Sistemes de fitxers

Sistemes Operatius 2

Grau d'Enginyeria Informàtica

Introducció

L'objectiu d'aquest tema se centra en veure l'evolució històrica dels sistemes de fitxers:

- El problema d'emmagatzemar, recuperar i manipular informació és un problema molt general i hi ha moltes possibles solucions.
- No hi ha una forma "correcta" d'implementar un sistema d'arxius. Tot depèn del tipus suport d'emmagatzematge de què disposem així com el tipus aplicacions a les quals està dirigit.

Introducció

Com a usuaris

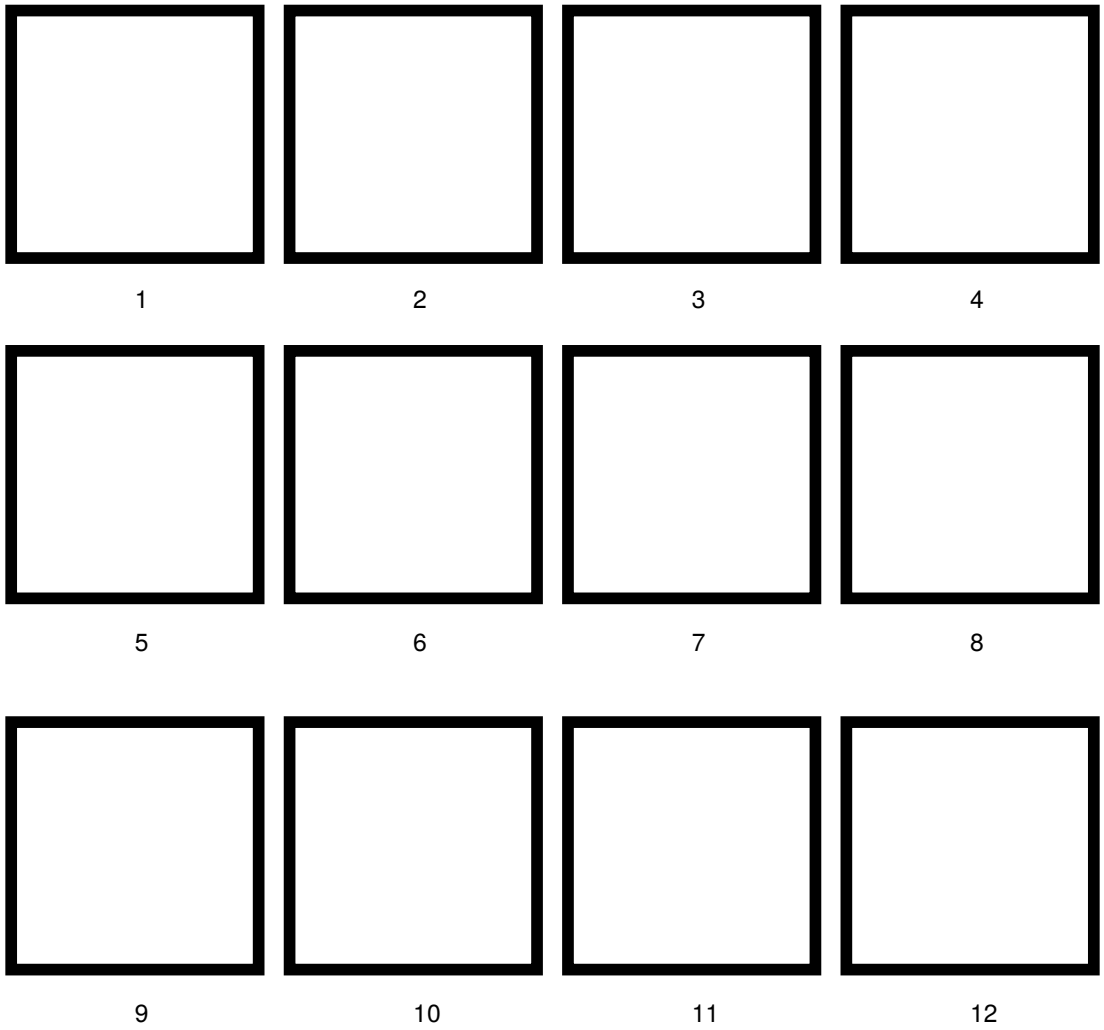
- Estem acostumats a treballar amb fitxers i directoris. La informació dels fitxers i directoris s'emmagatzema habitualment a un disc.
- El sistema de fitxers s'encarrega de gestionar de forma transparent tota aquesta informació per a nosaltres.

Des del punt del vista del sistema de fitxers (gestionat pel sistema operatiu)

- Com s'emmagatzema la informació associada als fitxers i directoris?
- Com es gestiona l'espai lliure de disc?
- Com es pot treballar de forma eficient i segura amb el disc (en cas que faci falta)?

Introducció

Dissenyem el nostre sistema de fitxers! Cada quadrat és un bloc de 4096 bytes... quines solucions podeu aportar?



Introducció

El suport d'emmagatzematge ha evolucionat molt al llarg dels darrers anys

- El diskette (1970–1999, màxim 200MB)
- El disc òptic (DVD, CD-ROM)
- El disc dur magnètic (els habituals a un ordinador)
- El disc d'estat sòlid (SSD, USB, ...)
- La cinta magnètica (utilitzats per fer *backups*).
- Els discos distribuïts en múltiples ordinadors en una xarxa

Cada suport pot necessitar un tipus específic de sistemes de fitxers. En aquestes transparències ens centrarem en sistemes d'ús general.

Concepte de sector i bloc

Un concepte important en un sistema d'arxius és la mida de bloc:

- Bloc o sector del disc: mínima unitat que el disc pot llegir o escriure (típicament 512 bytes).
- Bloc del sistema d'arxius: és la mínima unitat que el sistema d'arxius (del sistema operatiu) llegeix o escriu. Tot el que fa el sistema d'arxius està compost d'operacions sobre blocs.

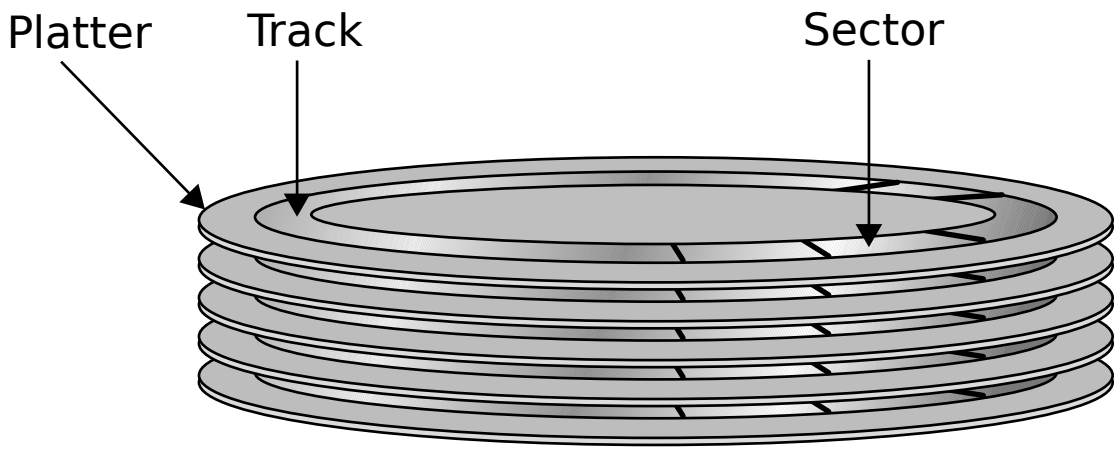
  Un bloc del sistema d'arxius té la mateixa mida o més gran (en múltiples sencers que són potència de 2) que la mida del sector de disc. Són comunes mides de bloc de 1024, 2048, 4096 o 8192 bytes.

  Una bona mida de bloc per Intel és 4096 bytes que és la mida de la pàgina de la memòria virtual. D'aquesta forma s'optimitza el sistema de la memòria virtual.

El disc dur magnètic

Diagrama d'un disc

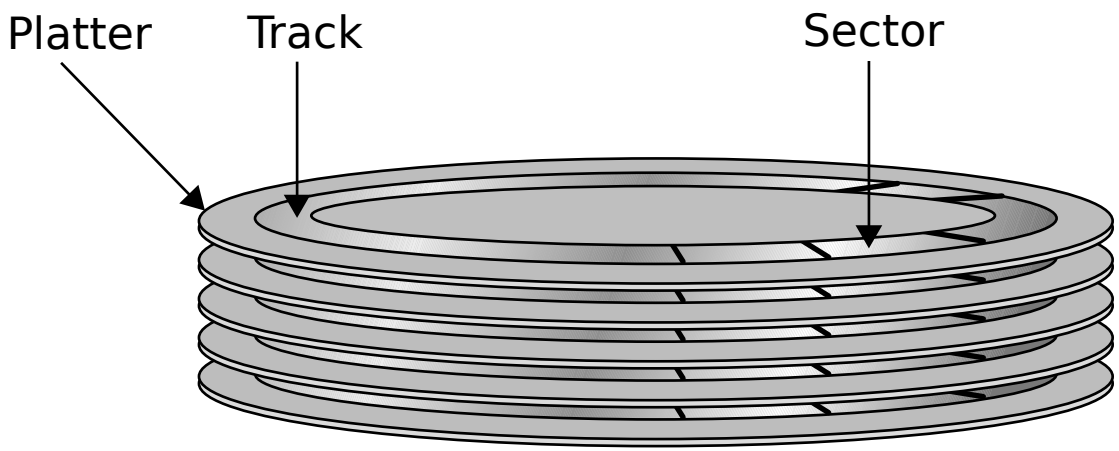
- Un disc dur magnètic està format per diversos discos (*platters*) col·locats un a sobre l'altre. Els discos es poden escriure per les dues bandes.
- Hi ha un únic braç que llegeix els discos amb un capçal de lectura per cada banda del disc. Només es permet una operació de lectura o escriptura en un instant determinat: no es pot llegir o escriure de múltiples discos a la vegada.



El disc dur magnètic

Diagrama d'un disc

- La quantitat mínima que es pot llegir o escriure a un disc és el sector o bloc, que típicament té una mida de 512 bytes. Els sectors estan repartits en pistes (*tracks*).
- Llegir o escriure sectors contigus és una operació molt més ràpida (+ 10 vegades) que fer-ho sobre sectors no contigus. Moure el braç és una operació molt lenta.



El disc dur magnètic

Existeixen multitud de discos de marques diferents, cadascú amb la seva geometria (nombre de discos, pistes, nombre de sectors per pistes, etc). El disc disposa d'un controlador de disc, integrat dins del maquinari. El gestor de disc es comunica amb el controlador per gestionar les operacions a realitzar.

- Els controladors actuals utilitzen l'anomenat adreçament lògic: el gestor de disc veu el disc com un dispositiu d'adreçament lineal. En accedir a una adreça lògica específica, el controlador s'encarrega de traduir l'adreça a la posició física corresponent al disc.
- El controlador disposa sovint d'uns MB de memòria *buffer*. S'augmenta l'eficiència en llegir perquè es llegeixen sectors contigus abans que el sistema operatiu els demani. En escriure, el controlador notifica al gestor que les dades estan escrites un cop arriben al buffer encara que físicament no hagin sigut escrites a disc.

Discos d'estat sòlid

Discos d'estat sòlid

- Tenen les mateixes característiques que discs durs tradicionals (sectors, ...) amb l'avantatge que no tenen parts mòbils. Això fa que el seu rendiment sigui molt superior.
- Abans de poder escriure a un bloc de disc cal esborrar-lo (això no passa amb els discs durs), una operació "lenta" en comparació amb la lectura o escriptura.
- Les dades s'emmagatzemen en circuits elèctrics. Els circuits, però, es poden anar degradant al llarg del temps i calen tècniques per assegurar la durabilitat. Cal "evitar" escriure sempre als mateixos sectors!

Requisits del sistema de fitxers: els fitxers i directoris

En un disc volem emmagatzemar fitxers i directoris. Per fer-ho

1. Cal que el sistema de fitxers (del sistema operatiu) emmagatzemi al disc les metadades associades al fitxer: la seva localització al disc, el nom del fitxer, la mida del fitxer, la protecció (drets d'accés), dates de creació i modificació, ...
2. Els sistemes de fitxers habituals permeten organitzar els fitxers en directoris. Els directoris també són metadades i han de permetre emmagatzemar el llistat de fitxers i subdirectoris que conté.

Aquestes metadades és informació que emmagatzema el sistema de fitxers a disc. Podem accedir al contingut d'aquestes metadades fent servir instruccions com "ls", per exemple.

Requisits del sistema de fitxers: operacions sobre fitxers

Quines operacions volem realitzar sobre els fitxers?

- Crear fitxers: el sistema de fitxers hauria de ser capaç de trobar ràpidament espai lliure al disc pel fitxer.
- Escriure a un fitxer: donat el nom d'un fitxer, el sistema de fitxers ha de ser capaç de trobar la localització del fitxer a disc i permetre escriure-hi.
- Llegir d'un fitxer: un cop trobada la localització del fitxer a disc s'han de poder llegir les dades que conté.
- Posicionament en un fitxer: s'ha de permetre posicionar el punt d'escriptura o lectura en qualsevol punt de fitxer.
- Esborrar un fitxer: aquesta operació implica eliminar el fitxer de les metadades del directori així com alliberar tot l'espai que ocupa el fitxer a disc.
- Truncar un fitxer: és una operació que permet alliberar l'espai que ocupa un fitxer sense esborrar-lo.

Requisits del sistema de fitxers: operacions sobre directoris

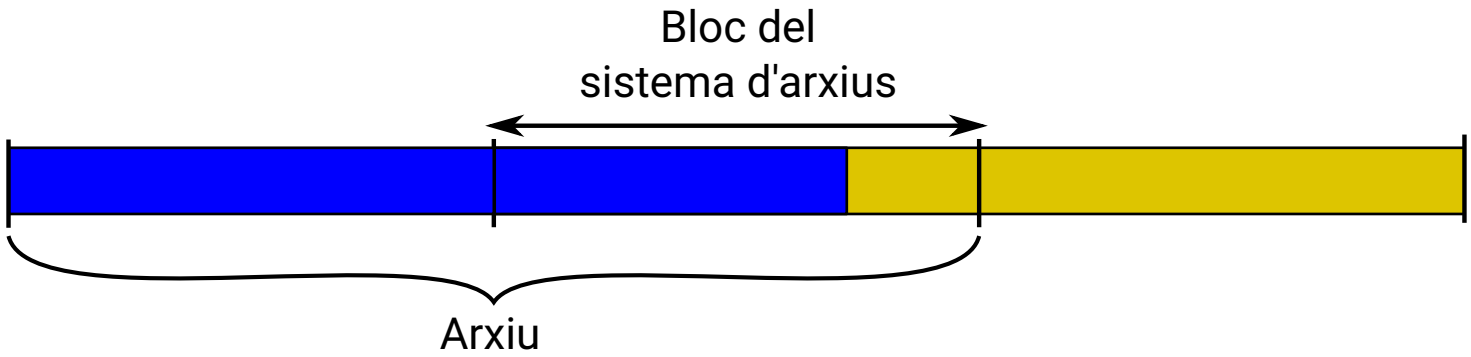
Quines operacions volem realitzar sobre els directoris?

- Buscar un fitxer: poder trobar un fitxer al directori per tal d'accedir a les metadades associades al fitxer.
- Crear un fitxer: permetre afegir nous fitxers a un directori.
- Esborrar un fitxer: permetre eliminar un fitxer d'un directori (i eliminar les metadades associades al fitxer).
- Reanomenar un fitxer: poder canviar el nom d'un fitxer del directori.
- Llistar el contingut d'un directori: poder llistar els fitxers (i subdirectoris) que conté un directori.

Requisits del sistema de fitxers: nivell físic

Als blocs del disc s'han d'emmagatzemar els arxius i les metadades associades.

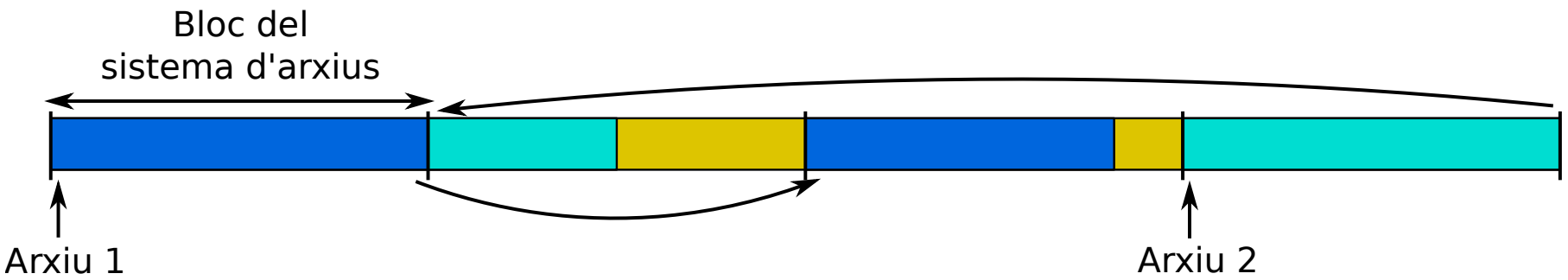
- Aquests "ocupen" un nombre sencer de blocs. Els bytes d'un bloc no utilitzats per un fitxer no poden ser utilitzats per altres arxius.
- No es pot llegir un byte d'un arxiu. Per fer-ho cal llegir tot el bloc del disc, emmagatzemar-lo a memòria i després accedir al byte demanat.



Requisits del sistema de fitxers: nivell físic

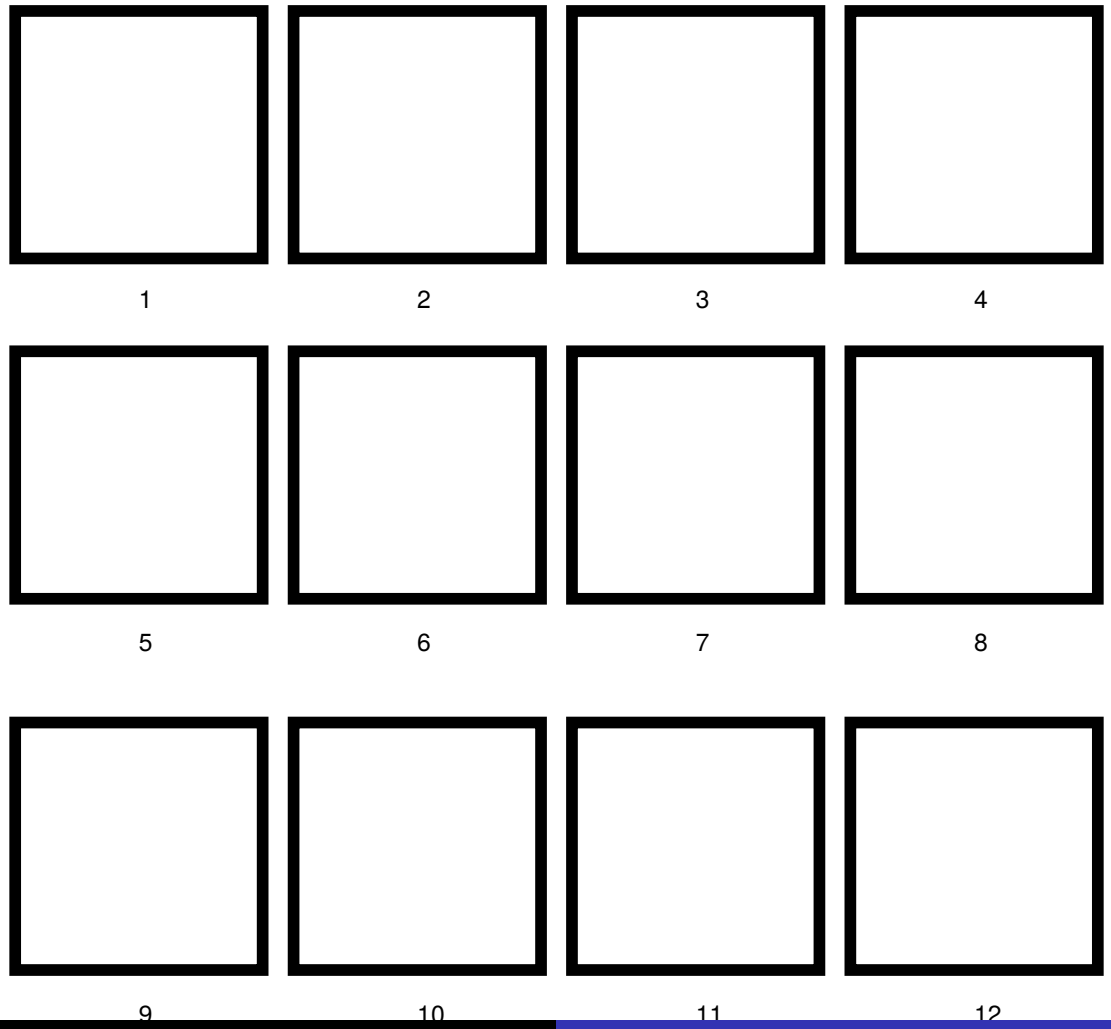
En un sistema d'arxius

- Els arxius (així com les seves metadades) no tenen perquè estar emmagatzemats de forma contigua al disc. Però des d'un punt de vista d'usuari del sistema d'arxius "veiem" els bytes dels arxius contigus (un byte al darrera l'altre).
- Idealment els blocs d'un arxiu han de ser contigus a disc, ja que així el rendiment del disc (dur) és major.

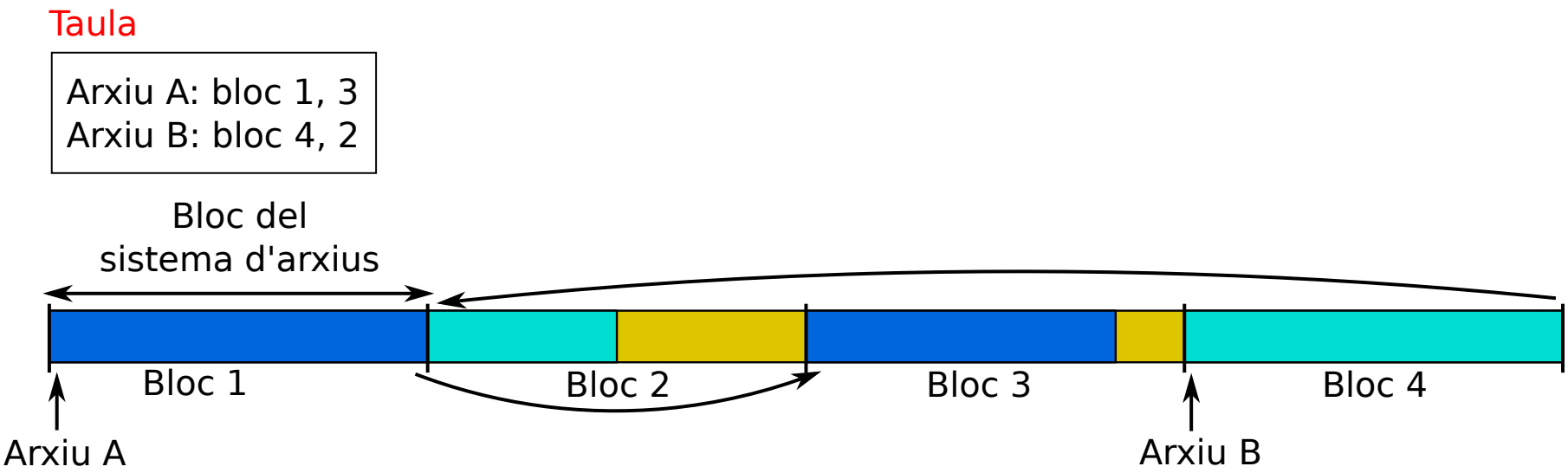


Fonaments dels dissenys implementats

Dissenyem el nostre sistema de fitxers! Donat els requisits, feu una proposta... Com emmagatzemem els fitxers i els blocs dels quals estan formats? Com emmagatzemem els directoris?



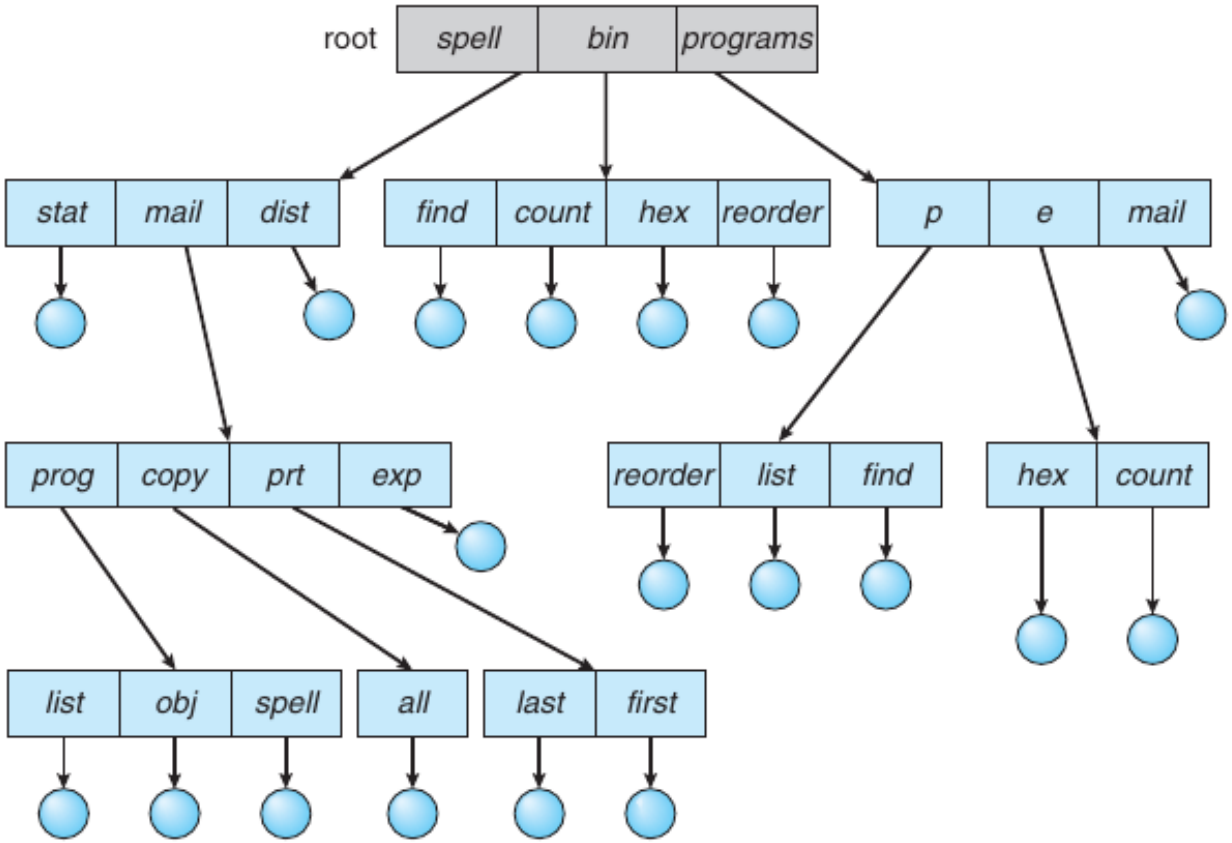
Fonaments dels dissenys implementats



Com implementar el fet que un arxiu està format per blocs no contigus? Una forma senzilla és mitjançant una taula o una llista enllaçada en què s'indiquen els blocs que formen part del fitxer. Al llarg de l'evolució dels sistemes d'arxius hi ha hagut diverses formes d'implementar-ho. Com fer-ho de forma que es pugui accedir de forma eficient a un bloc d'un arxiu i sigui escalable per fitxers petits i grans?

Fonaments dels dissenys implementats

Com emmagatzemem la informació dels directoris? A l'actualitat estem acostumats a fer servir sistemes d'arxius que permeten fer servir una estructura de directori...



Fonaments dels dissenys implementats

De forma intuïtiva, en un disc cada directori s'emmagatzema com un fitxer amb una estructura similar a una taula. Aquesta taula

- Conté el llistat de fitxers, amb la mida, els permisos, ... així com un "apuntador" cap als blocs que formen part del fitxer.
- Cada directori pot contenir subdirectoris. Per a cada subdirectori hi a un "apuntador" cap al fitxerque conté els fitxers del subdirectori.

Com gestionar aquesta taula de forma eficient? No és una tasca senzilla...

Fonaments dels dissenys implementats

Com gestionar l'espai lliure al disc?

- Els sistema de fitxers ha de saber quins blocs estan lliures. I si són contigus, millor! Els sistemes de fitxers han d'intentar evitar la fragmentació d'un fitxer.
- L'organització dels fitxers (a un disc magnètic) s'hauria de fer de forma que els directoris (les fulles de càlcul) i els fitxers continguts als directoris estiguin físicament propers. És el que en termes tècnics s'anomena l'heurística de localitat.

Objectiu: disseny d'un sistema de fitxers!

Es disposa d'una gran flexibilitat a l'hora de dissenyar un sistema de fitxers. En dissenyar el sistema,

- Les dades dels fitxers haurien de ser contigües al disc.
- Disposar d'un lloc al disc on emmagatzemar les metadades dels fitxers.
- Accés eficient a un bloc d'un fitxer (evitar haver de moure gaire el braç del disc).
- Ser escalable per donar suport a fitxers petits i grans.

S'han proposat múltiples sistemes de fitxers. Anem a veure l'evolució històrica amb quatre sistemes diferents.

Evolució històrica

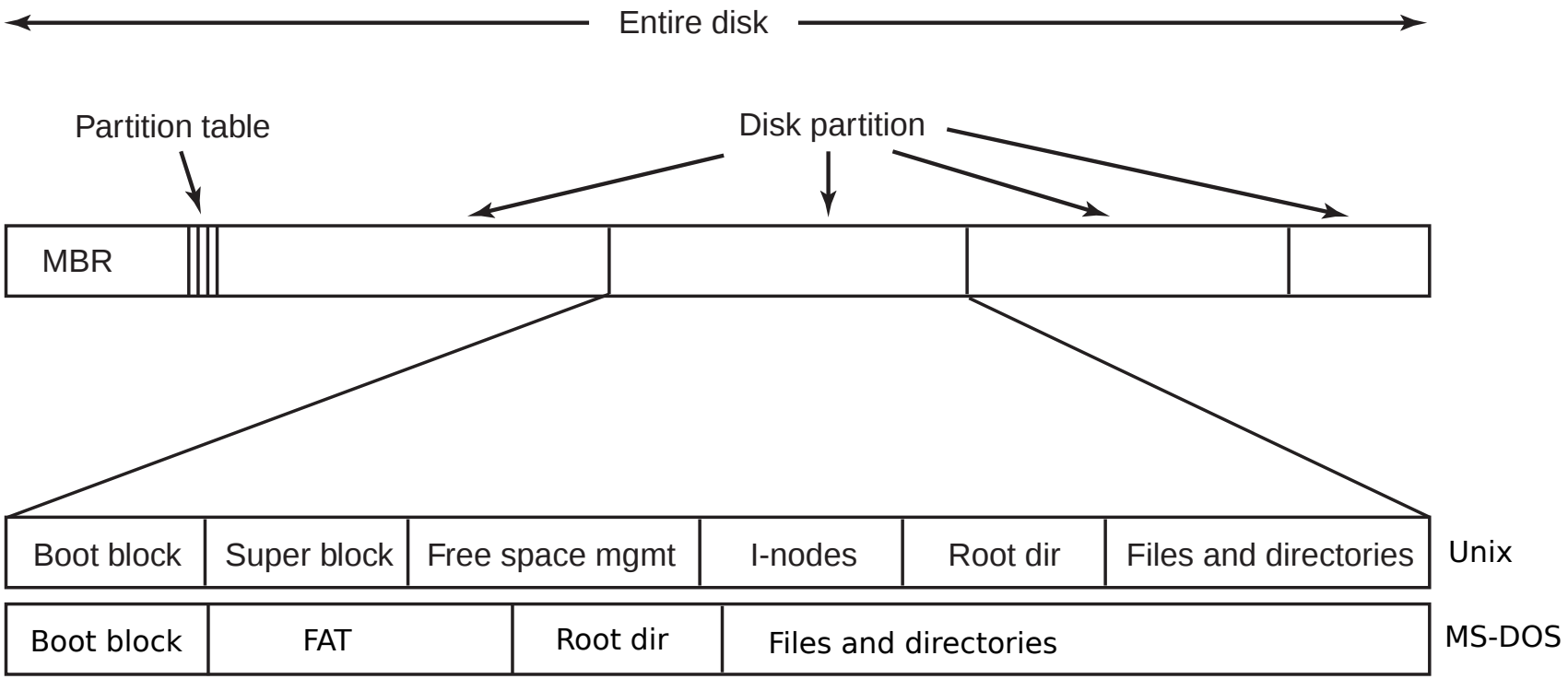
Aquests sistemes de fitxers s'utilitzen molt encara avui en dia

- FAT (File Allocation Table). Desenvolupat a finals del 1970, utilitzat avui en dia a memòries flash i càmeres digitals on la simplicitat és primordial.
- FFS (Unix Fast File System). Desenvolupat a mitjans de 1980. El sistema ext2 i ext3 de Linux està basat en les idees d'aquest sistema.
- NTFS (Microsoft New Technology File System). Introduït a principis del 1990 i utilitzat als sistemes Windows. És representatiu del ext4 de Linux, Reiser4, Journaled File System (JFS) o l'Apple Hierarchical File System (HFS).
- COW (ZFS, btrfs). El sistema ZFS va ser introduït a principis del 2000 per Sun Microsystem.

Anem a veure'ls amb una mica de detall per veure l'evolució històrica...

Organització física d'un disc

Es mostra aquí un esquema simplificat de l'organització d'un disc. Un disc es pot dividir en particions i, a cada partició, tenir un sistema de fitxers diferent.



Sistemes FAT

Els sistemes FAT

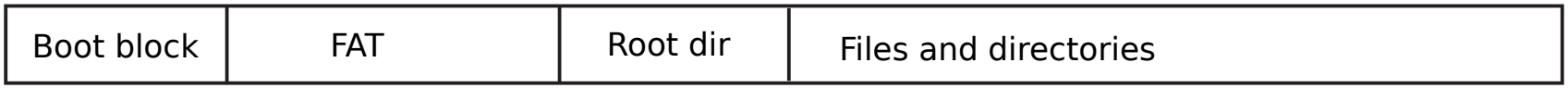
- Desenvolupat per Microsoft a finals del 1970. Va ser utilitzat pel sistemes operatius MS-DOS i les primeres versions de Windows.
- Ha sigut millorat aquests darrers anys per donar suport a la "gran" capacitat d'emmagatzematge dels dispositius dels darrers anys.
- Ens centrarem en la versió més recent, FAT-32, capaç de gestionar discos amb 2^{28} blocs i fitxers de fins a $2^{32} - 1$ bytes (4 GB).

Sistemes FAT

En sistemes FAT està estructurat d'aquesta forma (en localitzacions conegudes pel sistema de fitxers).

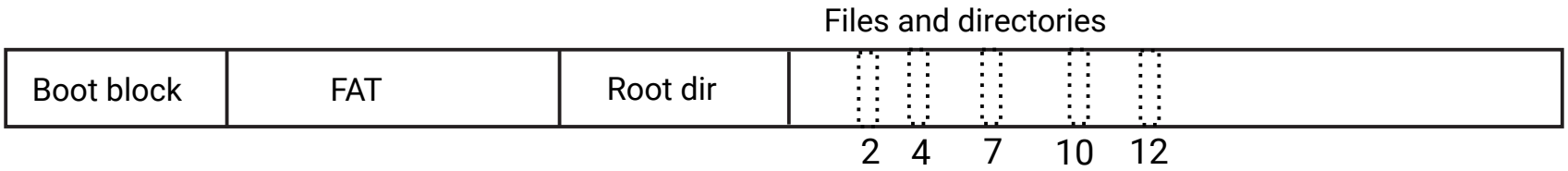
- La taula FAT: conté informació sobre els blocs que formen part de cada fitxer.
- El directori arrel: el directori arrel és el "fitxer" que conté la informació sobre els subdirectoris i els fitxers que formen el directori arrel.
- Fitxers i directoris: aquí és on s'emmagatzemen els fitxers i directoris del disc. Es va omplint a mesura que s'hi afegeixen dades.

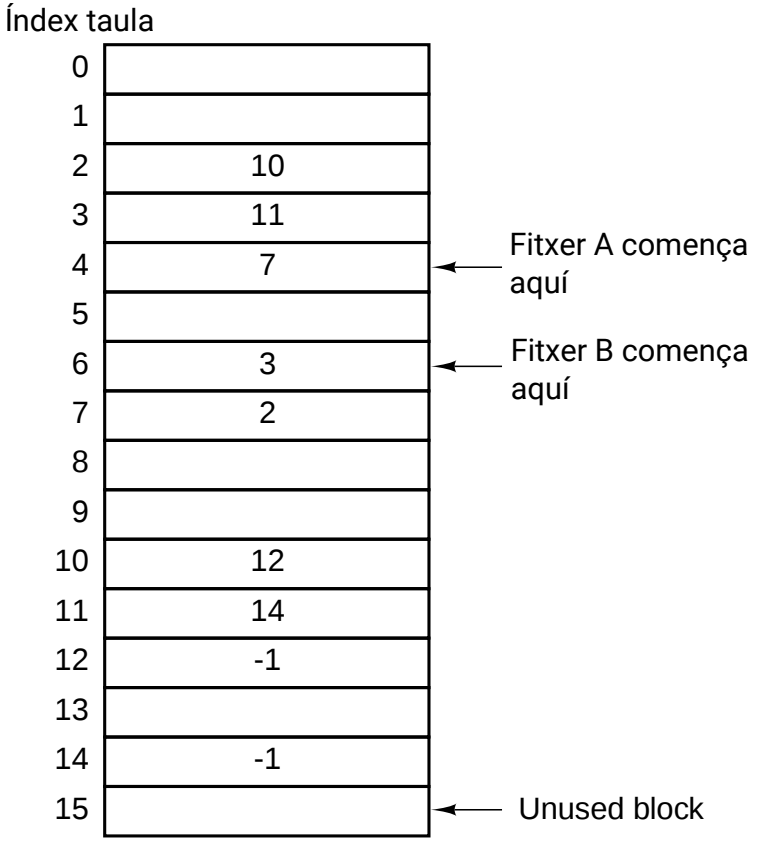
Ara veurem els detalls de cada punt...



La taula FAT (File Allocation Table)

La FAT és una llista de sencers (de 4 bytes) enllaçada que emmagatzema els blocs de què es componen els fitxers.





Per a l'exemple:

- El fitxer A ocupa els blocs 4, 7, 2, 10 i 12. El fitxer B ocupa els blocs 6, 3, 11 i 14.
- El blocs buits així com el "final" d'un fitxer es marquen amb un codi especial.

Els directoris a la FAT

Els fitxers de directoris són senzills: una taula (no ordenada) dels fitxers que componen el directori.

- Per a cada fitxer s'hi emmagatzemen metadades com la mida en bytes, la seva data de creació, ...
- També s'hi emmagatzema el primer índex a la taula. Pel fitxer A, s'hi emmagatzemaria un 4. Pel fitxer B s'hi emmagatzemaria un 6.

Fitxer B	Mida en bytes, data creació, etc.	Primer index a la taula FAT
Fitxer A	Mida en bytes, data creació, etc.	Primer index a la taula FAT
Fitxer Z	Mida en bytes, data creació, etc.	Primer index a la taula FAT
Fitxer G	Mida en bytes, data creació, etc.	Primer index a la taula FAT
Fitxer K	Mida en bytes, data creació, etc.	Primer index a la taula FAT

Trobar espai buit i fitxers contigus a la FAT

El sistema de fitxers fa servir un esquema molt senzill

- Per trobar espai buit per un nou fitxer, el sistema de fitxers recorre la taula FAT per veure si hi ha un element no ocupat (marcat amb un 0 a la taula FAT).
- S'acostumen a fer servir algorismes molt senzills per emmagatzemar fitxers que ocupin múltiples blocs: se cerca a la taula "el següent" bloc lliure. Això fa que els fitxers a un sistema FAT puguin estar molt fragmentats[1].

[1] Els sistemes FAT disposen d'un programari, un defragmentador, que "mou" les dades al disc per aconseguir que els fitxers estiguin contigus al fitxer

Algunes preguntes del sistema FAT

Algunes preguntes

1. Suposem un fitxer de 3MB a volem accedir a la posició 2.000.000 del fitxer. Detallar com se sap el bloc a llegir per accedir-hi.
2. Com se cerca un fitxer a la FAT?
3. Com s'esborra un fitxer a la FAT? Com ho faríeu?
4. Com es crea un fitxer a la FAT? Com ho faríeu?

Limitacions del sistema FAT

El sistema de fitxers FAT es molt utilitzat per la seva simplicitat (dispositius USB, càmeres digitals, ...). Però té moltes limitacions

- Fitxers fragmentats: l'estratègia utilitzada per aquest sistema de fitxers fa que els fitxers acostumin a estar molt fragmentats.
- Accés aleatori pobre: per poder accedir a un bloc del fitxer, cal recórrer la llista enllaçada de la taula FAT.
- Taula de fitxers d'un directori: el llistat de fitxers no està "ordenat" i per tant... ufff.
- Control d'accés pobre: no es permet controlar l'accés a diferents usuaris.
- Limitació en la mida de disc i fitxer: en un disc amb blocs de 4KB, la mida màxima del disc és de 1TB i la de fitxer 4GB.
- Seguretat: el sistema FAT no incorpora sistemes de seguretat (com els sistemes moderns) per assegurar, per exemple, la corrupció del disc en cas l'ordinador es pengi.

Evolució històrica

Aquests sistemes de fitxers s'utilitzen molt encara avui en dia

- FAT (File Allocation Table). Desenvolupat a finals del 1970, utilitzat avui en dia a memòries flash i càmeres digitals on la simplicitat és primordial.
- FFS (Unix Fast File System). Desenvolupat a mitjans de 1980. El sistema ext2 i ext3 de Linux està basat en les idees d'aquest sistema.
- NTFS (Microsoft New Technology File System). Introduït a principis del 1990 i utilitzat als sistemes Windows. És representatiu del ext4 de Linux, Reiser4, Journaled File System (JFS) o l'Apple Hierarchical File System (HFS).
- COW (ZFS, btrfs). El sistema ZFS va ser introduït a principis del 2000 per Sun Microsystem.

Anem a veure'ls amb una mica de detall per veure l'evolució històrica...

Sistemes FFS (Unix Fast File System)

Els sistemes FFS incorpora idees interessants

- Es permet controlar quins usuaris poden accedir (per lectura, escriptura o execució) a cadascun dels fitxers.
- Els blocs que formen part d'un fitxer s'indexen de forma que es pugui treballar de forma eficient tant per fitxers petits com fitxers grans.
- El sistema inclou heurístiques que aconsegueixen que els fitxers estiguin "repartits" al disc en blocs contigus.

Sistenes FFS (Unix Fast File System)

En sistemes Unix, en general, està estructurat d'aquesta forma (en localitzacions conegudes pel sistema de fitxers).

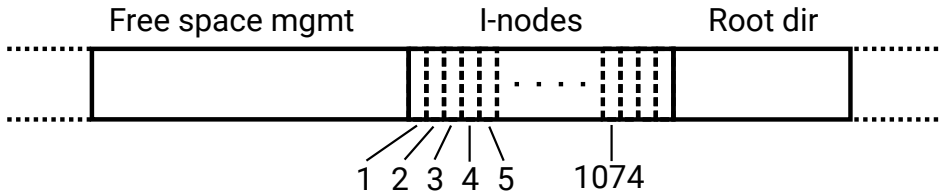
- Els i-nodes es fan servir per emmagatzemar els blocs que componen un fitxer.
- El directori arrel: el directori arrel és el "fitxer" que conté la informació sobre els subdirectoris i els fitxers que formen el directori arrel.
- El Free space mgmt és on s'emmagatzema la informació els blocs lliures a disc.
- Fitxers i directoris: aquí és on s'emmagatzemen els fitxers i directoris del disc. Es va omplint a mesura que s'hi afegeixen dades.

Sistemes FFS (Unix Fast File System)

En sistemes i-nodes s'emmagatzema aquesta informació en posicions conegudes:

- Els i-nodes: es poden interpretar com una taula indexada. Cada i-node té una mida fixe (en bytes) i té associat un índex en aquesta taula.

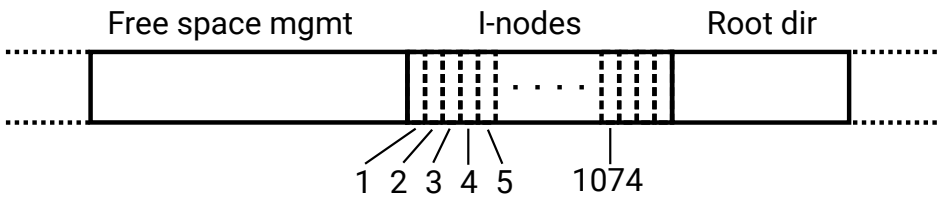
  En especificar l'i-node 1074, per exemple, el sistema de fitxers sap localitzar-lo de forma immediata al disc.



Els i-nodes

L'i-node és una estructura de dades que emmagatzema els blocs que formen part d'un fitxer.

- Cada i-node emmagatzema els atributs així com els blocs que formen part d'un fitxer (vegeu següent transparència).
- Un directori, intuïtivament, no és més que un llistat amb els noms de fitxers i l'i-node que li correspon.



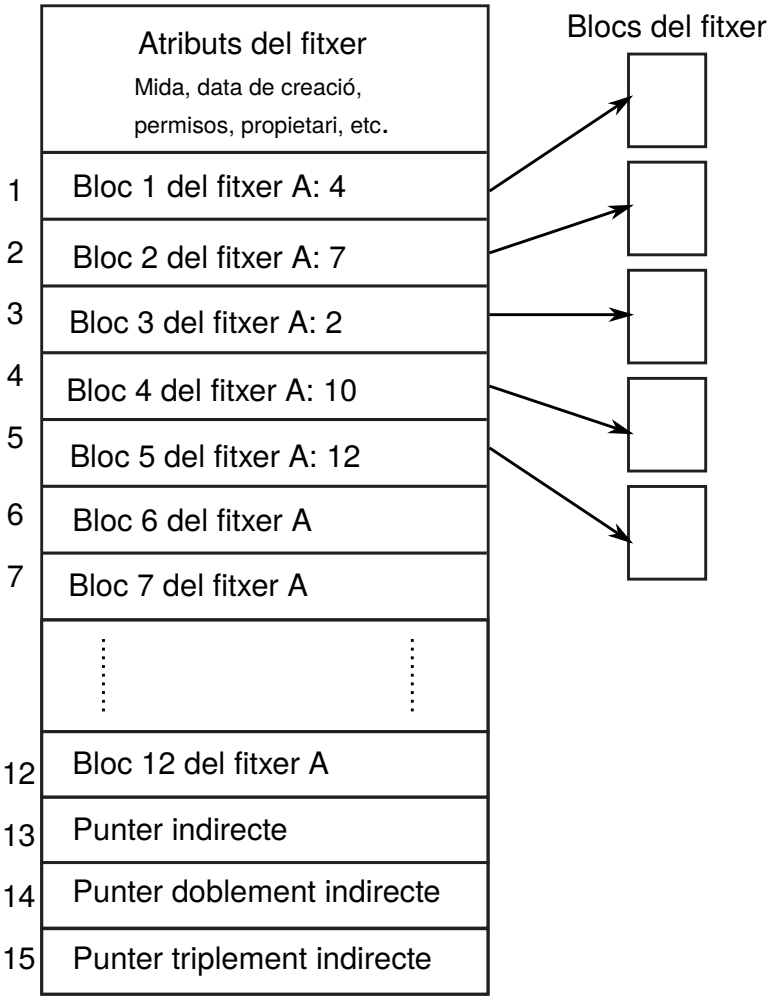
Organització del disc

Fitxer B	i-node
Fitxer A	i-node
Fitxer Z	i-node
Fitxer G	i-node
Fitxer K	i-node

Fitxer de directori

Els i-nodes

Un i-node es pot interpretar com un arbre, del qual l'arrel és l'inode, i les fulles els blocs del fitxer de dades.



Els i-nodes

Respecte els i-nodes

- Hi ha un i-node per a cada fitxer. Cada i-node té una mica fixa i, per tant, el sistema de fitxers en pot localitzar un ràpidament.
- L'i-node emmagatzema els atributs del fitxer.
- A més, conté un array de punters cap als blocs que formen el fitxer. En total n'hi ha 15: els 12 primers són punters directes (veieu dibuix anterior), i després conté altres tipus de punters... veiem-ho!
- Els fet de fer servir una estructura en arbre fa que es pugui accedir a qualsevol bloc del fitxer de forma eficient.

Els i-nodes

Respecte els i-nodes:

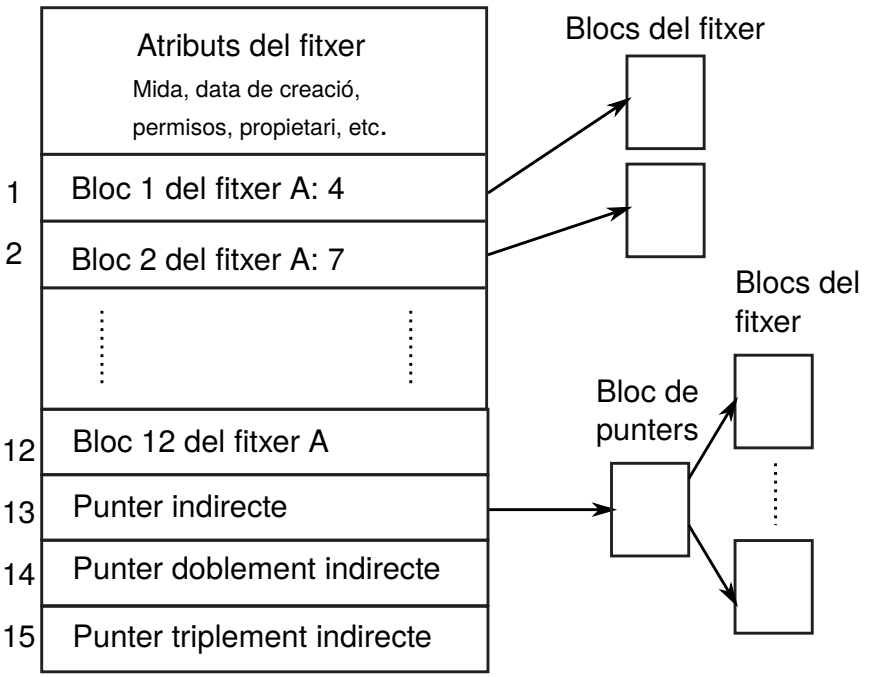
- En obrir un fitxer es llegeix l'i-node corresponent de disc i s'emmagatzema a memòria. Immediatament se saben els blocs que formen el fitxer.

Però...

- En un sistema de fitxers amb una mida de bloc de 4096 bytes, si només fem servir els punters directes implica una mida màxima de fitxer de $12 \times 4096 = 48\text{KB}$. Això és suficient per a molts fitxers.
- Què passa si el fitxer es fa més gran que el limit de 48KB? Se soluciona amb els punters indirectes, doblement indirectes i triplement indirectes...

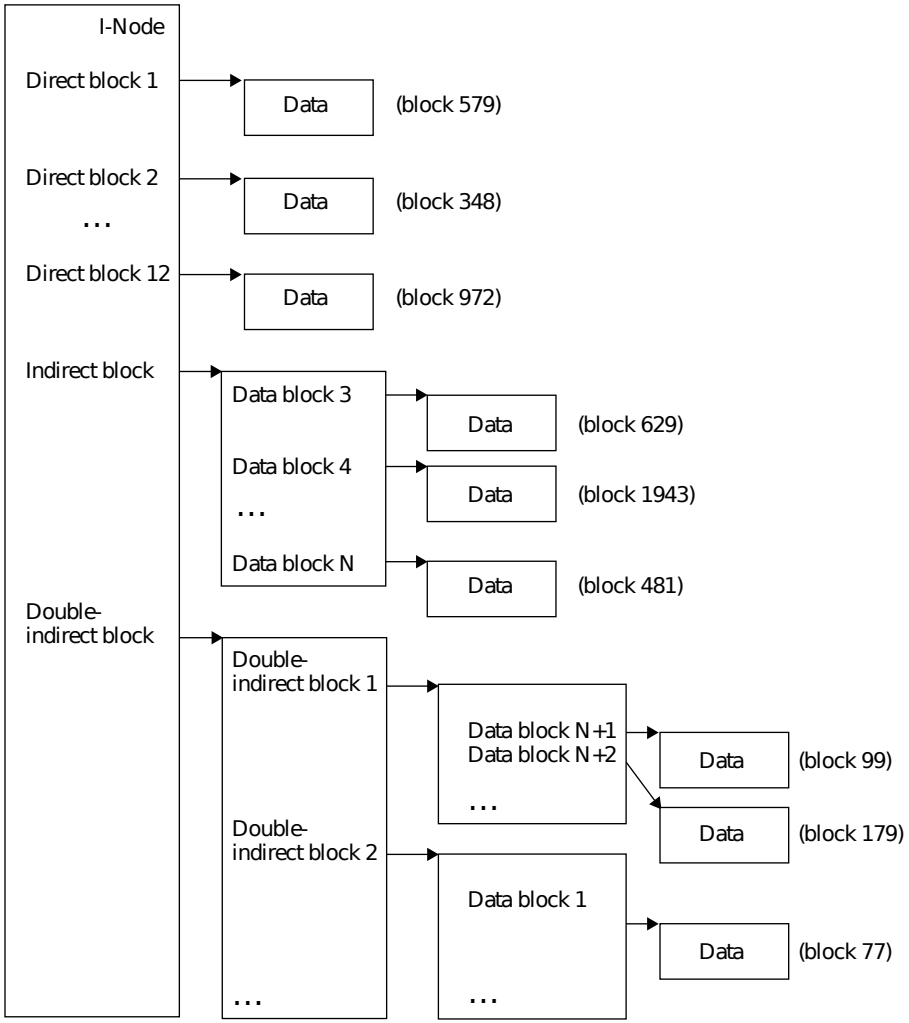
Els i-nodes

Els punters indirectes apunten a un bloc que conté (només) punters directes. Amb blocs 4 KB i punters de 4 bytes, un bloc de punters (indirectes) pot contenir 1024 punters. Això implica ara una mida de fitxer de $48\text{KB} + 1024 \times 4\text{KB} \approx 4\text{MB}$.



Els i-nodes

Els i-nodes, a més de les referències directes i les indirectes també suporten referències doblement indirectes:



Els i-nodes

En resum

- Amb 12 punters directes podem apuntar a 48KB de dades.
- Amb 1 punter indirecte (que emmagatzema 1024 punters directes) podem apuntar a 4MB de dades.
- Amb 1 punter doblement indirecte (que pot apuntar a 1024^2 punters directes) podem apuntar a 4GB de dades.
- Amb 1 punter triplement indirecte podem apuntar a 4TB de dades

El sistema FFS utilitza un arbre fixe multinivell asimètric capaç de donar suport a fitxers petits i grans de forma eficient. Múltiples sistemes de fitxers diferents han adoptat aquest enfocament.

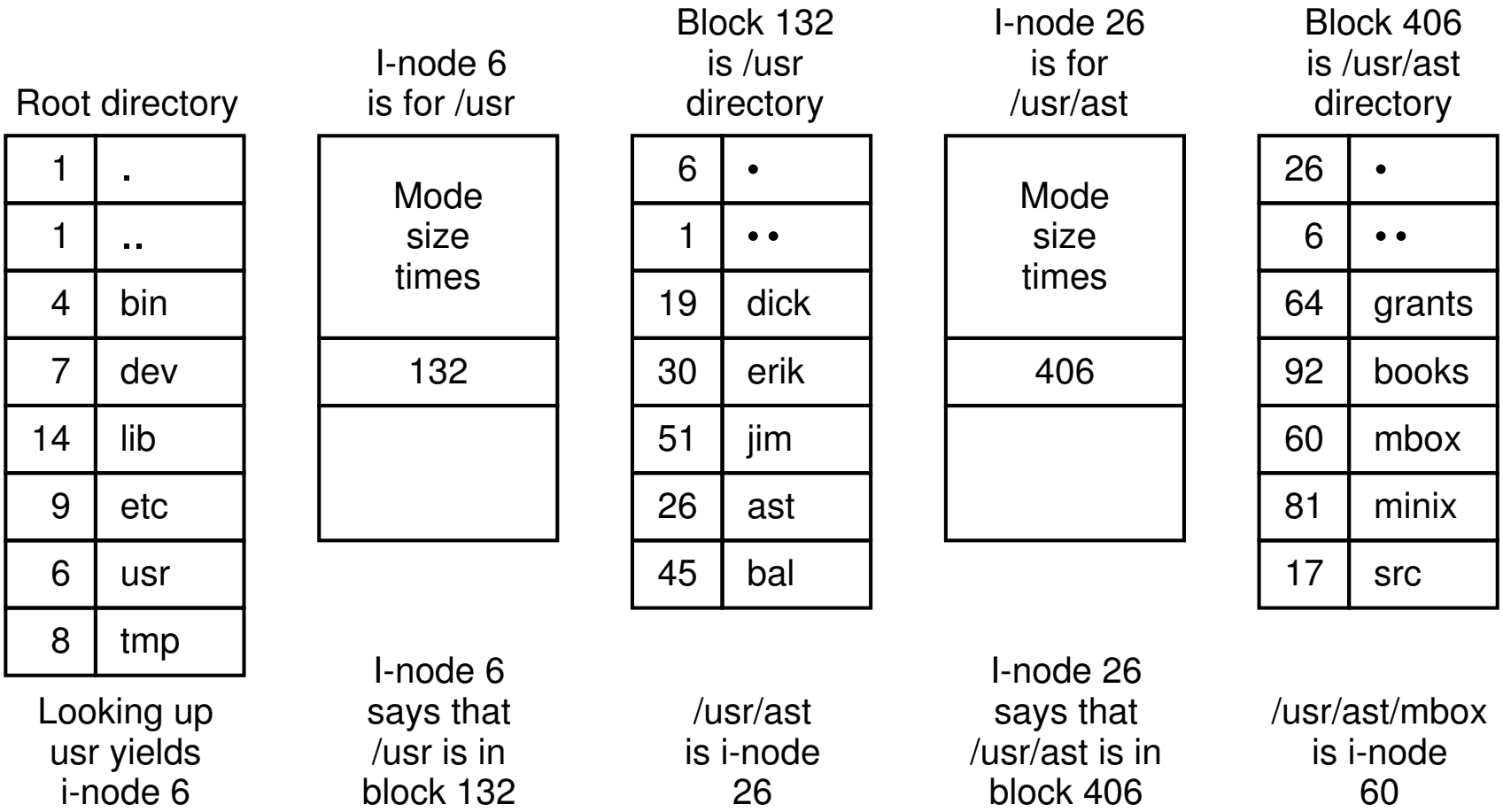
Els i-nodes: algunes preguntes

Algunes preguntes

1. Suposem un fitxer de 3MB i volem accedir (i.e. llegir) al byte 2.000.000. Detallar com se sap el bloc a llegir per accedir-hi.
2. L'esquema presentat utilitza un arbre asimètric. Per què és una solució eficient per a fitxers petits i grans?
3. Com se cerca un fitxer al sistema FFS?
4. Com s'esborra un fitxer al sistema FFS? Com ho faríeu?
5. Com es crea un fitxer al sistema FFS? Com ho faríeu?

Els i-nodes: exemple de localització d'un fitxer

Veiem un exemple complet. Suposem que es vol obrir el fitxer `/usr/ast/mbox` del sistema de fitxers. Quins passos s'han de seguir per aconseguir-ho ?

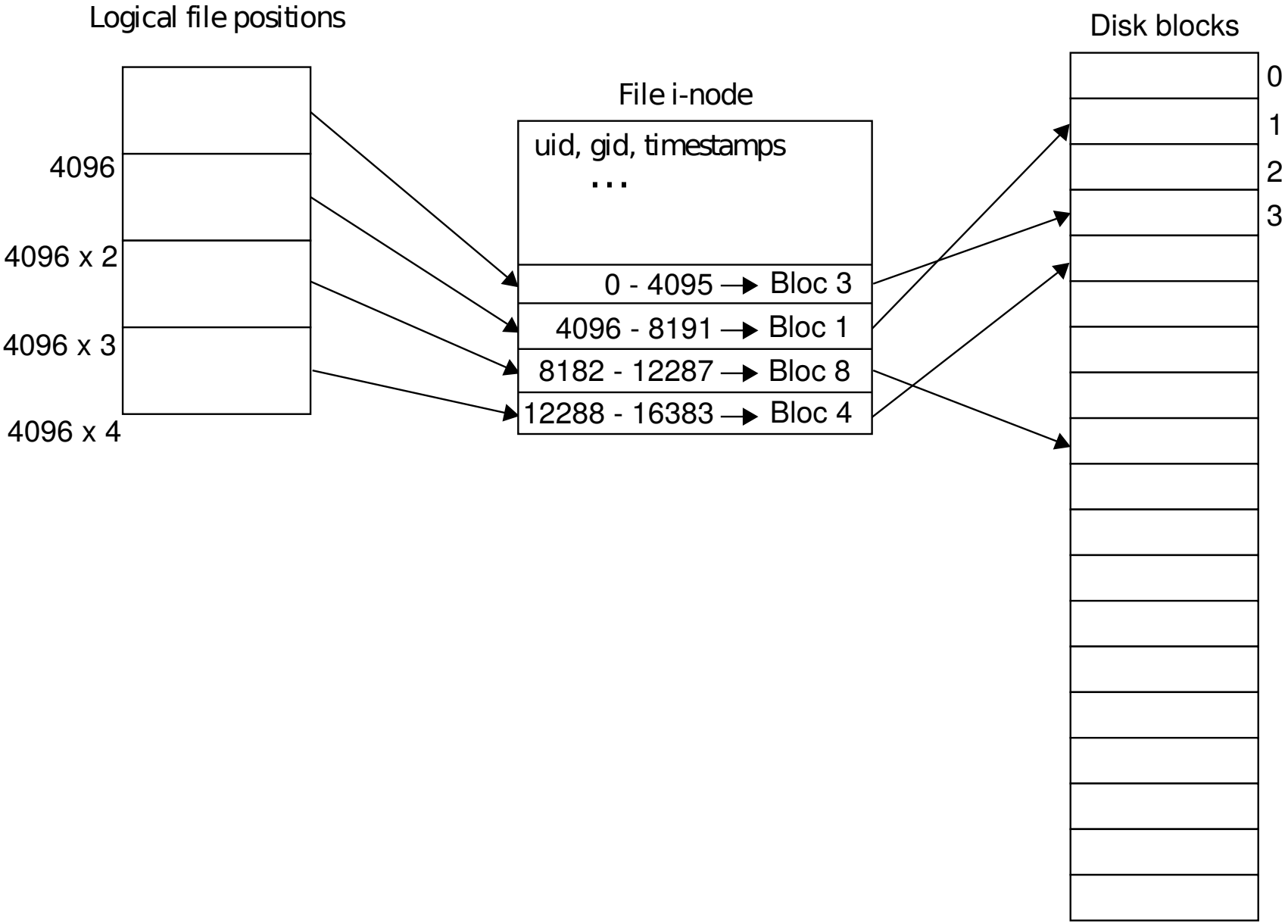


Els i-nodes: exemple de localització d'un fitxer

Passos a seguir

1. El sistema de fitxers coneix la localització del directori arrel. Cal cercar-hi l'entrada `usr`.
2. L'entrada `usr` fa referència a l'i-node 6 en què hi ha indicat que els continguts d'aquest directori es troben al bloc 132.
3. Al bloc 132 es cerca la paraula `ast`.
4. El directori `ast` es troba a l'i-node 26 en què hi ha indicat que els continguts són al bloc 406.
5. Al bloc 406 es cerca la paraula `mbox`.
6. El fitxer `mbox` té associat l'i-node 60. Es llegeix aquest i-node i ja es pot obrir el fitxer! Un cop obert, el sistema operatiu guarda internament l'i-node associat al fitxer.

Els i-nodes: exemple de lectura d'un fitxer



Els i-nodes: exemple de lectura d'un fitxer

A la figura anterior

- Se suposa una mida de bloc de 4096 bytes.
- Un usuari del sistema operatiu "veu" que els bytes (o blocs) que formen un fitxer qualsevol són contigus .
- L'i-node fa el mapat entre els bytes lògics i la posició física a disc.

Exemple

- L'usuari executa la instrucció C

```
fread(&a, sizeof(char), 4096*4, fp);
```

- El codi del sistema d'arxius s'ha d'encarregar de
  - Dividir la instrucció anterior en els blocs que el componen.
  - Fer la petició de lectura de cadascun dels blocs.
  - Unir cadascun dels blocs llegits en un sol vector.

FFS: resum del que hem vist fins ara

Resumim

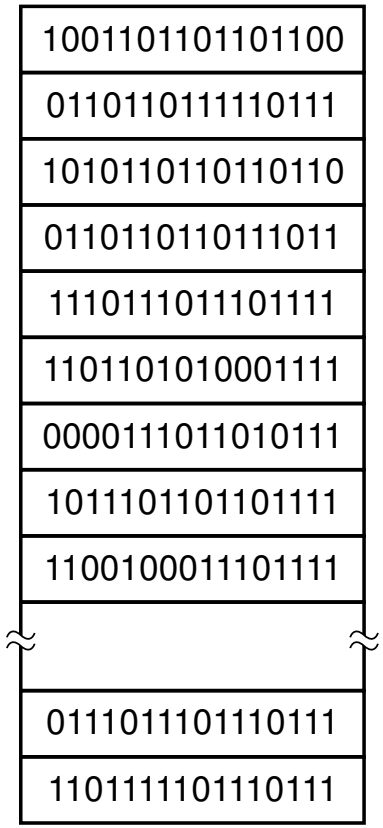
- Cada fitxer té associat un i-node, el qual es pot interpretar com l'arrel d'un arbre que emmagatzema els blocs que formen part del fitxer.
- L'arbre és asimètric i permet tractar de forma eficient fitxers petits i grans (nodes directes, nodes indirectes, nodes doblement indirectes, nodes triplement indirectes).

Falta per veure

- Com es gestiona l'espai lliure?
- Com podem gestionar l'heurística de la localitat dels fitxers?

FFS: com es cerca espai buit?

Per cercar espai buit, el FFS utilitza un *bitmap* emmagatzemat a disc en què cada bloc es representa amb un bit.

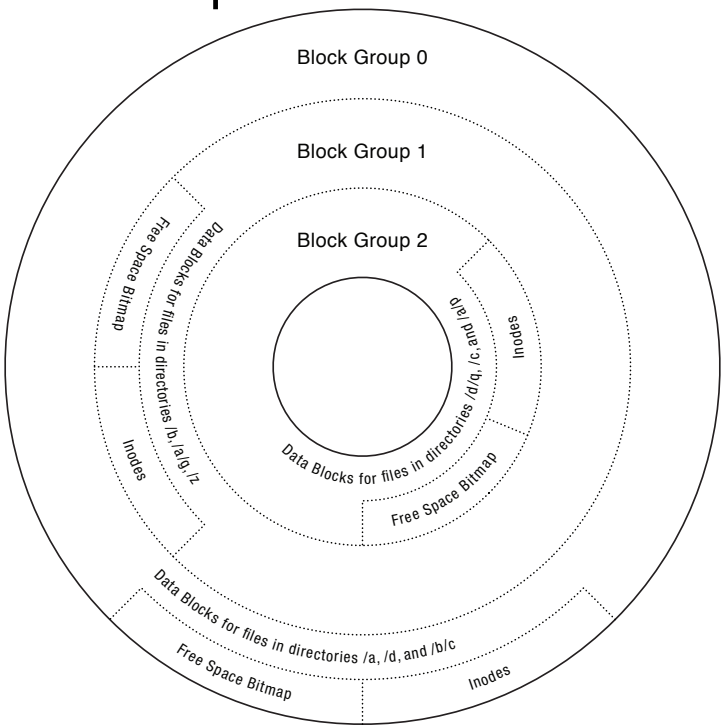


- El *bimap* s'emmagatzema en una posició coneguda pel sistema de fitxers.
- La cerca d'espai lliure es pot fer de forma "intel·ligent" per tal d'intentar aconseguir que els blocs d'un fitxer siguin contigus. Com s'aconsegueix?

FFS: heurística de localitat

En el sistemes basats en FFS

- El disc es divideix el grups. Cada grup permet emmagatzemar les metadades (els i-nodes, directoris, el bitmap d'espai buit, ...).
- Un determinat directori està associat a un determinat grup. El sistema FFS (intenta) emmagatzemar els fitxers del directori en el grup que li correspon.



Què opineu del sistema FFS

El sistema FFS és la "base" dels sistemes de fitxers més moderns. Comparem amb el FAT...

- Fitxers fragmentats?
- Accés aleatori pobre?
- Taula de fitxers d'un directori?
- Control d'accés pobre?
- Limitació en la mida de disc i fitxer?
- Seguretat?

Evolució històrica

Aquests sistemes de fitxers s'utilitzen molt encara avui en dia

- FAT (File Allocation Table). Desenvolupat a finals del 1970, utilitzat avui en dia a memòries flash i càmeres digitals on la simplicitat és primordial.
- FFS (Unix Fast File System). Desenvolupat a mitjans de 1980. El sistema ext2 i ext3 de Linux està basat en les idees d'aquest sistema.
- NTFS (Microsoft New Technology File System). Introduït a principis del 1990 i utilitzat als sistemes Windows. És representatiu del ext4 de Linux, Reiser4, Journaled File System (JFS) o l'Apple Hierarchical File System (HFS).
- COW (ZFS, btrfs). El sistema ZFS va ser introduït a principis del 2000 per Sun Microsystem.

Anem a veure'ls amb una mica de detall per veure l'evolució històrica...

NTFS

Recordem que en sistemes FFS:

- El sistema FFS utilitza i-nodes amb estructura d'arbre "fixa" (transparència 41).
- Es fa referència a les dades del fitxer fent servir punters a blocs de mida de 4KB.
- Un directori és un llistat lineal que conté els noms dels fitxers (desordenats) i els seus i-nodes corresponents.

Els sistemes NTFS (i similars) introdueixen més "flexibilitat" respecte els i-nodes del FFS, així com noves característiques.

- Es fan servir estructures d'i-nodes amb estructura més flexible. En comptes de fer referència a blocs individuals de 4KB es fa referència a blocs de mida variable (anomenats "extents").
- Els directoris s'emmagatzemen fent servir arbres balancejats[2]. Donat un nom de fitxer, es calcula un valor de hash associat que permet localitzar ràpidament el fitxer en el directori.

[2] El mateix es fa amb l'espai disponible de disc.

NTFS: Journalling

Com podem assegurar la consistència de les metadades en cas d'una fallada del sistema ? Amb una tècnica coneguda amb el nom de *journalling* (registre per diari).

Aquesta tècnica fou inventada per la comunitat de base de dades per garantir que les estructures emmagatzemades són consistents. Ha estat traslladada als sistemes de fitxers per assegurar que les estructures es mantenen consistents independentment de les fallades que hi pugui haver.

NTFS: Journalling

En sistemes de fitxers el sistema de journalling:

- Permet garantir la consistència de les metadades dels fitxers, no de les dades en sí de cada fitxer. Les metadades són tot allò que no són dades emmagatzemades al fitxer.
- En produir-se una fallada del sistema es pot recuperar el disc en qüestió de segons independentment de la mida del disc. Si el sistema no té *journalling* pot trigar molts minuts.
- Exemples de sistemes amb *journalling*: ext3, ext4, NTFS, ... però no el suporten els sistemes FAT o ext2.

NTFS: Journalling

Intuïtivament, com funciona?

- S'implementa mitjançant un "fitxer" especial, un registre (log) circular en què s'emmagatzemen les operacions previstes a realitzar a disc.
- Abans de realitzar una modificació als i-nodes s'indica (s'emmagatzema) al registre circular quina operació es realitzarà. Un cop realitzada, es marca com a realitzada al registre circular.
- Si l'ordinador falla (pex s'apaga el llum) mentre es realitza una operació es podrà comprovar, en engegar el sistema operatiu, si totes les operacions han estat realitzades. En cas negatiu, es pot procedir a "corregir" els i-nodes corresponents.

Evolució històrica

Aquests sistemes de fitxers s'utilitzen molt encara avui en dia

- FAT (File Allocation Table). Desenvolupat a finals del 1970, utilitzat avui en dia a memòries flash i càmeres digitals on la simplicitat és primordial.
- FFS (Unix Fast File System). Desenvolupat a mitjans de 1980. El sistema ext2 i ext3 de Linux està basat en les idees d'aquest sistema.
- NTFS (Microsoft New Technology File System). Introduït a principis del 1990 i utilitzat als sistemes Windows. És representatiu del ext4 de Linux, Reiser4, Journaled File System (JFS) o l'Apple Hierarchical File System (HFS).
- COW (ZFS, btrfs). El sistema ZFS va ser introduït a principis del 2000 per Sun Microsystem.

Anem a veure'ls amb una mica de detall per veure l'evolució històrica...

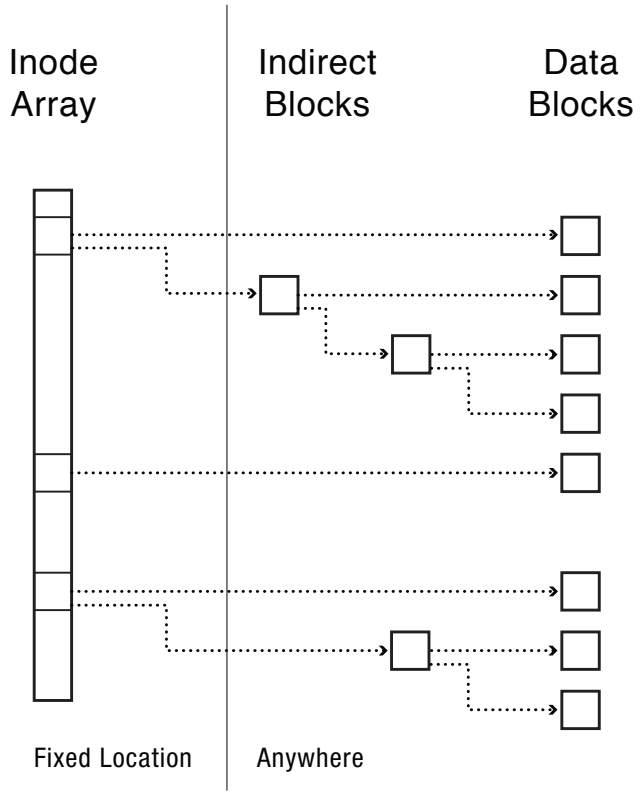
COW: Copy on Write

Què falta per millorar?

- Els sistemes de fitxers NTFS només s'asseguren seguretat sobre les metadades, no les dades del fitxer... què es pot fer?
- Els sistemes Copy on Write ho solucionen amb una proposta interessant. En actualitzar les dades d'un fitxer, no se sobreescriuen (actualitzen) les metadades i dades que hi ha escrites, sinó que escriuen noves versions a noves posicions obtenint una còpia del fitxer... Anem a veure els principis.

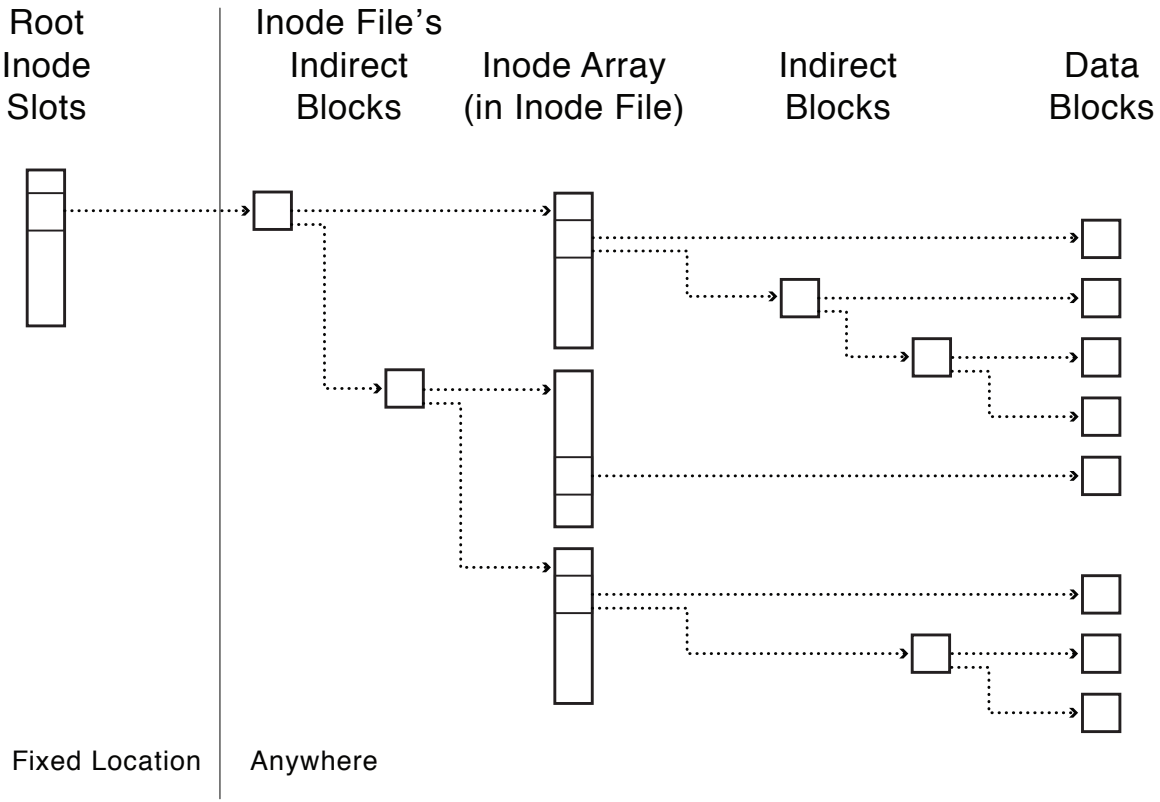
COW: Copy on Write

En els sistemes que hem vist fins ara, hi ha un únic vector d'i-nodes en una posició coneguda, cadascun associat a un fitxer diferent, els quals apunten a les dades.



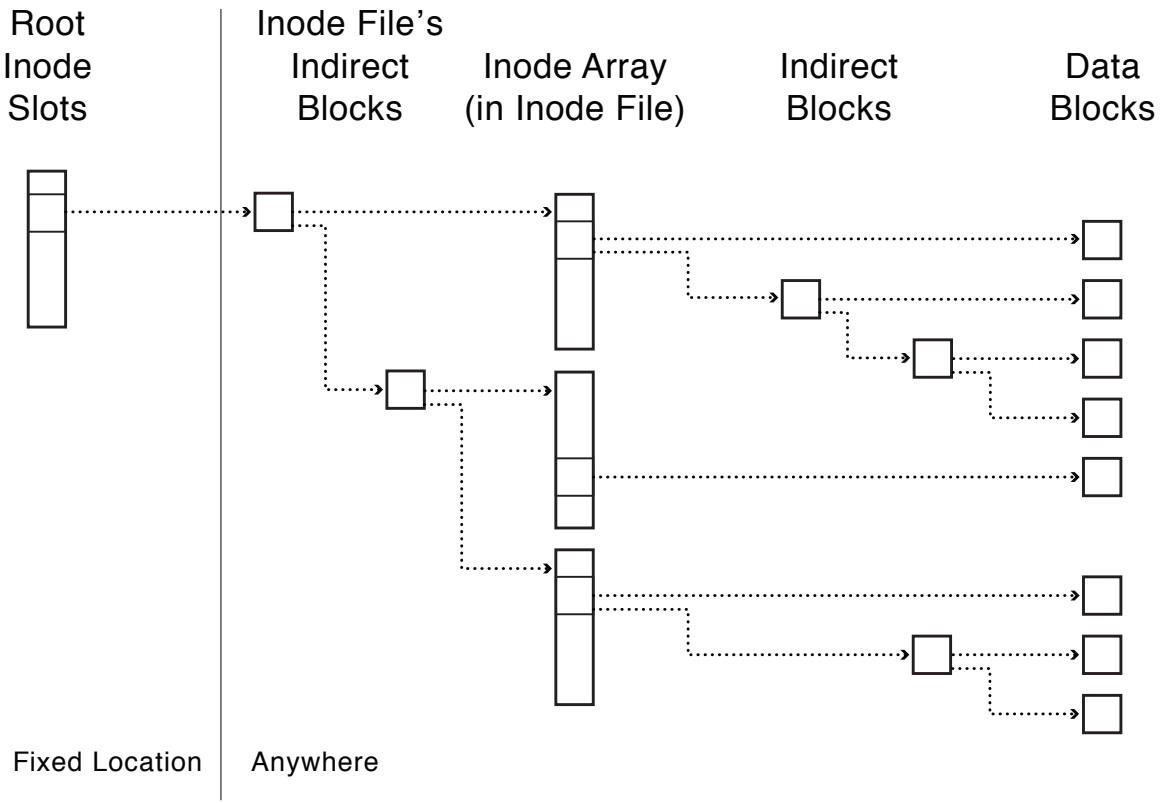
COW: Copy on Write

En els sistemes COW el vector d'i-nodes s'emmagatzema a un fitxer (blocs del disc). Hi ha un "meta-vector", situat en una posició fixa, que apunta als fitxers d'i-nodes. Compareu els dos dibuixos!



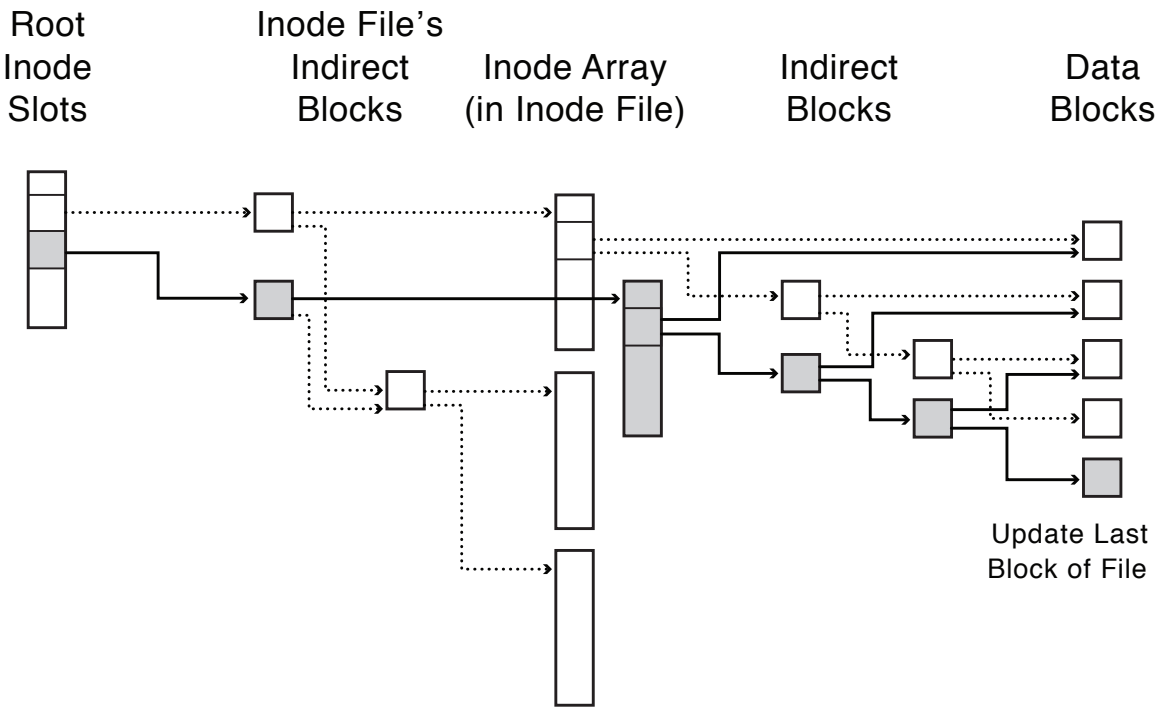
COW: Copy on Write

El COW es pot interpretar com múltiples sistemes de fitxers. La idea és fer servir el COW per poder tenir seguretat a les dades.



COW: Copy on Write

Idea idealitzada: en actualitzar un fitxer de dades s'hi associa un nou i-node a la següent posició del "meta-vector". Al nou i-node s'hi emmagatzemen les dades actualitzades del fitxer.
Cada fitxer de dades té associat múltiples i-nodes, on cada i-node es correspon a una versió "actualitzada" de l'anterior "versió" del fitxer!



COW: Copy on Write

Intuïu les avantatges d'aquest tipus de sistemes de fitxers?

- Cada i-node d'un fitxer té associat un checksum i número de versió. En cas que el sistema falli en emmagatzemar les dades a disc, el sistema operatiu pot recuperar la darrera versió del fitxer accedint al i-node anterior del fitxer.
- En aquest disseny, l'arrel del sistema és un "meta-vector" circular que permet emmagatzemar múltiples versions del sistema de fitxers.
- Es pot rotar el "meta-vector" cada setmana, cada mes o en determinats moments definits pel sistema operatiu.

COW: Copy on Write

Pel cas particular del ZFS

- En el cas del ZFS, la mida d'aquest "meta-vector" (anomenat *uberblock*) és de 256 posicions. En engegar l'ordinador, el ZFS escaneja aquest array i comprova quin té associat la darrera versió i un checksum vàlid. De forma periòdica, el ZFS rota aquest array per tenir emmagatzemades versions anteriors del sistema de fitxers.
- El ZFS s'ho manega perquè en "actualitzar" un fitxer aquesta es realitzi en una sola escriptura contigua a disc, operació més eficient que fer accessos aleatoris a disc (cosa que es fa en els sistemes de fitxers anteriors).
- Aquest sistema requereix emmagatzemar més dades a disc que en els sistemes de fitxers vistos fins ara. Suport per a discos molt grans, de PetaBytes.

Sistemes de fitxers

Ja hem acabat! Hem pogut veure l'evolució històrica dels sistemes de fitxers

- FAT
- FFS
- NTFS
- COW

Aquests sistemes han estat dissenyats i/o optimitzats per a discos magnètics amb parts mecàniques. Però... i els discos d'estat sòlid?

Discos d'estat sòlid

Discos d'estat sòlid

- No tenen parts mòbils. Això fa que el seu rendiment sigui molt superior que els discos magnètics, especialment per accés aleatori.
- Sistemes de fitxers com FAT, ext3, ext4, NTFS utilitzats als discos magnètics no són adequats per aquests tipus de dispositius. Per què?

Discos d'estat sòlid

Per augmentar la durabilitat dels discos d'estat sòlid

- Es poden fer servir sistemes de fitxers específics per a aquests tipus de dispositius. Són els anomenats log-structured filesystems. Els sistemes COW són adequats per aquests tipus de dispositius.
- A l'actualitat els discos d'estat sòlid poden fer servir els sistemes de fitxers habituals (FAT, FFS, NTFS, ...) perquè utilitzen una tècnica anomenada "wear leveling". Sabeu el que és? Busqueu...

Conclusions

Hem vist l'evolució històrica dels sistemes de fitxers

- La capacitat d'emmagatzematge ha crescut pràcticament de forma exponencial i això ha fet necessari re-dissenyar cada cop aquests sistemes per tal d'incloure-hi les funcionalitats necessàries.
- Tot sistema de fitxers té limitacions i avantatges. Feu servir el sistema de fitxers que us faci falta...
- Els sistemes de fitxers continuen evolucionant...
  ara cap a sistemes distribuïts! Us imagineu com funcionen?