Filsystem i Unix

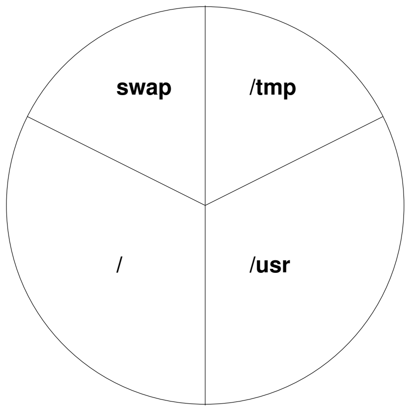
Filsystem i Unix består av två lager; det så kallade *flata filsystemet* med *i-noder*, där i-noderna ger varje fil ett unikt ID, och det så kallade *hierarkiska filsystemet*, där varje fil identifieras av en unik sökväg.

Det hierarkiska filsystemet är byggt ovanpå det flata filsystemet. När man anger att man vill öppna en fil, i t.ex. `open`, så översätter open (eller någon funktion som `open` anropar) filens namn till ett i-nods-nummer.

I katalogen `/usr/include/sys/fs` på SGI-maskiner finns include-filer som beskriver filsystemet.

Men vi börjar från början, men själva hårddisken:

En hårddisk med 4 partitioner:



Varje partition innehåller ett filsystem.

En hårddisk kan delas upp i flera delar, s.k. *partitioner*.

Dessa partitioner kan vara av olika storlek.

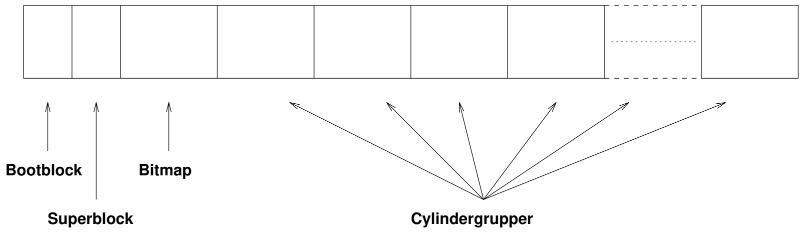
Används till olika saker; swap-partition, root-partition, tmp-partition etc..

Varje partition innehåller ett *filsystem*.

Finns många typer av filsystem: ext2fs (Linux), fat (dos), ntfs (NT), hpfs (OS/2), och efs (Irix) bl.a..

Vi beskriver här efs.

Ett filsystem:



Bootblocket: Innehåller information som används om datorn bootar från detta filsystem

Superblocket: Innehåller information om storlek, placering etc.

Bitmappen: Används för att hålla reda på lediga / upptagna block i filsystemet

Cylindergrupperna: Innehåller data

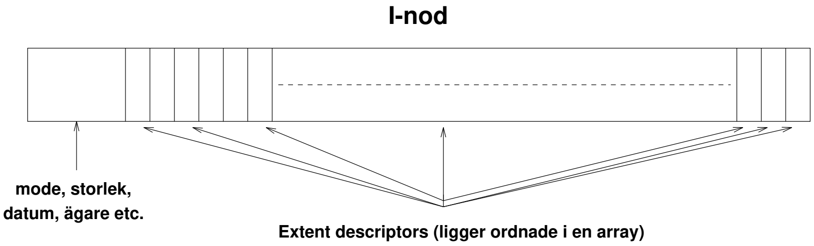
Efs består av flera delar:
Boot-block, som systemet bootar från,
Superblock, som innehåller info om filsystemets storlek, placering på hårddisken, vart eventuella kopior av superblocket finns etc.,
Bitmap, som innehåller en bit för varje block. Är den biten satt, är det blocket ledigt,
Cylindergrupper, som innehåller själva datan

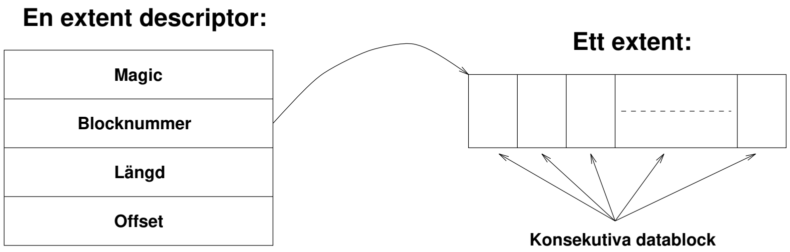
Cylindergrupp

I-nods-block: Innehåller en mängd i-noder

Datablock: Innehåller data från en fil eller ett bibliotek

Cylindergrupper består av två delar:
En del med block innehållande *i-noder*,
En del med block innehållande data ur en fil eller ett bibliotek





I-noderna innehåller information om filer eller bibliotek.

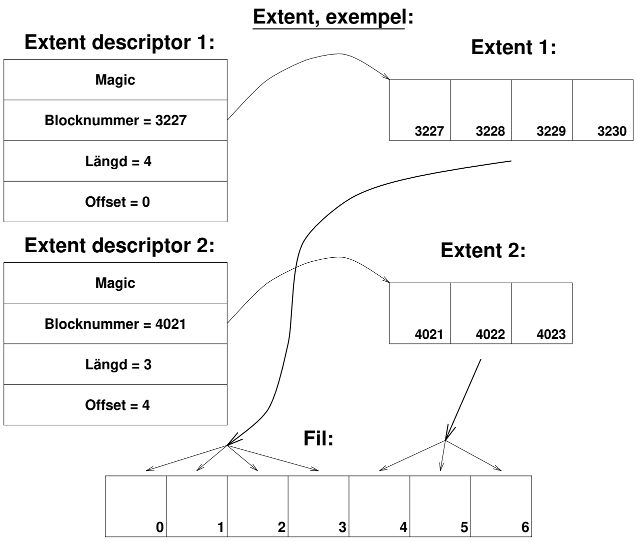
Börjar med en avdelning innehållande info om datum, mode, ägare, storlek etc..

Reasten av i-noden innehåller s.k. *extent descriptors*.

En extent descriptor innehåller 4 fält:
Magic, som är `0xbeef`,
Blocknummer, som anger vilket block datan börjar på,
Längd, som anger hur många block som "sitter ihop",
Offset, som anger vilket block i filen som första blocket motsvarar

Den följd av block som Blocknummer pekar ut, kallas *extent*.

Exempel:



En fil består av 7 block. Den är uppdelad i två delar på filsystemet.

Detta kan bero på att filen växt efter att den ursprungligen sparades.

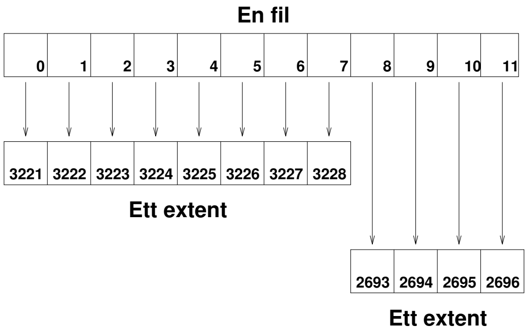
Extent descriptor 1 pekar ut de fyra första blocken i filen.

Offset 0 anger att det är blocken i början av filen som pekas ut.

Extent descriptor 2 pekar ut de följande 3 blocken i filen.

Offset 4 anger att det är filens block nummer 5 som pekas ut.

Detta är ett exempel hur en fil kan se ut fysiskt på hårddisken:



Observera att för program ser filen ut att bestå av en enda stor enhet.

Block 0-7 i filen ligger i ett extent, och block 8-11 i ett annat.

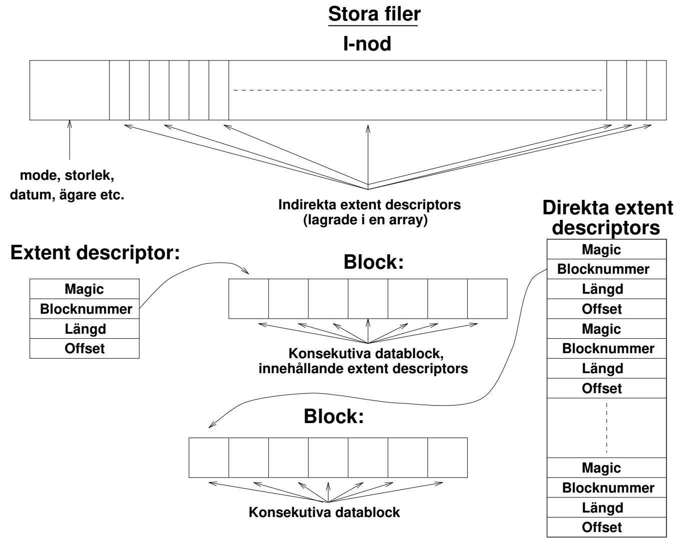
Numren som står i blocken i filen är filens logiska blocknummer.

Numren som står i extentens block är de block i filsystemet som motsvarar de logiska blocken.

På detta sätt kan man lagra filer som är 1,523,712 bytes stora. Varför?

Jo, en i-nod kan max innehålla 12 st. extent descriptors, och ett extent kan maximalt peka ut 248 block. Dessa block innehåller i sin tur 512 bytes. 12*248*512=1,523,712 bytes. (Siffror tagna ur `sys/fs/efs_ino.h` och `sys/param.h`).

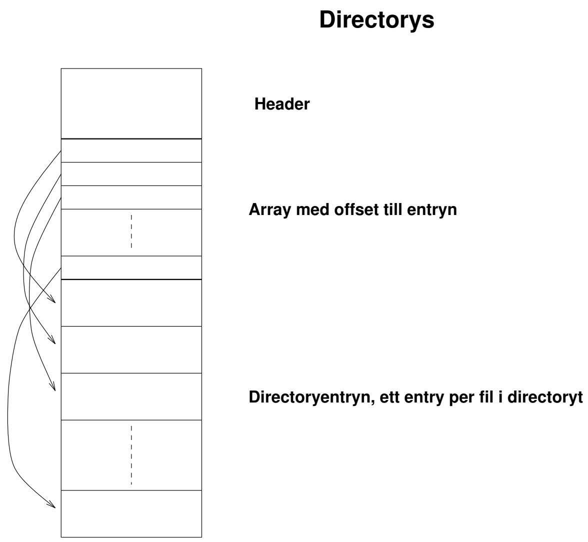
Vad gör man om man vill lagra större filer? Jo:



Man låter helt enkelt varje extent descriptor peka ut ett block med extent descriptors. Detta block innehåller extent descriptors som ser ut som de i en vanlig fil.

Man skiljer i detta fall på indirekta extent descriptors (de som finns i i-noden), och direkta extent descriptors, som finns i de block som de indirekta extent descriptorerna pekar ut.

Med denna metod kan man ha filer som är 4,160,622,592 bytes stora. Denna siffra fås genom att multiplicera det maximala antalet extents en fil kan bestå av (32,767) med max antal block ett extent kan peka ut (248) och antal bytes per block (512).



Directorys då? Jo, de lagras som filer. Deras storlek beror på hur många entrys (filer, länkar, underkataloger) som finns i dem.

Dock är det inte direkt fildata som lagras i directory-filerna, utan info om vilka entryn som finns i directoryt.