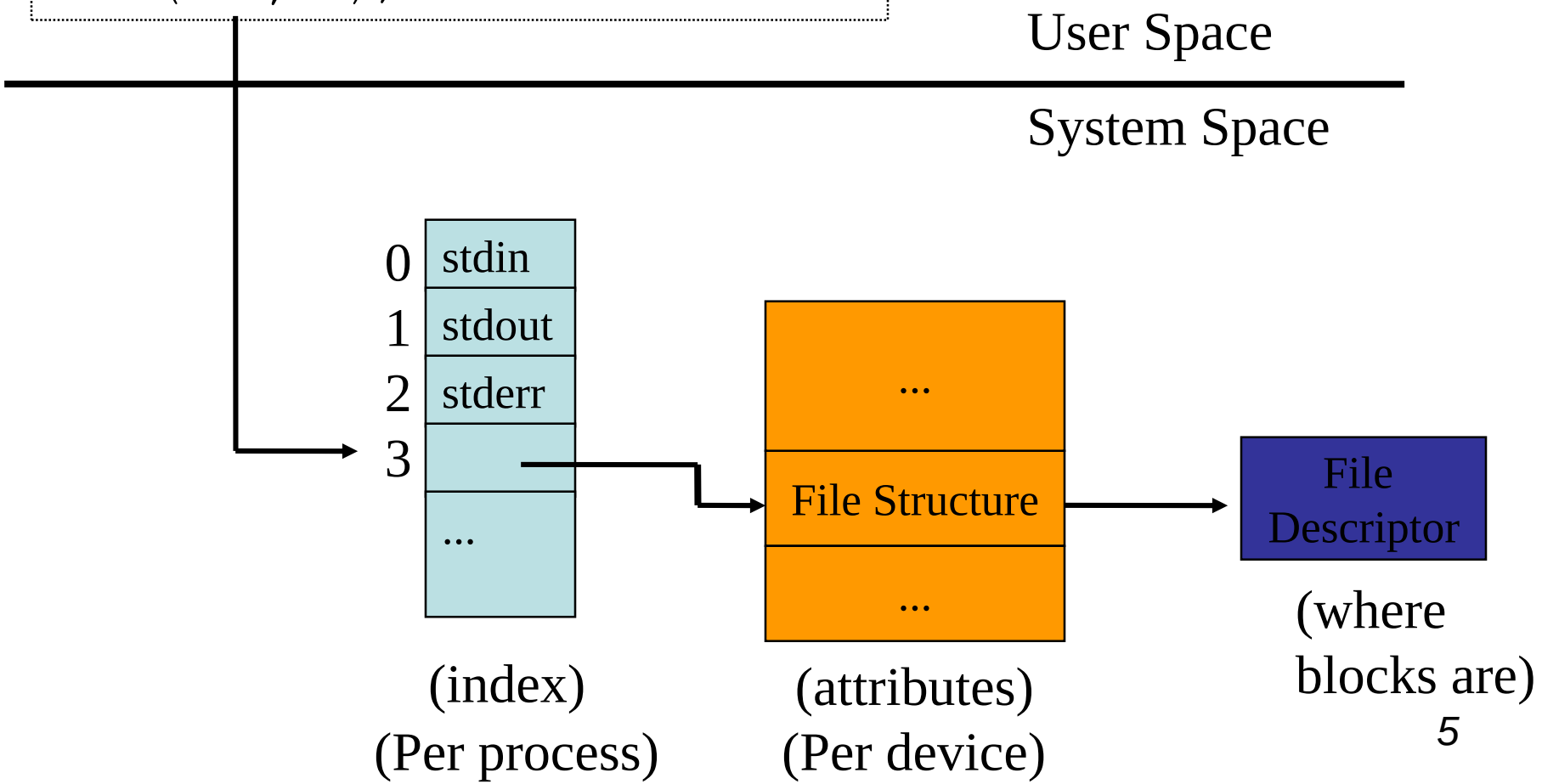
Exampel: Unix `open()`

```
int open(char *path, int flags [, int mode])
```

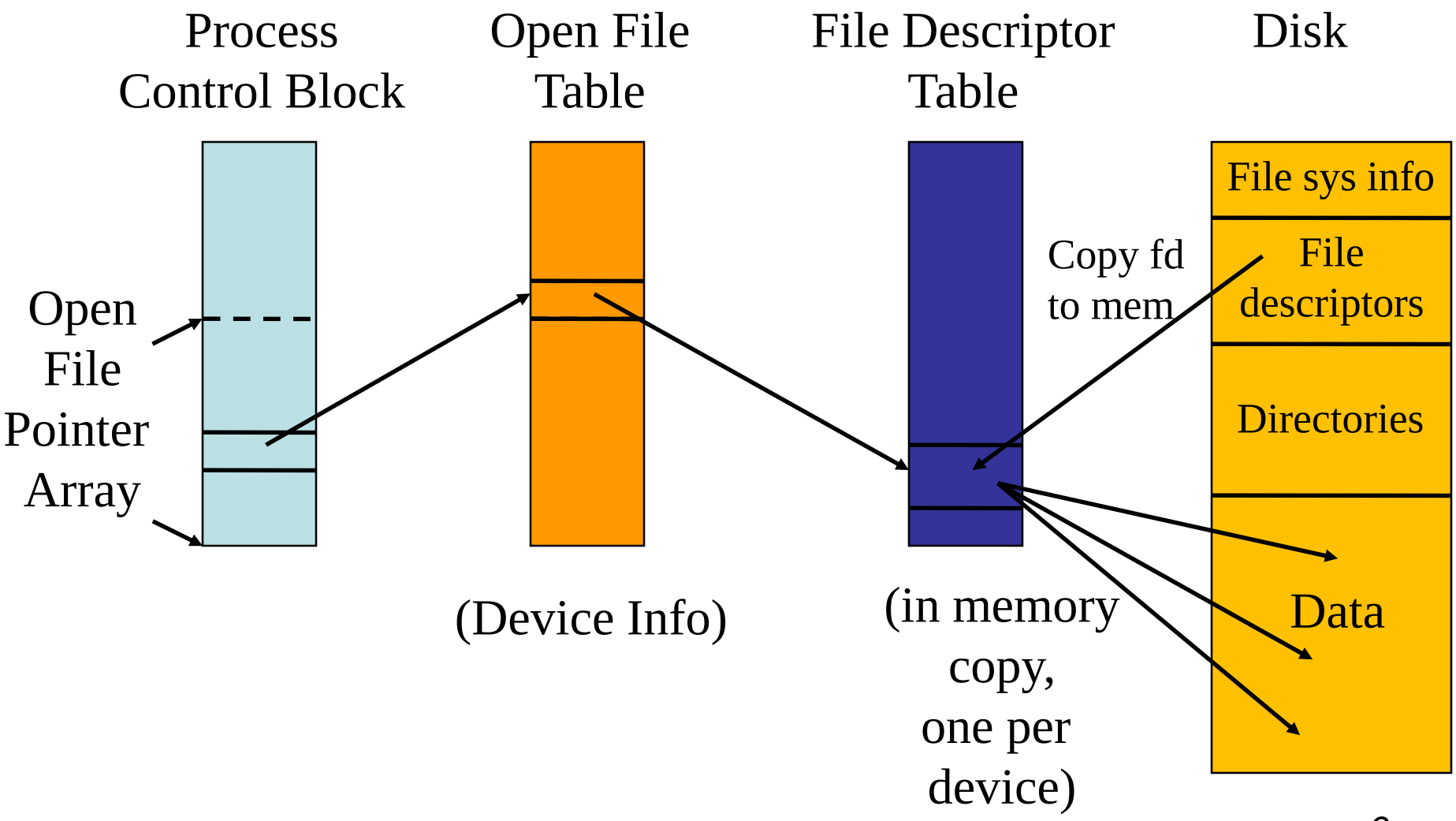
- `Path` är stigen till filen
- `flags` är bitmap med vilken sätts t.ex.
  - O_RDONLY, O_WRONLY…
  - O_CREATE sedan anv. `mode` för rättigheter
- Om, ok, returnera fildeskriptor

Unix `open()` – I närbild

```
int fid = open("blah", flags);
read(fid, …);
```



Implementation av filsystem



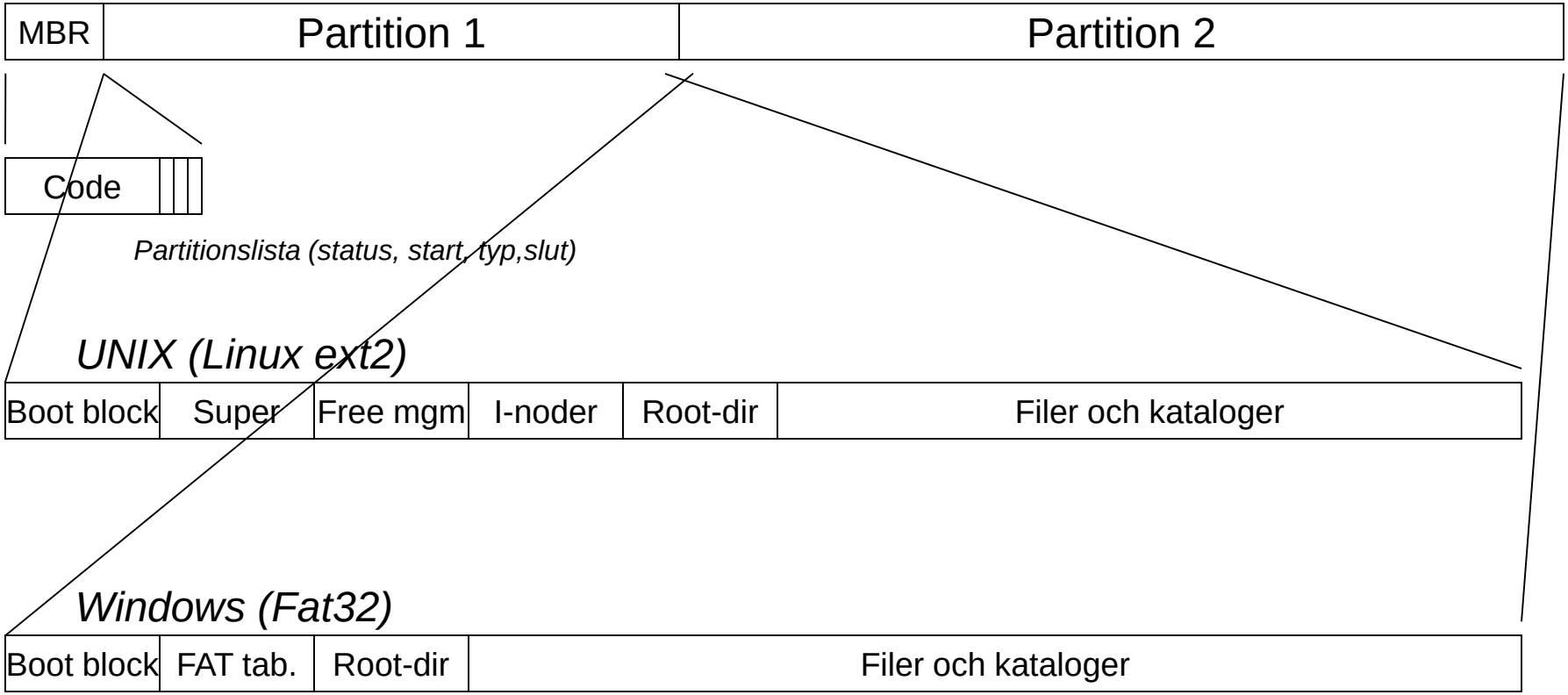
Minnesmappad filaccess

- Ibland vill man accessera filer på ett strukturerat sätt
- Minnesmappade filer: man ”mappar” in filen i det virtuella minnet så att filen kan användas som minne
- `void * mmap(start, length, prot, flags, fd, offset)`
- `munmap(start, length)`

Exempel

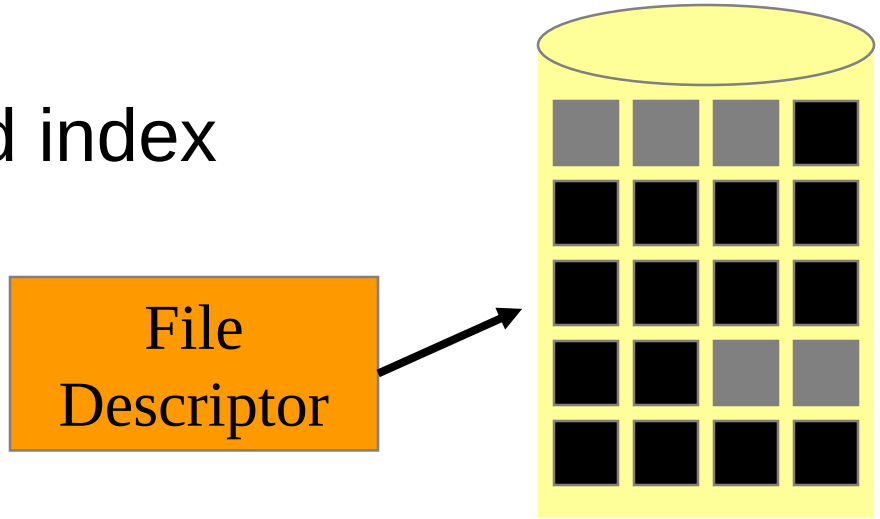
```
#include <unistd.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

main() {

  int fd;
  char buf[1024];
  char *p;


  fd = open("test", O_RDWR|O_CREAT, 0700);
  write(fd, buf, 1024);
  fsync(fd);

  p = (char *)mmap(0, 1024, PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0);
  printf("%p\n", p);
  if (p == MAP_FAILED) perror();
  strcpy(p, "Hello world");

  exit(0);
}
```

Hårdskivan totalt



Implementation av filsystem

- Vilka block hör till vilka filer?
- Implentationsprinciper:
  - Löpande
  - Länkade listor
  - Länkade listor med index
  - I-noder



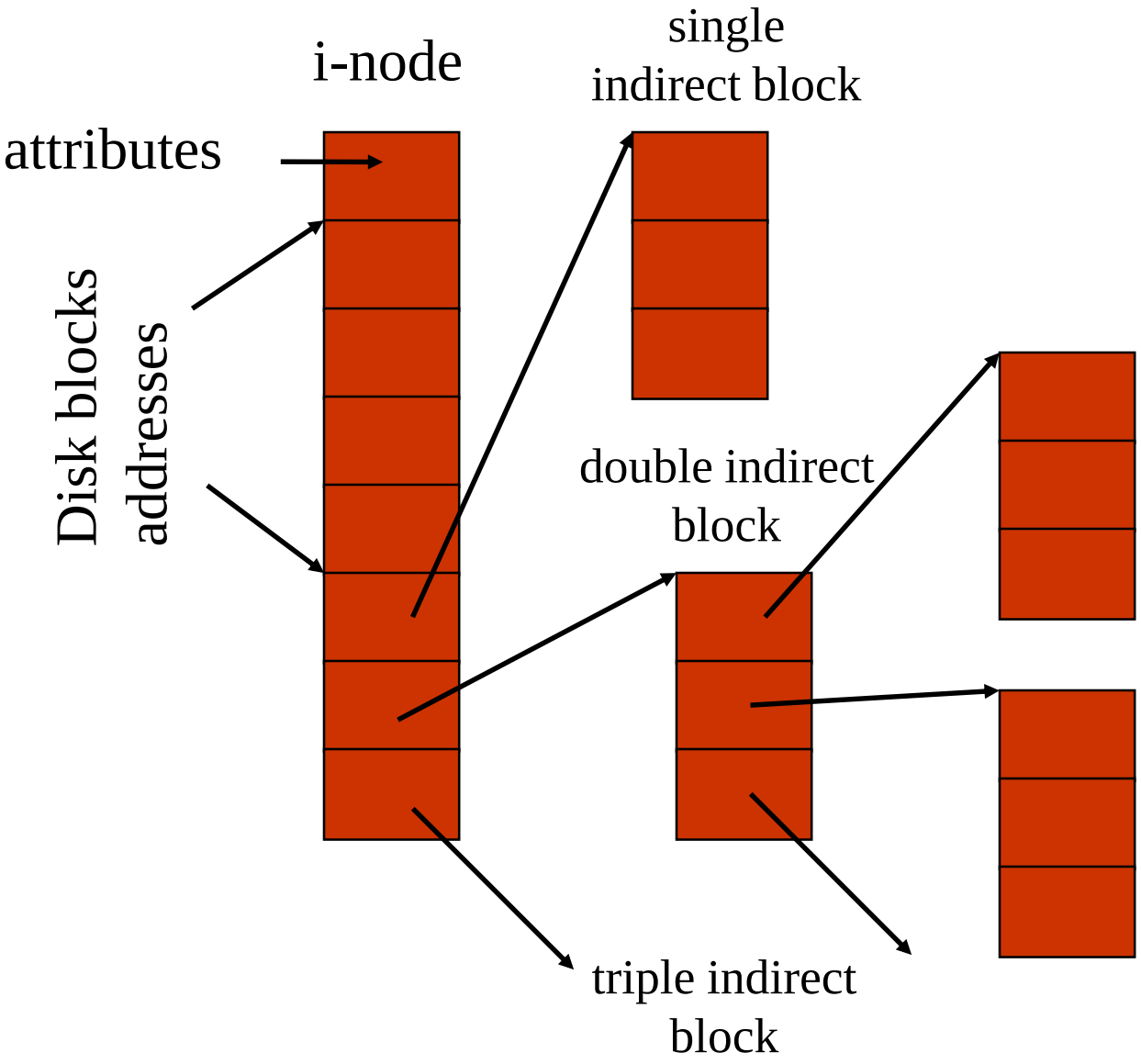
Implementation av filsystem

- Löpande allokering
  - Filer består av en mängd fortlöpande block
    - Blockstorlek normalt 512 B, 1 KB or 2KB
    - Snabb och enkel att implementera
    - Problem: Fragmentering
- Länkade listor
  - Första dataordet är pekare till nästa block
    - Ingen fragmentering (förutom intern…)
    - Random access är långsam (måste traversera listan, multipla seek-operationer på skivan)

Implemention av filsystem

- File allocation table (FAT)
  - En fil behandlas som en räcka av kluster. Indexeringen av dessa kluster sker via en filallokeringstabell, där varje “rad” i tabellen anger var nästa kluster befinner sig
    - FAT16: 16-bitars pekare (65536 kluster)
      - T.ex 20 GB disk - > 16 kB klusterstorlek
    - FAT32: 32-bitar pekare
- I-node
  - Varje fil associeras med en datastruktur kallad I-nod (index-node)
    - Attribut för filen
    - Block-address för själva datan

I-nodes



- Snabb om små filer
- Kan också accessera stora filer
- Storlek?
  - Ext2: 512 byte, 1kb, 2 kb, 4 kb

Linux ext2 I-node (on-disk)

```
/*
 * Structure of an inode on the disk
 */
struct ext2_inode {
        __u16   i_mode;         /* File mode */
        __u16   i_uid;          /* Low 16 bits of Owner Uid */
        __u32   i_size;         /* Size in bytes */
        __u32   i_atime;        /* Access time */
        __u32   i_ctime;        /* Creation time */
        __u32   i_mtime;        /* Modification time */
        __u32   i_dtime;        /* Deletion Time */
        __u16   i_gid;          /* Low 16 bits of Group Id */
        __u16   i_links_count;  /* Links count */
        __u32   i_blocks;       /* Blocks count */
        __u32   i_flags;        /* File flags */

        __u32   i_block[EXT2_N_BLOCKS];/* Pointers to blocks */
        __u32   i_generation;   /* File version (for NFS) */
        __u32   i_file_acl;     /* File ACL */
        __u32   i_dir_acl;      /* Directory ACL */
        __u32   i_faddr;        /* Fragment address */
...
}
```

WinNT NTFS

- Master File Table (MFT) – Sparar metadata om alla 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.

Kataloger

- För att organisera större mängder filer, brukar de delas in i kataloger
  - Katalogerna i själv är filer
- Normalt hierarkiska katalog-system

- Filer hittas via stignamn
  - t.ex. /usr/src/linux-2.4.20/README

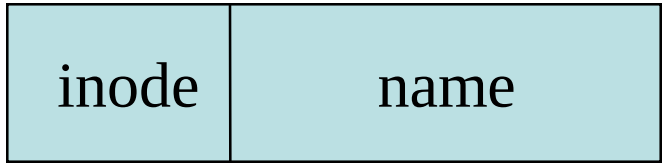
Kataloger

- Absoluta stignamn
  - `/home/jbjorkqv/os/fil1.txt`
- Relativa stignamn
  - relativt till current directory / hemkatalog
  - `os/fil1.txt`
- Speciella namn
  - ”.” – current directory
  - ”..” – katalog i nivå ovanför

Hierarkiska kataloger (Unix)

- Tree
- Rad:
  - namn
  - Inode-nummer(try “ls –l” or “ls –iad .”)
- exempel:

```
60   /usr/bob/mbox
```



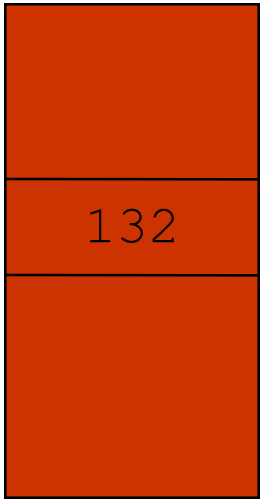
Unix: Exempel på kataloger

Root Directory

1	.
1	..
4	bin
7	dev
14	lib
9	etc
6	usr
8	tmp

Looking up usr gives I-node 6

I-node 6



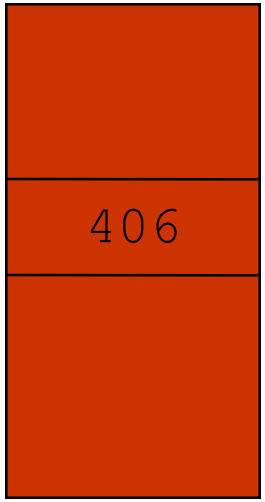
Relevant data (bob) is in block 132

Block 132

6	.
1	..
26	bob
17	jeff
14	sue
51	sam
29	mark

Looking up bob gives I-node 26

I-node 26



Data for /usr/bob is in block 406

Block 406

26	.
6	..
12	grants
81	books
60	mbox
17	Linux

Aha! I-node 60 has contents of mbox

Länkar till filer

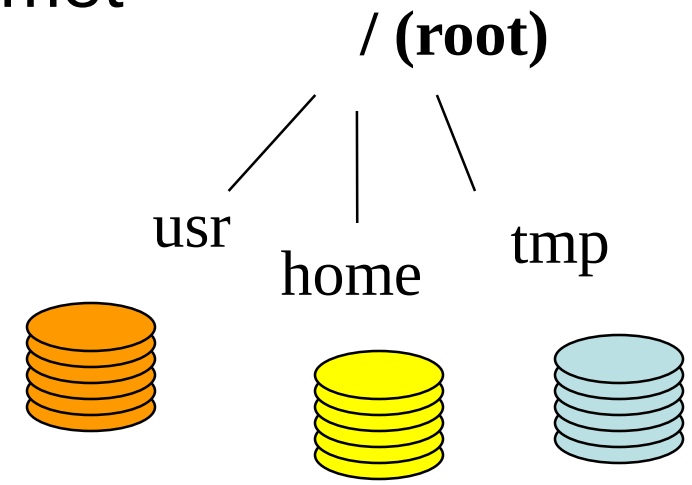
- Hard links – link()
  - Flere filer pekar på samma i-nod
  - Fungerar endast inom ett och samma filsystem
- Symbolisk länk – symlink()
  - Innehållet i en fil specificerar stigen till den egentliga filen

Administrering av diskutrymme

- Blockstorlek
  - Effektivitet mot utnyttjande av diskutrymme
- Listor med fria block
  - Länkade listor
  - Bitmap
- Disk quota
  - Begränsad kapacitet per användare

Partitioner

- `mount`, `unmount`
  - ladda “super-block” från skiva
  - välj “access point” i filsystemet
- Super-block
  - filsystemtyp
  - Storlek på block
  - Lediga block
  - Lediga I-noder



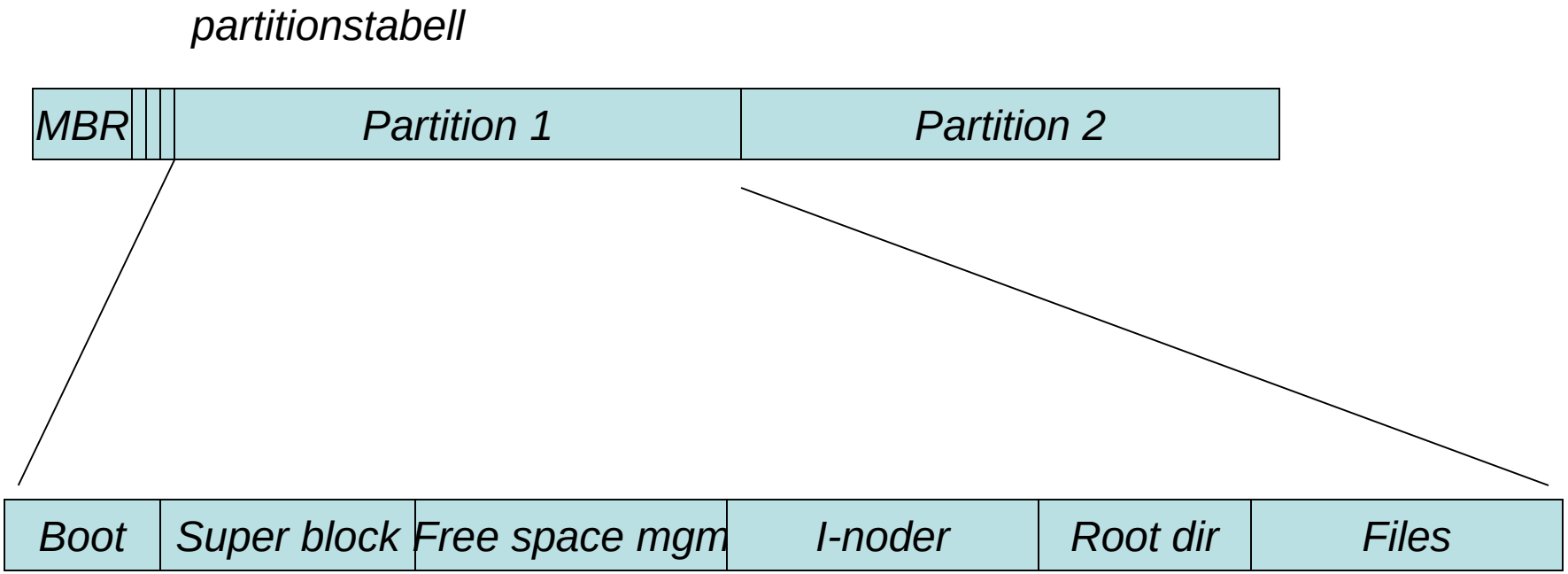
Partitioner: `fdisk`

- En partition är en större mäng sektorer allokerade för en
  - IDE-skivor begränsade till 4 “fysiska” partitioner
  - Logiska (extended) partitioner inuti de fysiska partitionerna
- Hur många cylindrar skall användas
- Specifiera typ
  - “magiska” nummer igenkända av OS

Underhåll av filsystem

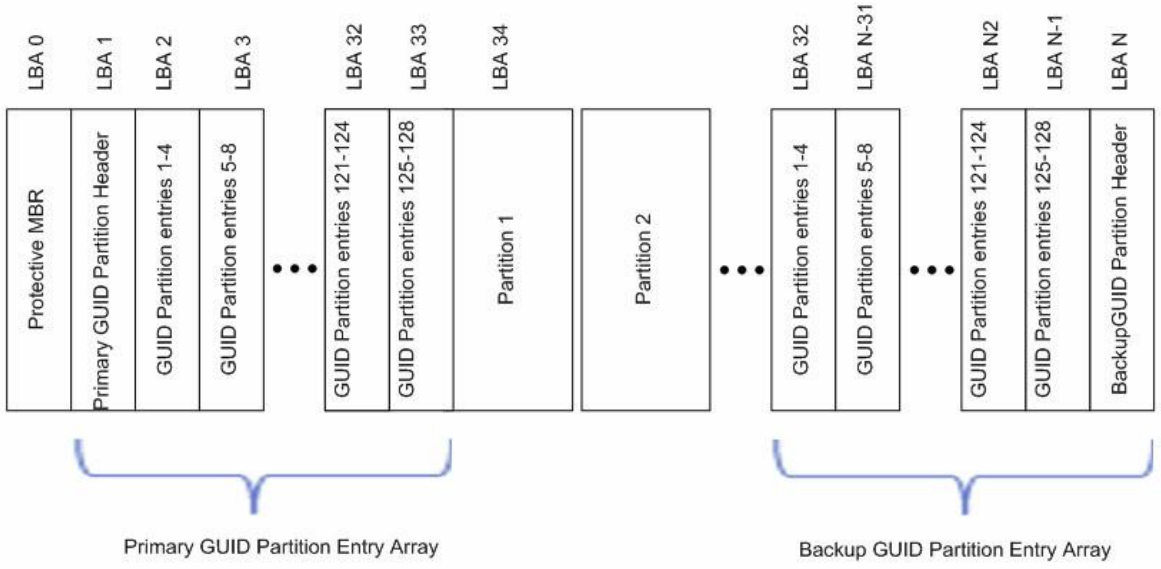
- Formatering:
  - Skapa datastrukturer för filsystem: super block, I-noder
  - `format` (Win), `mke2fs` (Linux)
- “Korrupta block”
  - Flesta hårdskivor har sådana
  - `scandisk` (Win) or `badblocks` (Linux)
  - Lägg till lista med “korrupta block” list (filsystemet kan ignorera)
- Defragmentera
  - Omorganisera block på skivan
- validera (om systemet har kraschat)
  - lost+found, korrigera fildeskriptorer...

Hårdskiva - layout



Partitioner GUID partitioner

- Sedan ca 2010 stöder de flesta OS partitionstabeller av typen Global Unique Identifiers (GUID)
  - Egentligen del av Unified Extensible Firmware Interface (UEFI)



Filsystem - pålitlighet

- Det att filsystemet förstörs är ofta värre än att själva datorn förstörs
- Backup
  - Recover from disaster
  - Recover from stupidity
- Filsystem – koherens
  - FAT, ext2, ext3, ReiserFS, XFS

Backup

- Fysisk dump
  - Startande från block 0
- Logisk dump
  - Rekursivt dumpar alla filer
- Inkremental dump
  - Dumpar de filer som uppdaterats sedan senaste backup

Filsystem - koherens

- Listan använda block motsvarar ej listan lediga block
  - Både ledigt och i användning
  - I ingedera listan
- Ett block finns allokerat till två filer
- Räkna länkar från filkataloger till i-noder
  - jfr med antal referenser i själva i-noden, stämmer detta?

Journaling filesystems

- Nackdel med många filsystem: Tar länge att restarta filsystem efter krasch (fschk())
- För att undvika, varje filaccess sparas först i en Journal
- Först efter detta uppdateras metadatan i filsystemet
- Vid fel måste man nu endast gå igenom journalen, inte hela filsystemet