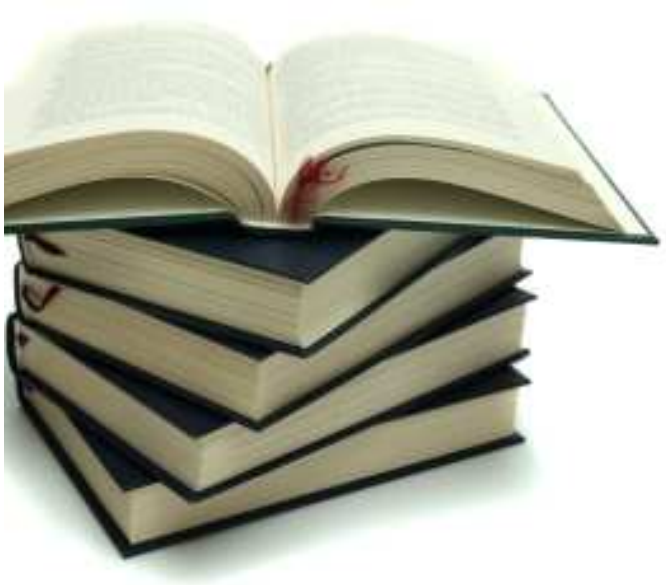
Cursul 10



10

Gestiunea fisierelor

7 mai 2009

Test Curs 9

- Pe un sistem Linux cu split 3G/1G, la ce adresă fizică este mapată adresa virtuală 0xC0001000? Dar 0xF8001000?
- Care este complexitatea (e.g. O(1), O(n), etc.) alocării unei pagini? Dar a unei structuri task_struct?
- Pentru folosirea unei zone de memorie dinamic alocată, în context întrerupere, ce primitive de alocare pot fi folosite în Linux? Dar în Windows?

Gestiunea fişierelor

- Gestiunea fişierelor – noțiuni generale
- Gestiunea fişierelor în Linux
  - VFS
  - Inode cache, dcache, page cache
  - Buffer cache
- Gestiunea sistemelor de fişiere în Windows
  - Cache manager
  - Fast I/O

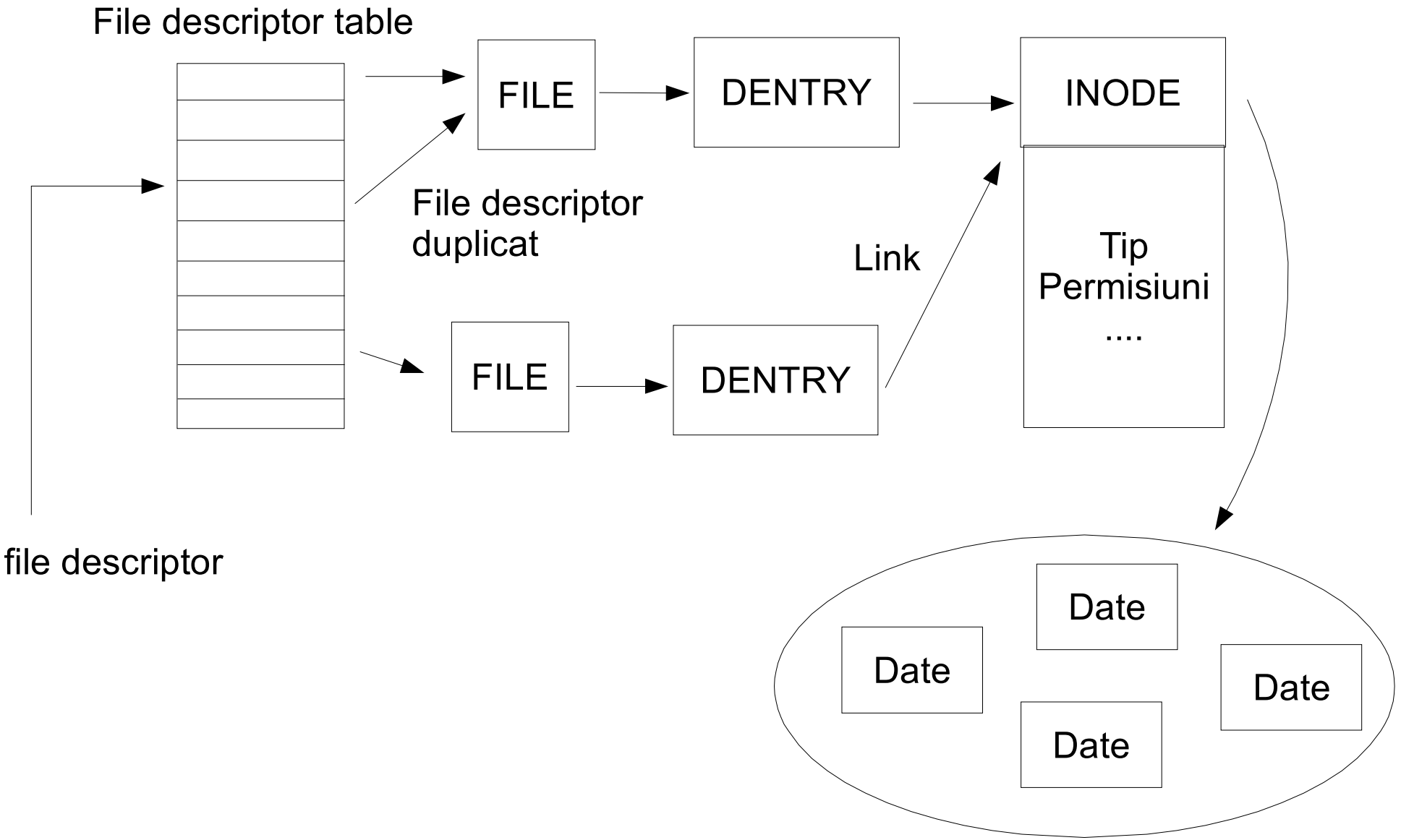
Bibliografie

- LKD: capitolele 12, 15
- WI: capitolele 11, 12

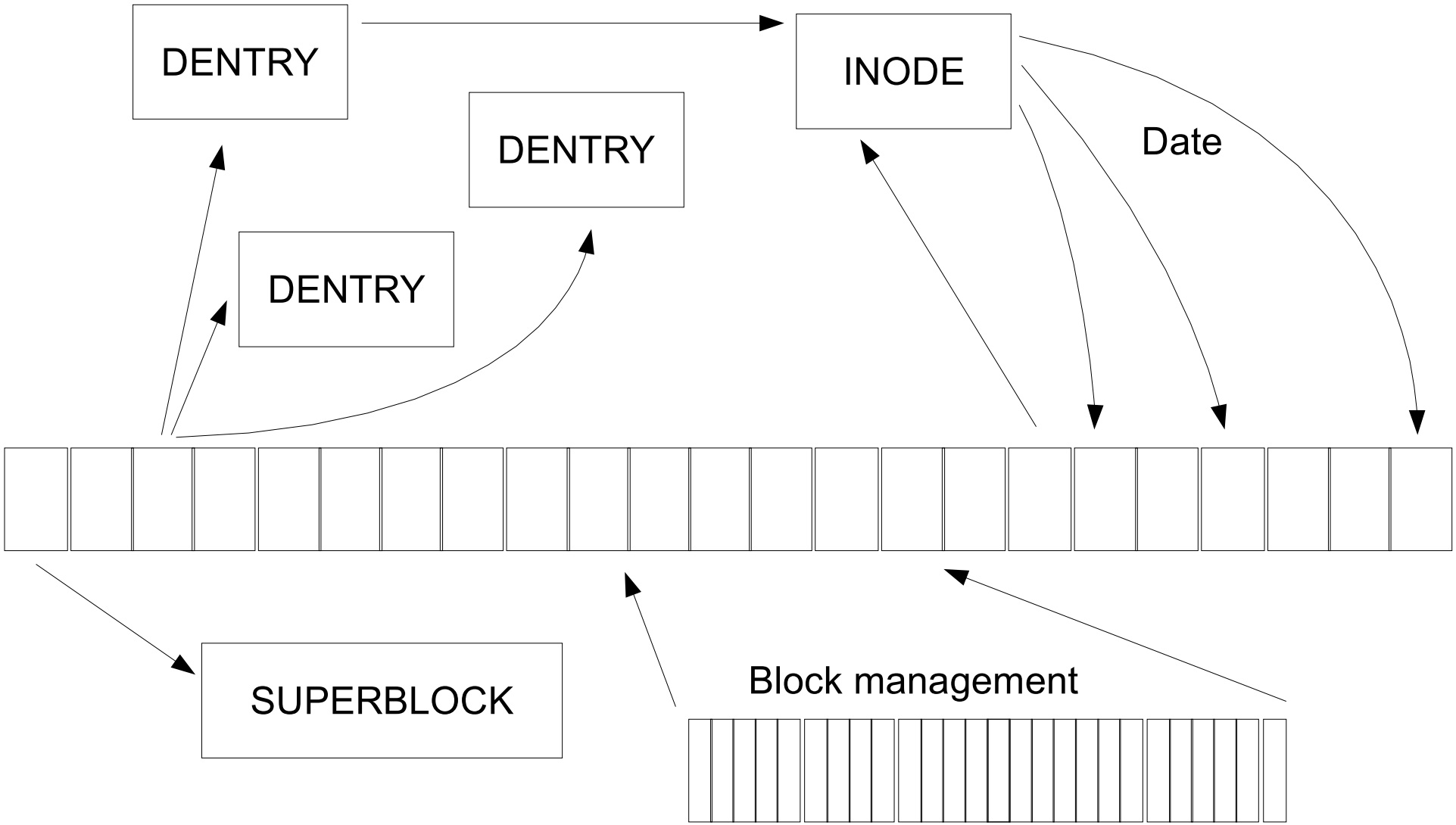
Entități fundamentale

- Superblock – conține informații despre sistemul de fişiere (dimensiunea blocului, inode-ul rădăcină); există atât pe disc cât şi în memorie
- File – structura internă SO ce descrie un fişier deschis; există doar în memorie
- Inode (FCB - file control block) – entitatea ce identifică un fişier în mod unic; există atât pe disc cât şi în memorie
- Dentry- asociază un nume cu un fişier; există atât pe disc cât şi în memorie

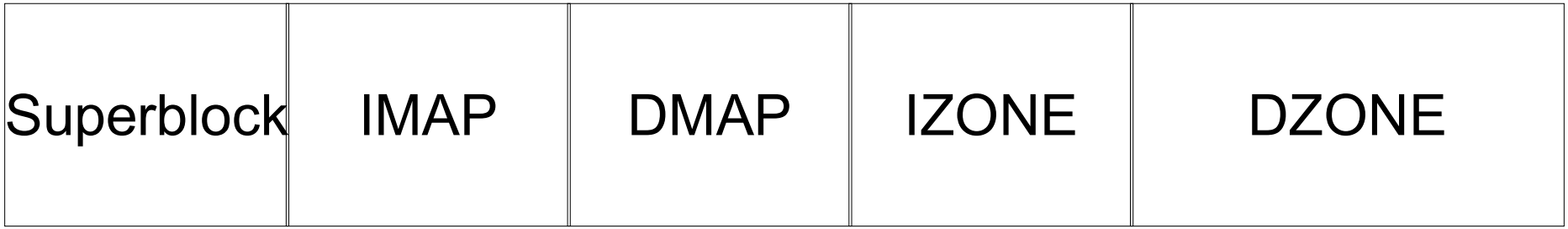
Entități fundamentale – în memorie



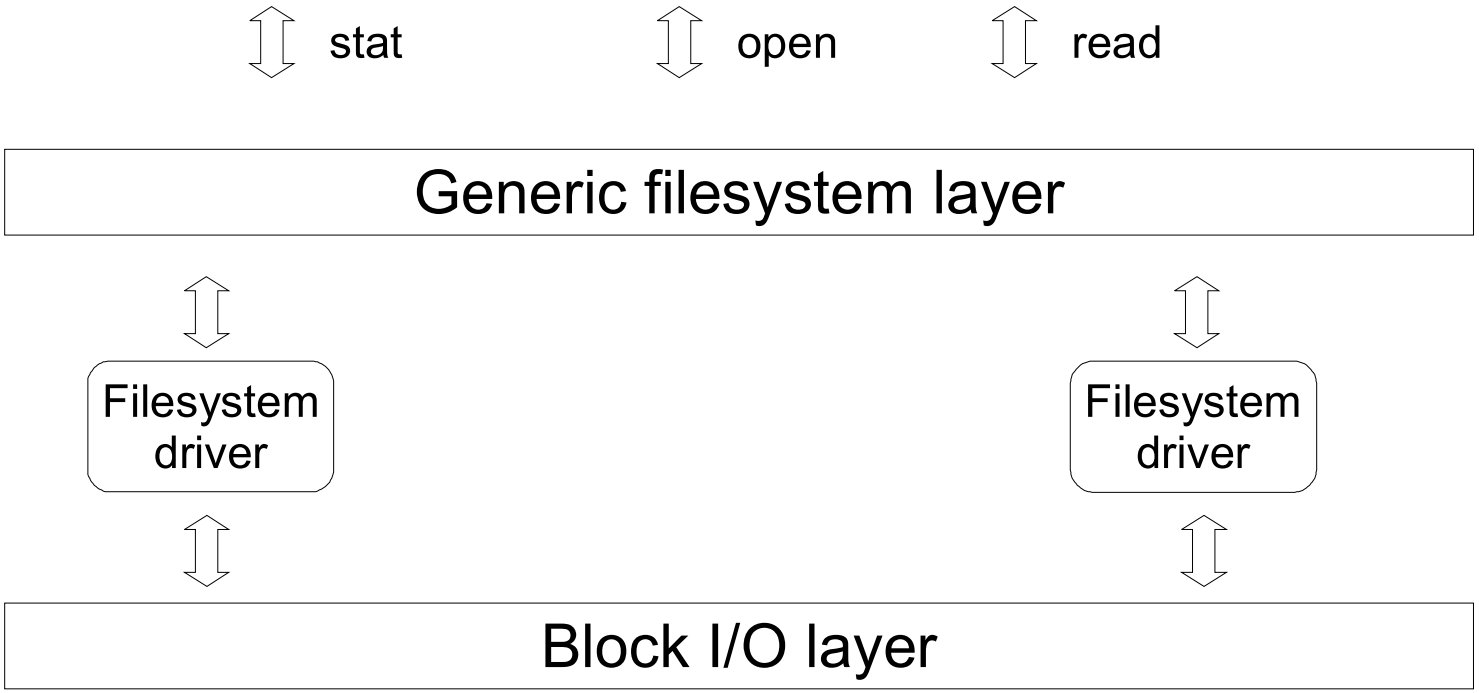
Entități fundamentale – pe disk



Exemplu: PITIX



Gestiunea sistemelor de fişiere în sistemul de operare



Operații fundamentale

- Montarea
- Deschiderea unui fişier
- Determinarea atributelor fişierului
- Citirea de date din fişier
- Scrierea de date în fişier
- Închiderea unui fişier
- Crearea unui fişier
- Ştergerea unui fişier

Montarea sistemului de fişiere

- (sau înregistrarea lui în cadrul sistemului)
- Intrare: un disk (partiție)
- Ieşire: un DENTRY către directorul rădăcină
- Operații: verificare partiție, determinare diverşi parametri, determinare inode-ului rădăcină
- Exemplu PITIX:
  - Se verifică MAGIC-ul
  - Se determină dimensiunea blocului
  - Se citeşte inode-ul rădăcină şi se crează dentry-ul asociat

Deschiderea unui fişier

- Intrare: o cale
- Ieşiere: un file descriptor
- Operații:
  - Determinarea sistemului de fişiere
  - Pentru fiecare nume din cale, determinarea inode-ului asociat cu numele
  - Odată găsit inode-ul ultimului nume din cale se crează structura FILE şi se alocă o intrare în tabela de descriptori de fişier

Determinarea atributelor fişierului

- Intrare: file descriptor
- Ieşiere: atributele fişierului
- Operații:
  - Se accesează inode-ul (file->dentry->inode)
  - Se citesc atributele din inode

Citirea de date din fişier

- Intrare: file descriptor, offset, length
- Ieşire: date
- Operații:
  - Se accesează inode-ul (file->dentry->inode)
  - Se determină blocurile de date
  - Se copiează datele de pe disk în memorie şi sunt apoi trimise utilizatorului

Scrierea de date în fişier

- Intrare: file descriptor, offset, length, date
- Ieşire:
- Operații:
  - Se accează inode-ul
  - Se aloca unul sau mai multe noi blocuri pe disc – se caută blocuri libere, se marchează ca fiind ocupate
  - Se adaugă blocurile alocate la inode
  - Se copieaza datele de la user in bufere interne şi apoi se scriu pe disk

Închiderea unui fişier

- Intrare: file descriptor
- Ieşiere:
- Operații:
  - Se decrementează reference counter-ul structurii FILE
  - Dacă referecen counter-ului structurii FILE ajunge la 0, se şterge structura
  - Se setează pe NULL intrarea din tabela de descriptori de fişier

Directoare

Directoarele sun fişiere ce conțin o înşiruire (vector, câteodată arbore) de DENTRY-uri.

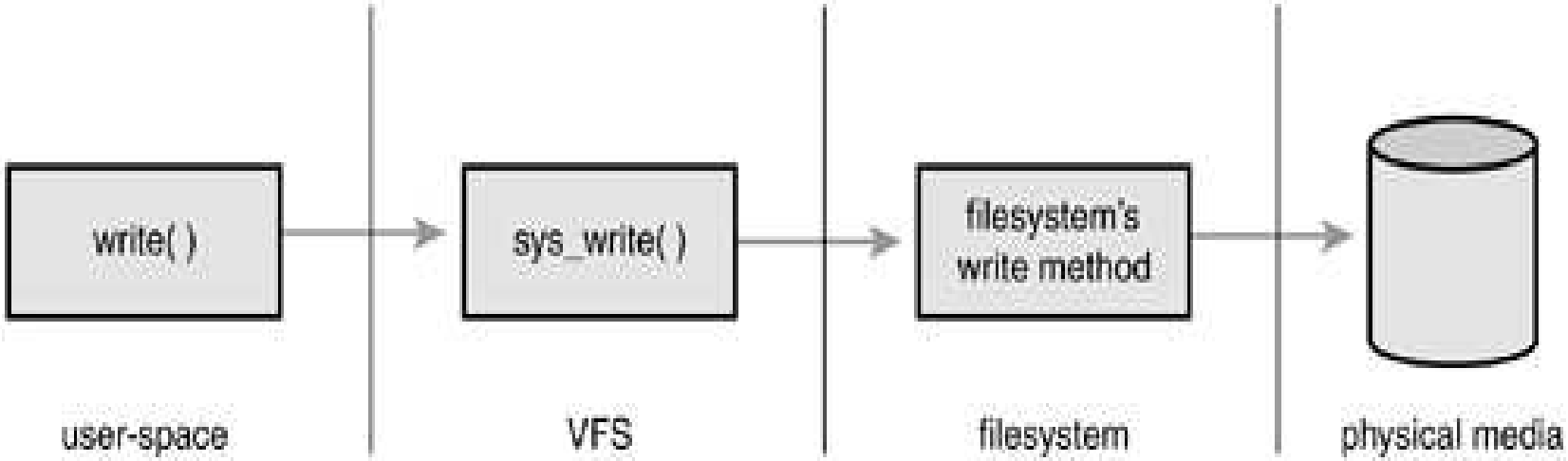
Crearea unui fişier

- Intrare: o cale
- Ieşire:
- Operații:
  - Se determină inode-ul directorului în care trebuie adăugat fişierul
  - Se citesc blocurile de date
  - Se inserează un nou dentry
  - Se scriu pe disc blocurile de date modificate

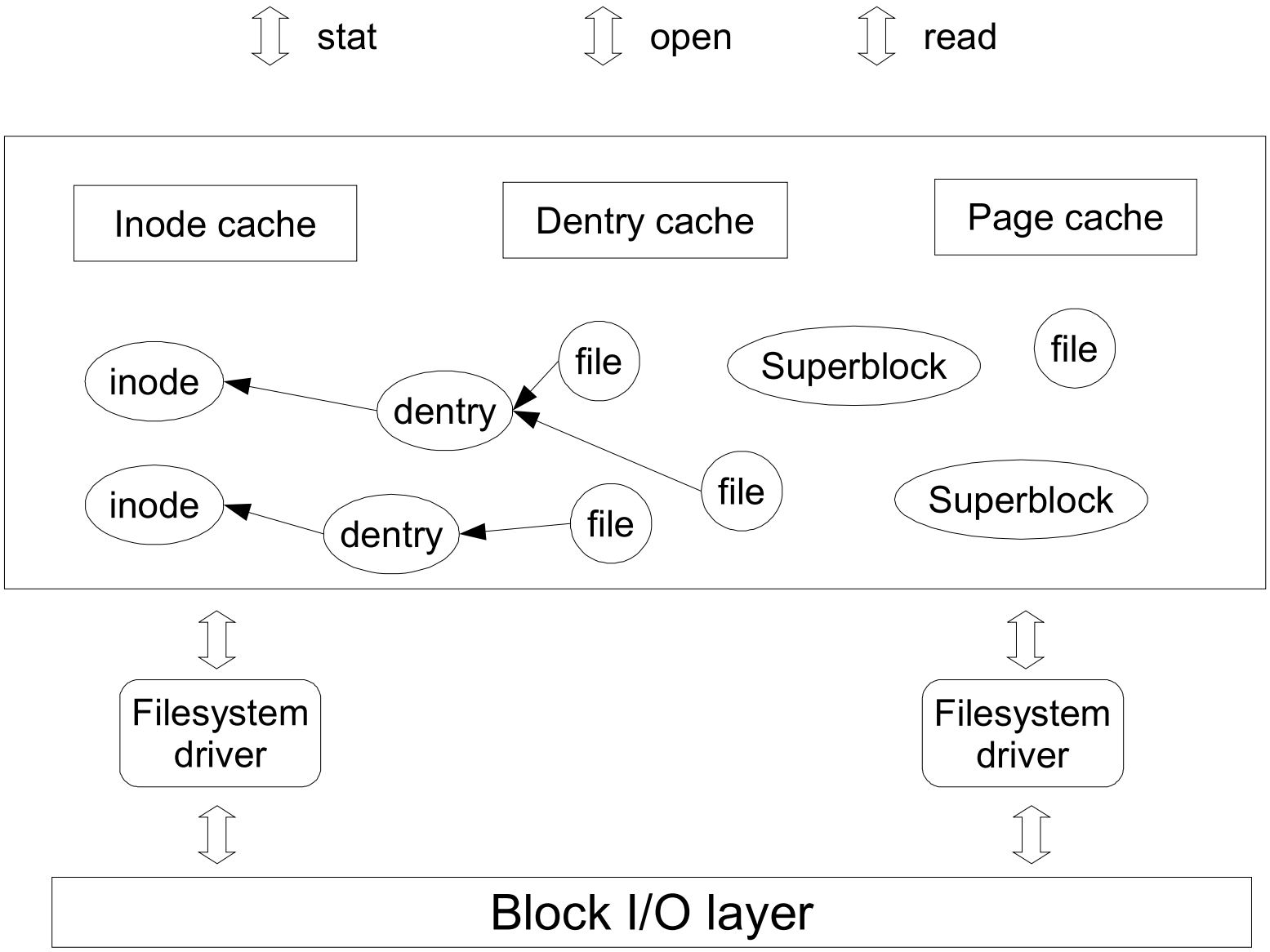
Ştergerea unui fişier

- Intrare: o cale
- Ieşire:
- Operații:
  - Se determină inode-ul directorului în care trebuie adăugat fişierul
  - Se citesc blocurile de date
  - Se caută şi se sterge DENTRY-ul cu numele fişierului (caz special: link-uri)
  - Se scriu pe disc blocurile de date modificate

Gestiunea sistemelor de fişiere în Linux



VFS


stat
open
read
Inode cache
Dentry cache
Page cache
inode
dentry
file
Superblock
file
inode
dentry
file
file
Superblock
Filesystem driver
Filesystem driver
Block I/O layer

Operații superblock

- fill_super
- put_super
- write_super
- read_inode
- write_inode
- delete_inode
- clear_inode
- Statfs
- remount_fs

Operații inode

- Create
- Lookup
- Link
- Unlink
- Symlink
- Mkdir
- rmdir

- Rename
- Readlink
- follow_link
- put_link
- Truncate
- ...

Inode cache

- Operațiile de citire a inode-urilor de pe disc sunt costisitoare
- Un inode citit se va menține în memorie (până când apar condiții de low memory)
- Căutarea inode-ului se face cu ajutorul unei tabele hash
- Funcția de hash (sb, inode_number)

Dentry

- Stări:
  - Used – d_inode este valid şi obiectul dentry este folosit
  - Unused – d_inode este valid dar obiectul dentry nu este folosit
  - Negative – d_inode nu este valid (NULL); fie inode-ul nu a fost încă încărcat fie a fost şters
- Dentry cache
  - Lista de dentry-uri folosite (dentry->d_state == used)
  - Lista celor mai recent folosite dentry-uri (sortată după timpul de access)
  - Un hash table pentru a elimina parcurgerea arborelui

Page cache

- Menține în memorie datele citite de pe disc
- Se foloseşte pentru:
  - read/write
  - Mmap
- Cache-ul se face la nivel de fişier şi nu de block device -> este nevoie de o infrastructură pentru translatarea offset-urilor din fişiere în blocuri pe disc
- Cache-ul se bazează pe un radix tree

address_space

- Structura care este folosită de page-cache pentru a translata adrese
- Datorită acestei abordări se poate face caching nu doar pe mapări de fişiere
- Operații address_space:
  - write_page, read_page
  - Bmap
  - direct_IO
  - ...

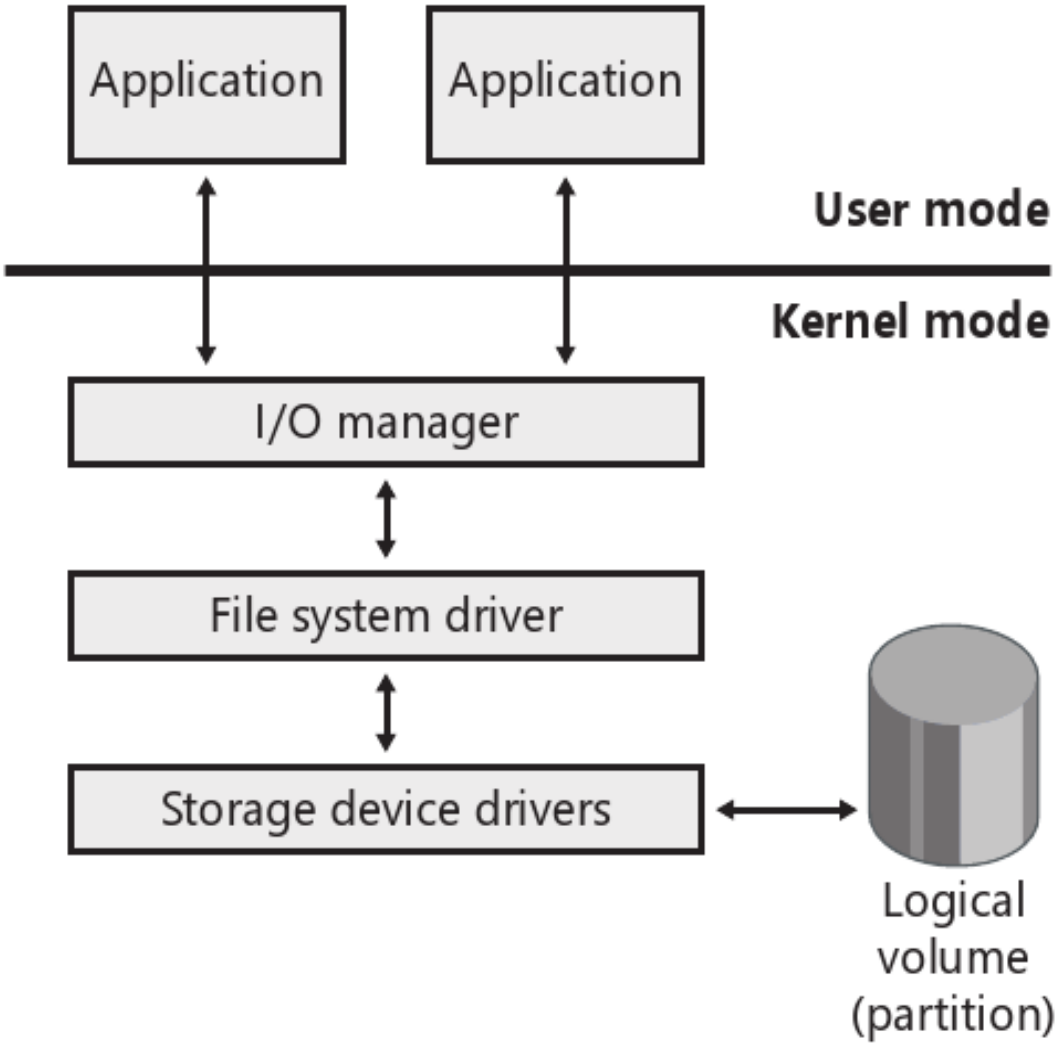
Citirea de date din fişier

- generic_file_read – implementare generică
  - Se verifică dacă pagina nu există în page cache
  - Daca există se copiază datele din pache cache în buffer-ul utilizatorului
  - Dacă nu există se apelază a_ops->readpage()
- block_read_full_page – implementare generică, foloseşte a_ops->bmap()

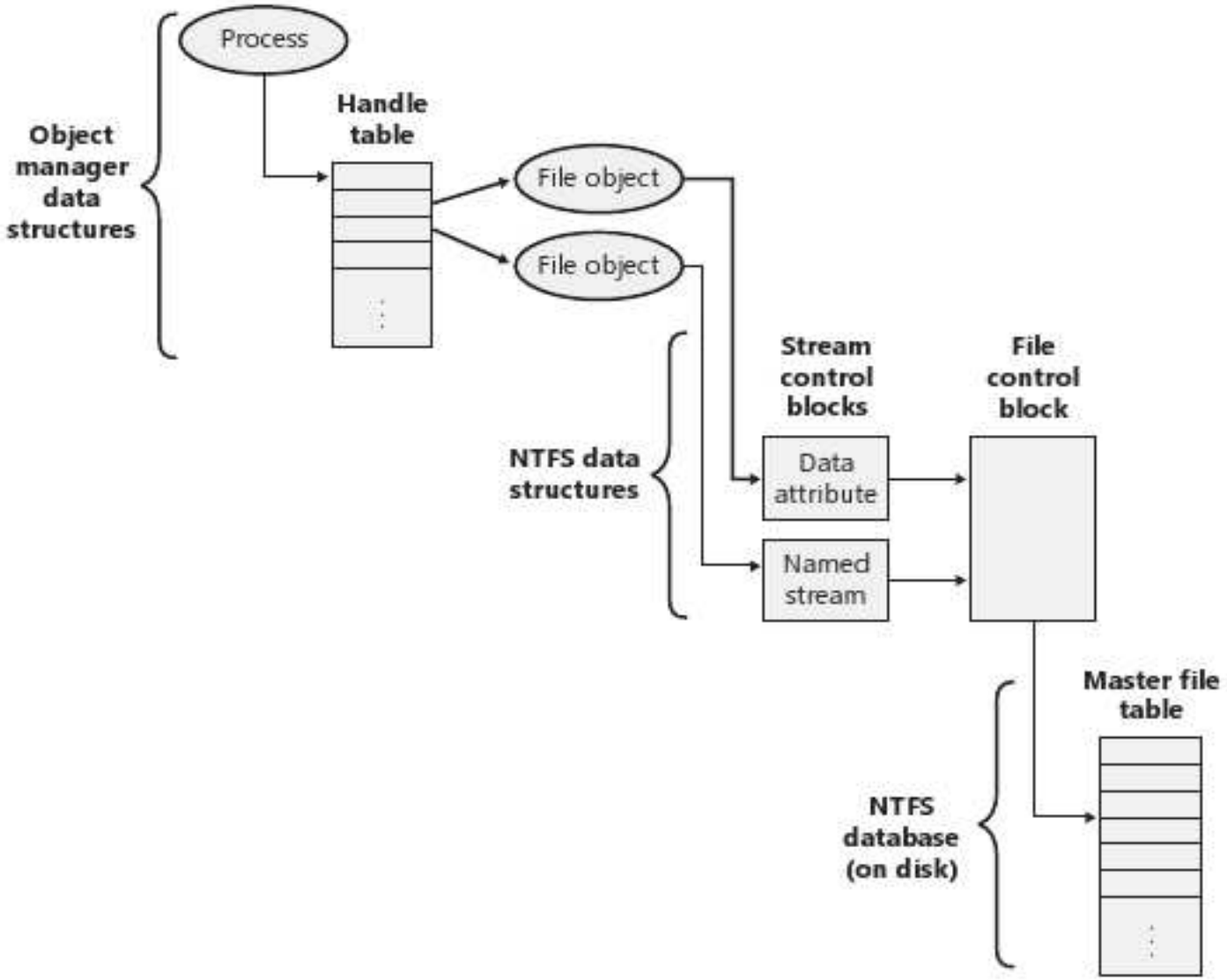
Buffer cache

- De la versiunea 2.4 nu mai există un buffer cache separat
  - S-a eliminat problema double buffering
  - S-au eliminat problemele de sincronizare
- buffer_head-urile – structurile care fac maparea dintre un buffer în memorie şi un block pe disc folosesc memorie alocată de page cache

Gestiunea sistemelor de fişiere în Windows



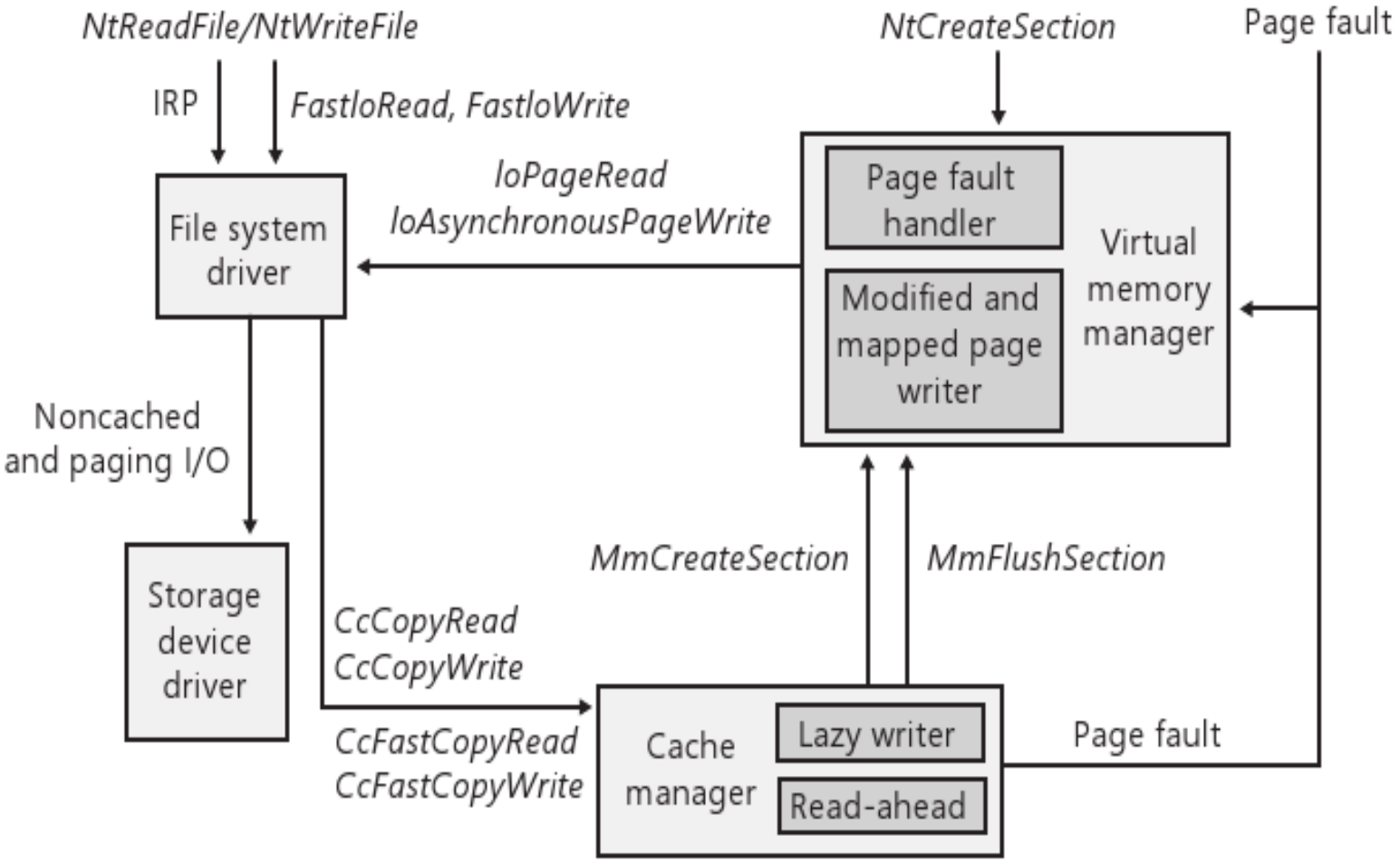
Entități fundamentale



Analogie UNIX- Windows

- Superblock – Volume Parameter Block
- Inode – File Control Block
- File – File Object
- Dentry -

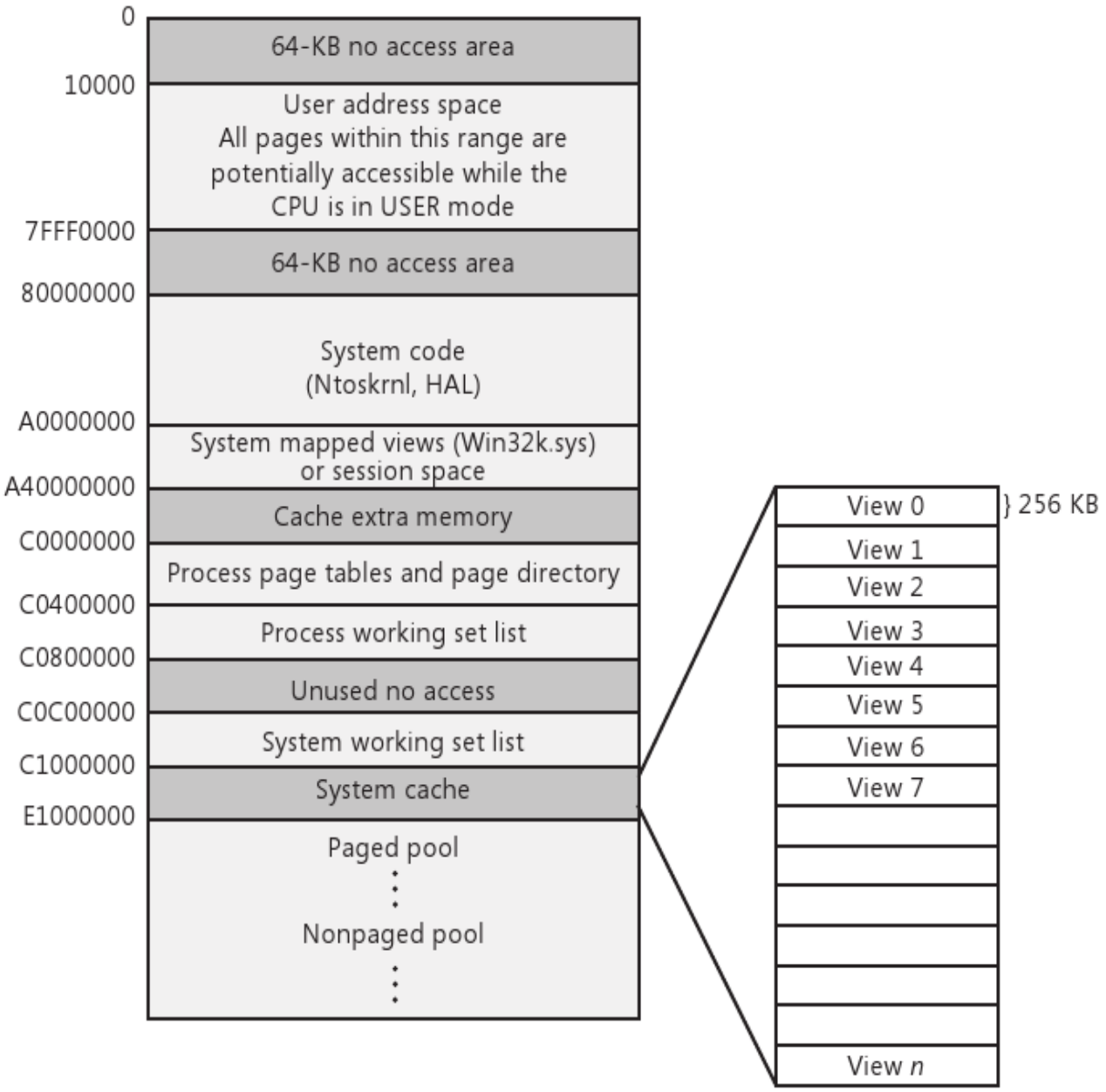
Interacțiune FSD – MM – Cache manager



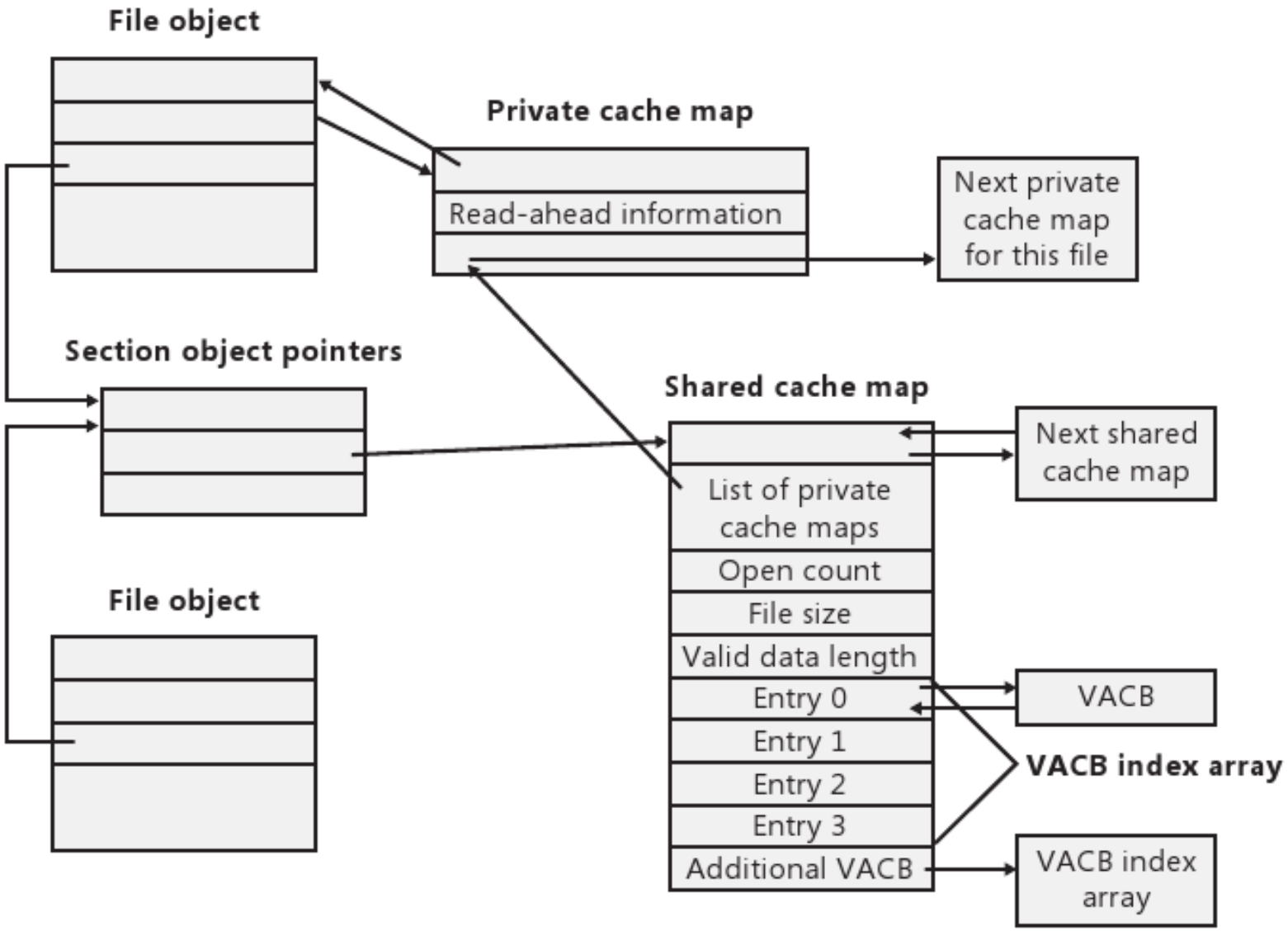
Tratarea IRP-urilor de scriere / citire în FSD

- Dacă FILE_FLAG_NO_BUFFERING nu a fost setat:
  - se copiează datele în buferul utilizatorul cu ajutorul cache managerului: CcCopyRead
- Altfel:
  - Se crează noi IRP-uri ce vor fi pasate storage device-ului

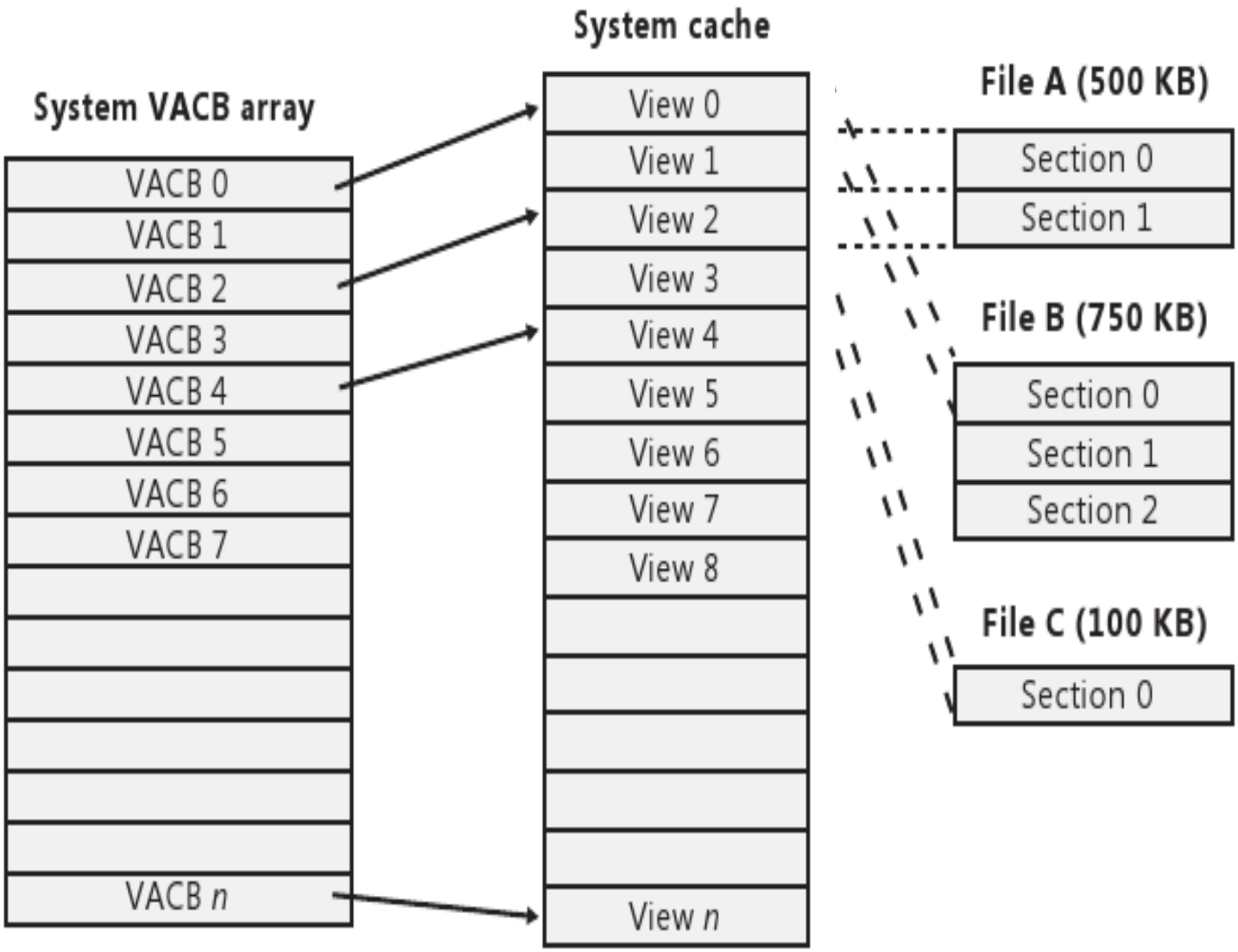
System cache address space



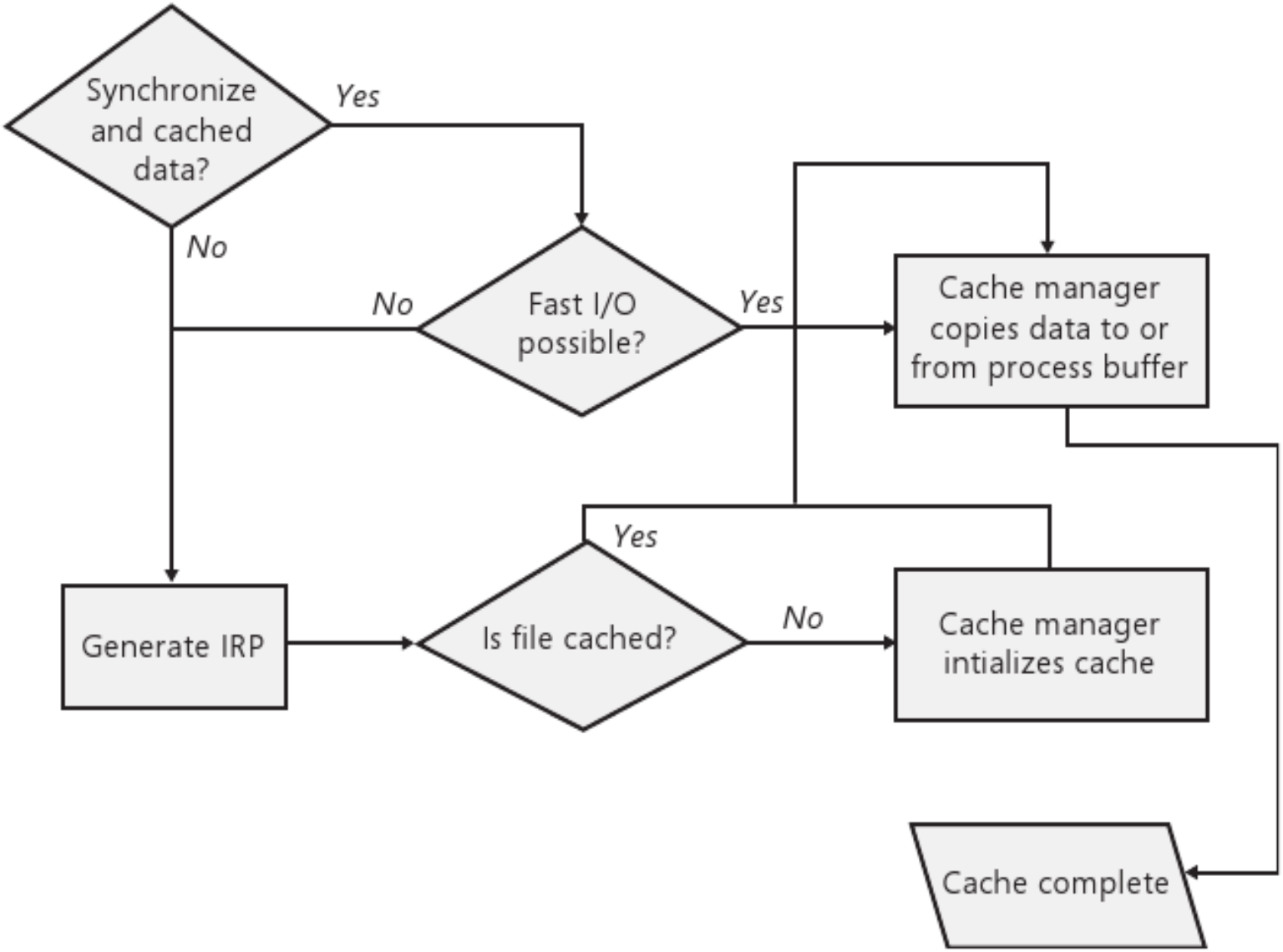
Maparea fişierelor în system cache



Maparea fişierelor în system cache (2)



Fast I/O


Synchronize and cached data?
Yes
No
Fast I/O possible?
No
Yes
Cache manager copies data to or from process buffer
Generate IRP
Is file cached?
Yes
No
Cache manager intializes cache
Cache complete

Intrebări

?