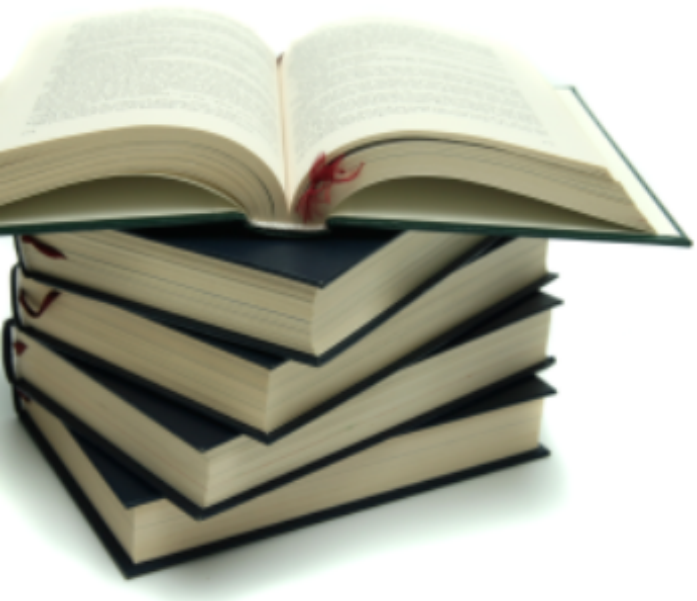
page crunch

Cursul 2



2

Sistemul de fișiere – interfața de user space

20 februarie 2012 - 26 febuarie 2012

Suport curs 2

- OSC
  - Capitolul 10 – File-System Interface
  - Capitolul 11, Secțiunea 11.1 – File-System Structure
- MOS
  - Capitolul 6 – File System Implementation
    - secțiunile 1 și 2
- LSP – Capitolul 2 (parte programatică)
- WSP – Capitolul 2 (parte programatică)

Cuprins

- Sisteme de fișiere: definiții, concepte
- Fișiere. Atribute ale fișierelor
- Operații cu fișiere
- API pentru lucrul cu fișiere
- Directoare. Operații cu directoare
- Tipuri speciale de fișiere
- Montarea sistemelor de fișiere

De ce avem nevoie de sisteme de fișiere?

- memoria RAM are dimensiuni reduse și este volatilă
- este necesar un suport persistent (e.g. HDD) care:
  - are dimensiune mare
  - stochează aplicațiile (executabilele) și datele durabil
- provocare: suportul persistent este în practică mult mai lent decât memoria RAM

probleme:

cum putem organiza informația stocată persistent,

cum o putem accesa rapid și eficient?

Sistemul de fișiere este

- un mod de organizare a informației (persistente):
  - unitatea de bază este fișierul
  - mapează datele pe suportul fizic
  - oferă scalabilitate și acces facil la fișiere
- din punct de vedere al utilizatorului
  - o interfață de acces la informație formată din directoare și fișiere (în general ierarhică)
- din punct de vedere al sistemului de operare
  - structuri de date și algoritmi pentru organizarea informației pe disc

Exemple de SF

- ext3
- ReiserFS
- HFS
- NTFS
- FAT32
- ISO9660
- UDF

Ierarhia de apeluri/interfețe

utilizator

aplicație

metainformație

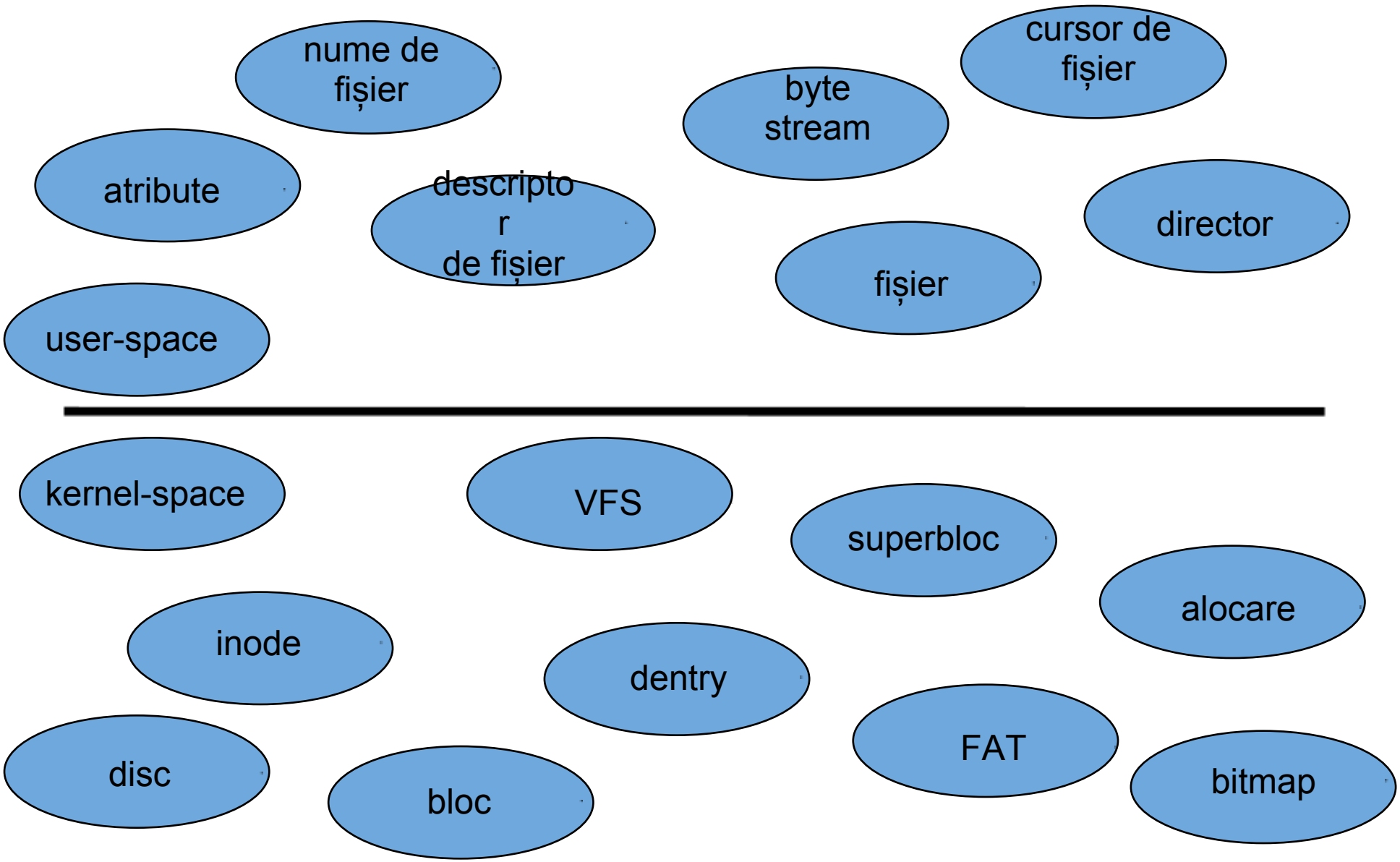
translatare blocuri logice – blocuri fizice

implementare SF

device driver

dispozitiv fizic

Sistemul de fișiere – vedere de ansamblu



Fișierul

- **unitatea de bază** a sistemului de fișiere
- în lumea SO există două abstractizări fundamentale
  - **fișierul** – abstractizează informația/datele
  - **procesul** – abstractizează efectuarea de acțiuni
- în sistemele Unix-like, totul este un fișier
  - de fapt, aproape totul (Plan9 este foarte aproape)
  - aceleași atribute și tipuri de operații cu fișiere sunt corelate cu diverse interfețe ale sistemului de operare

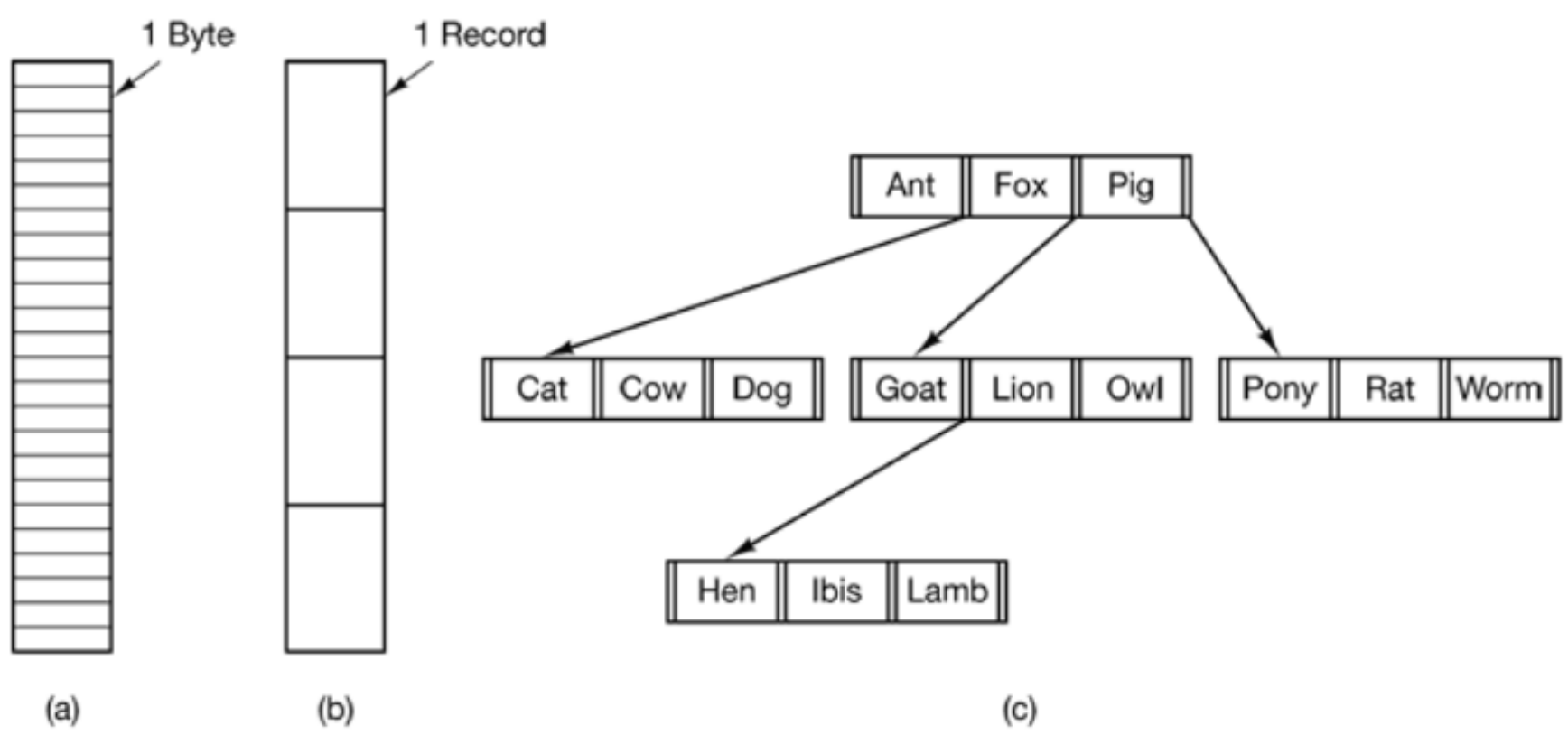
Tipuri de fișiere

- fișiere simple – regular files
- directoare – directories
- legături (simbolice) – hard/soft links
- dispozitive tip caracter – char devices
- dispozitive tip bloc – block devices
- pipes/FIFOS
- socketi UNIX
- cum aflați tipul unui fișier în UNIX?

Fișiere simple

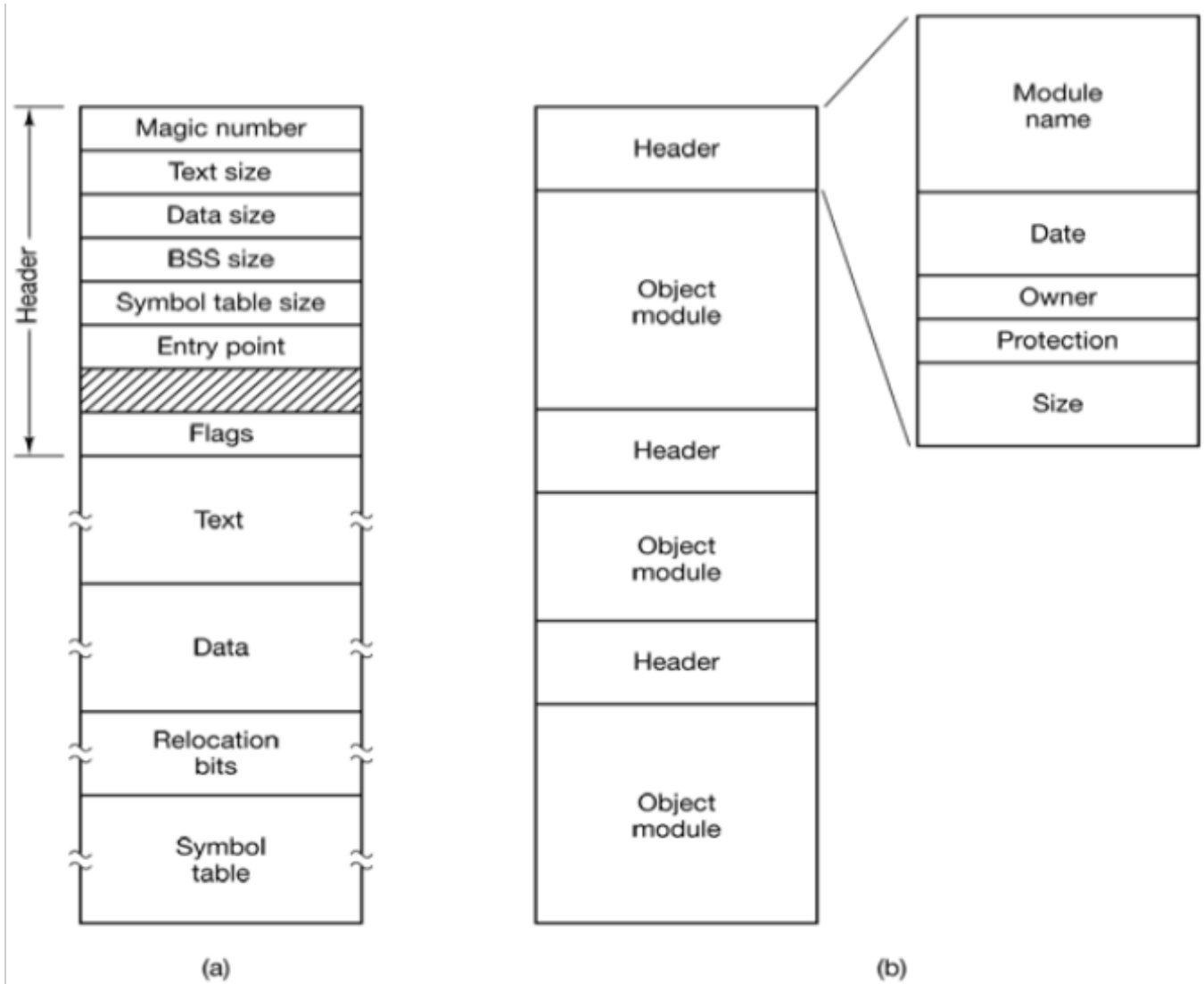
- binare, text
- extensie
- organizare/structură
  - record
  - șir de octeți
- atribute
  - drepturi
  - dimensiune
  - utilizator
  - timestamps

Organizare fișier



a) byte-stream b) record file c) tree

Fișiere binare



a) fișier executabil [1] b) arhivă

Operații cu fișiere

- creare
- deschidere
- citire
- scriere
- poziționare în cadrul fișierului
- trunchiere
- închidere
- ștergere [2]

Creare

- shell

```
touch /path/to/file
```

  - comenzile care scriu într-un fișier îl și creează

- ISO/ANSI C

```
FILE *f = fopen ("/path/to/file", "rw");
```

- Unix

```
int fd = open ("/path/to/file", O_CREAT | O_EXCL, 0644);
```

- Windows

```
HANDLE fileHandle = CreateFile ("/path/to/file", ..., CREATE_NEW);
```

Deschidere

- ISO/ANSI C

```
FILE *f = fopen("/path/to/file", "rt");
```

- Unix

```
int fd = open("/path/to/file", O_RDONLY);
```

- Windows

```
HANDLE f = CreateFile("/path/to/file", ..., OPEN_EXISTING);
```

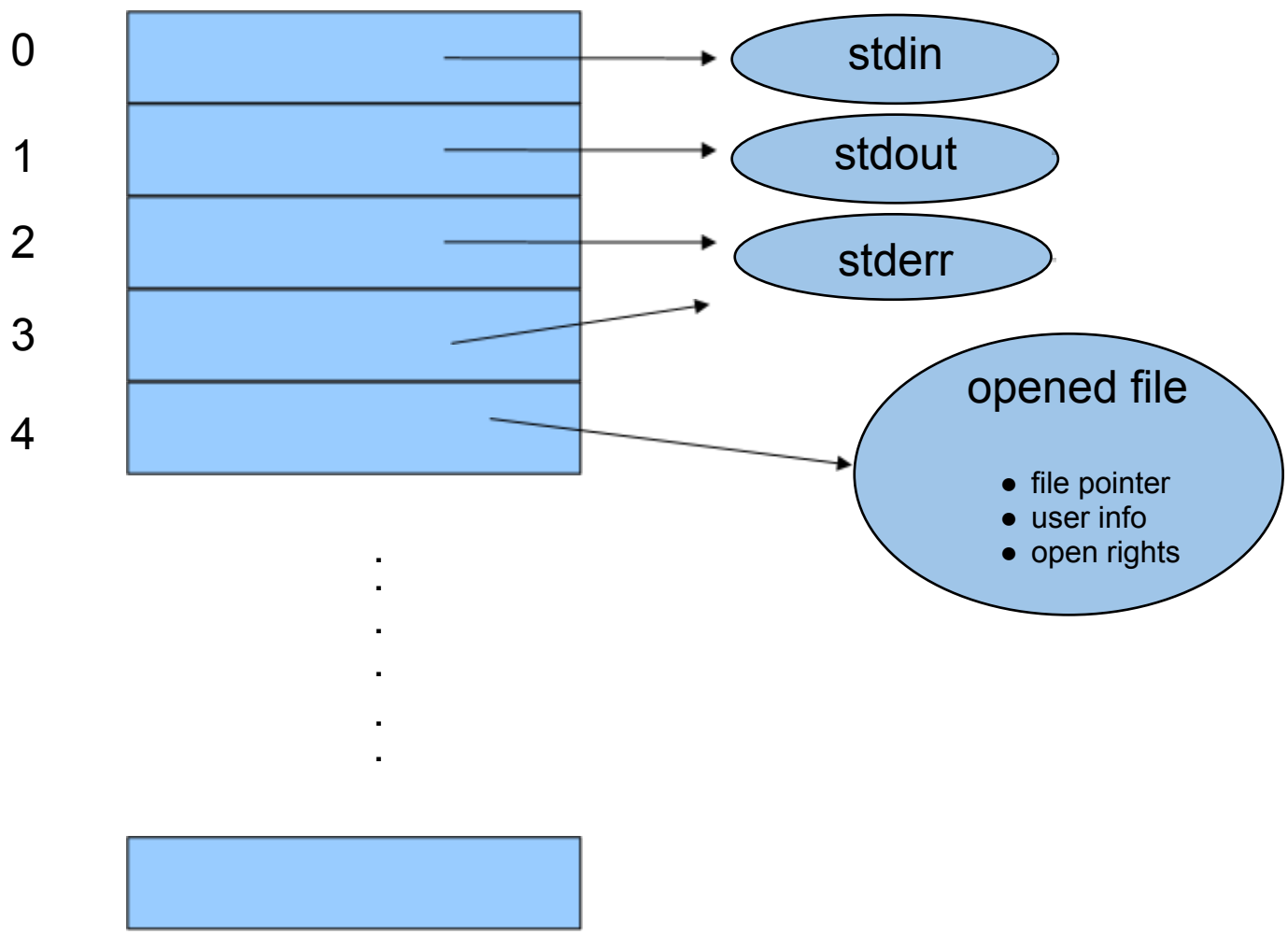
Atribute fișier deschis

- identificare
  - handle, descriptor
  - de ce nu nume?
- file pointer, cursor de fișier
- drepturi de deschidere
- contor de utilizare (file-open count) [3]

Descriptor de fișier

- un întreg ce identifică o instanță de fișier deschisă în cadrul unui proces
- Care este asocierea între un descriptor și un fișier? (unu la unu, mai multe la mai multe?)
- fiecare proces are o tabelă de descriptori de fișier [4]
- 0, 1, 2 sunt descriptori speciali – stdin, stdout, stderr
- FILE (ISO C)
- HANDLE (Win API)

Tabela de descriptori [5]



FILE (ISO C)

```
/* /usr/include/libio.h */
struct _IO_FILE {
    [...]
    int _fileno;
    [...]
}

/* /usr/include/stdio.h */
typedef struct _IO_FILE FILE;
```

Citire

- stocarea informației într-un buffer
- avansul cursorului de fișier
- ISO C

```
n_recs = fread(buf, RECSZ, N_RECS, fin);
    /* se folosește cu feof(), ferror() */
```

- Unix

```
n_read = read(fd, buf, BUFSIZ); [6]
    /* se folosește obligatoriu într-un while */
```

- Windows

```
ReadFile(fHandle, buf, bytesToRead, &bytesRead, NULL);
```

Scriere

- scrierea informației dintr-un buffer
- avansul cursorului de fișier
- ISO C

```
n_recs = fwrite(buf, RECSZ, N_RECS, fin);
    /* scrie garantat, sau eroare de scriere */
```

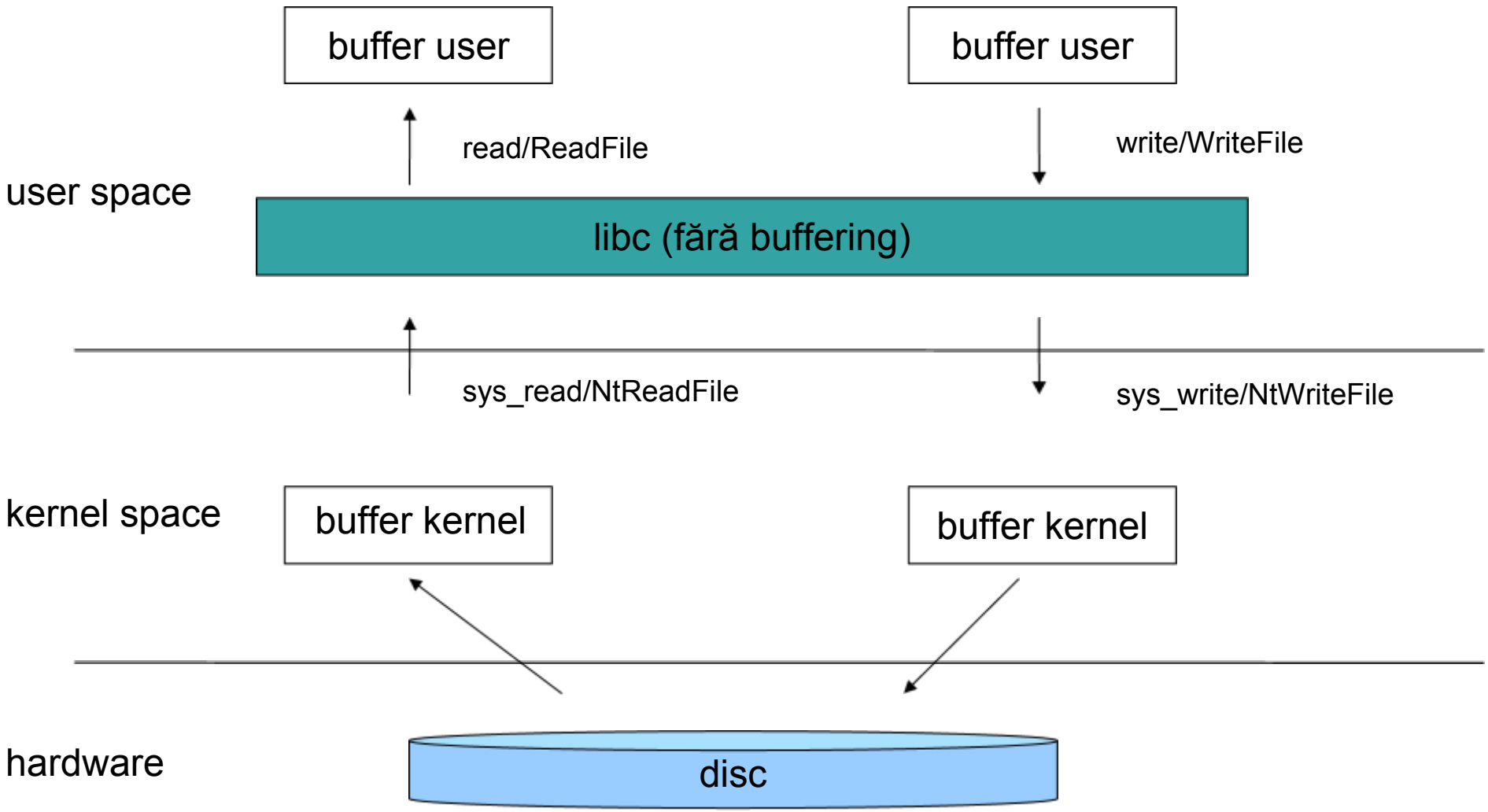
- Unix

```
n_written = write(fd, buf, BUFSIZ); [7]
    /* se folosește obligatoriu într-un while */
```

- Windows

```
WriteFile(fHandle,buf,bytesToWrite,&bytesWritten,NULL);
```

Citire/scriere – fluxul datelor



Cursor de fișier

- modificare la read și write
- inițializare la deschidere
- modificare:
- ISO C

```
fseek(f, offset, SEEK_SET);
```

- Unix

```
lseek(fd, offset, SEEK_END);
```

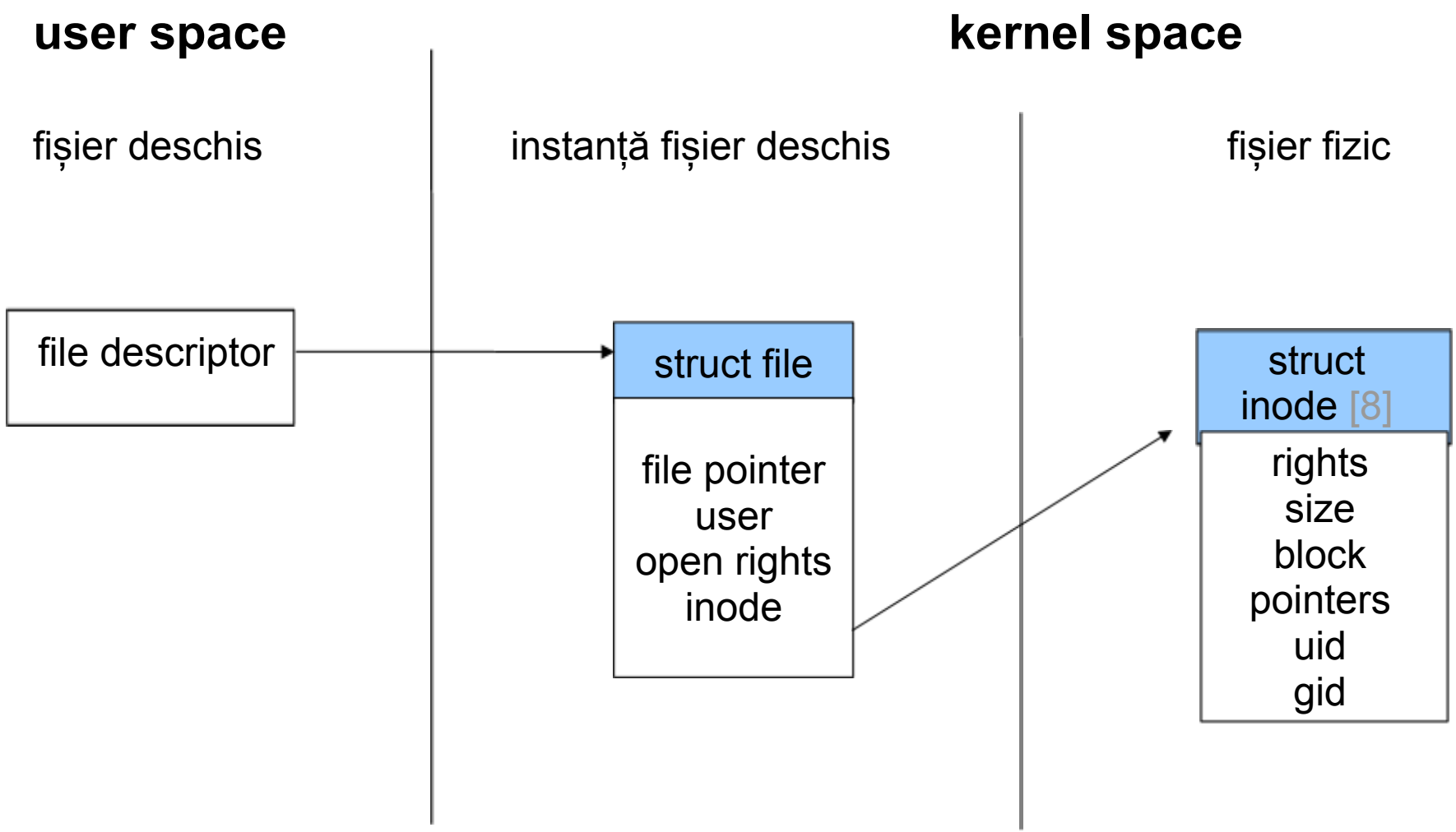
- Windows

```
SetFilePointer(hFile, distanceToMove, NULL, FILE_BEGIN);
```

Tipuri de date

- utilizatorul recunoaște fișierul prin nume
- programele folosesc un descriptor
- descriptorul indexează tabela de descriptori
- un element al tabelei este un pointer la o structură dinamică (instanță de fișier deschis)
- structura dinamică are referință la structura ce descrie fișierul fizic
- ce asociere există între cele trei structuri? (file descriptor, structură dinamică, fișier fizic)

Tipuri de date specifice

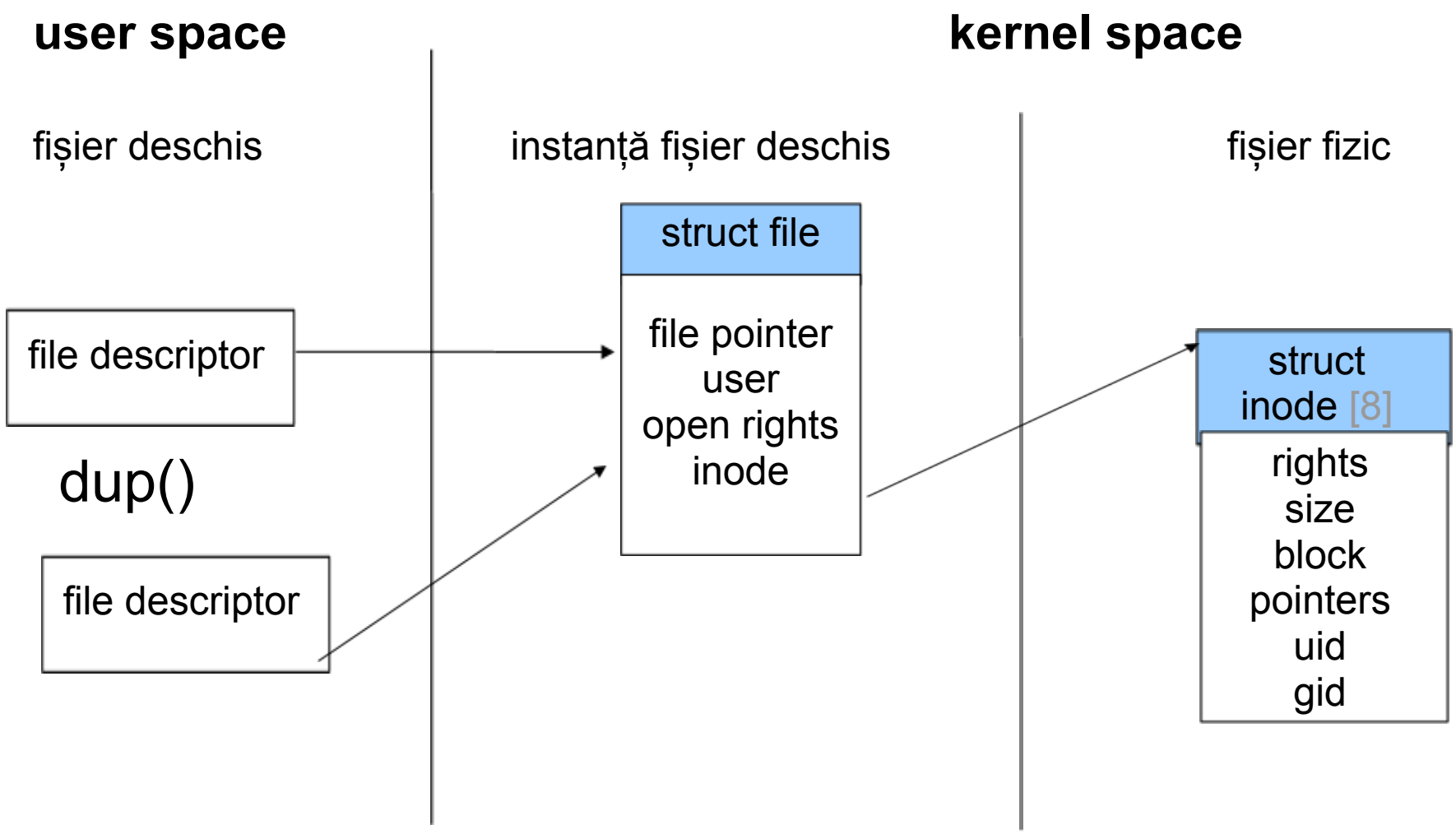


Duplicare descriptor

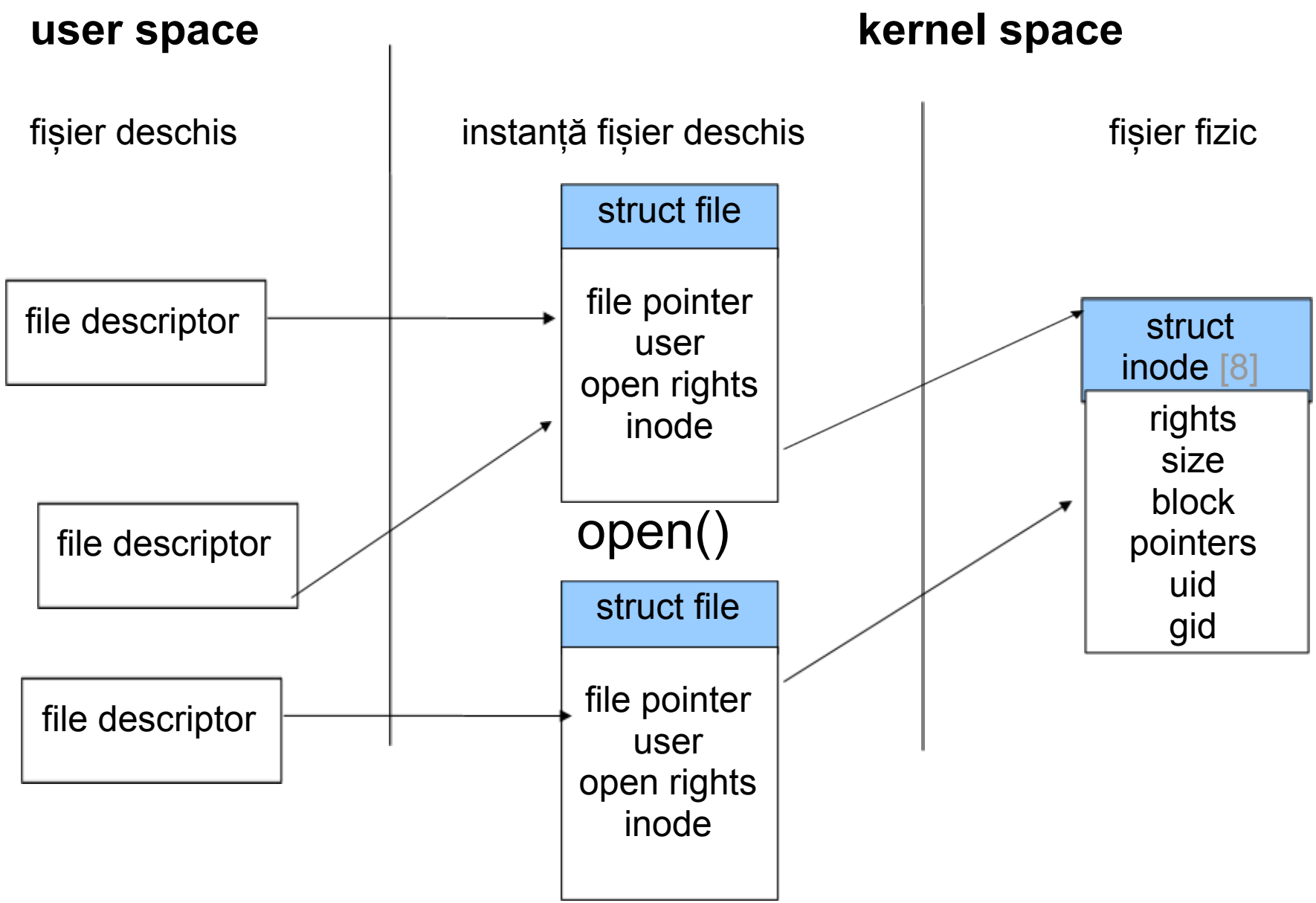
- duplicarea unui descriptor în alt descriptor
- cei doi descriptori lucrează asupra aceluiași fișier

```
newfd = dup(oldfd);

dup2(oldfd, newfd);


DuplicateHandle(...);

SetStdHandle(...);
```

Tipuri de date specifice



Tipuri de date specifice



Trunchiere fișiere

- eliminarea conținutului fișierului
- cursorul de fișier este poziționat pe 0

- un fișier poate fi deschis și trunchiat

```
open("/path/to/file", O_RDWR | O_TRUNC);
truncate("/path/to/file", size);
```

- un fișier se poate trunchia după deschidere

```
ftruncate(fd, size);
SetEndOfFile(hFile);
```

- dupa trunchiere atributele unui fisier raman neschimbate (singurul atribut care se modifica este dimensiunea)

Închidere

- se pierde accesul la fișier
- este eliminată intrarea din tabela de descriptori

```
close(fd);

CloseHandle(fHandle);
```

Acces simultan la un fișier

- Ce se întâmplă când mai multe programe deschid același fișier?
  - pentru citire: fiecare citește independent fișierul (fiecare are un cursor diferit)
  - un program citește, altul scrie - nu se știe exact ce o să citească primul
  - mai multe scriu - nu se știe ordinea în care datele vor fi scrise
- Pentru a media accesul simultan la fișiere se folosesc lock-uri
  - vor fi descrise pe larg în cursul 10

Accesul concurent la descriptori duplicați

- nucleul menține o singură structură de date pentru ambii descriptori
  - un singur cursor de fișier
- operațiile sunt serializate folosind această structură
  - rezultatele operațiilor sunt consistente
  - doar ca nu știm sigur ordinea în care au fost executate
- pentru a asigura o ordine trebuie să sincronizăm accesul la fișier folosind comunicație între procese/thread-uri
  - mai multe detalii în cursul 5

Redirectare

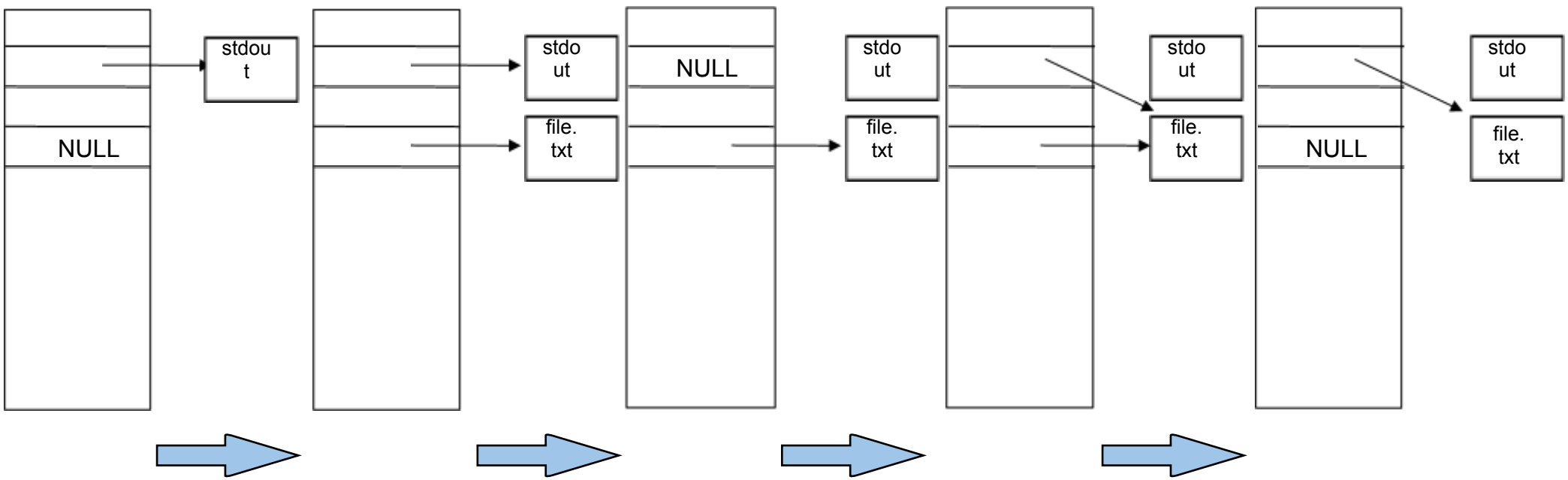
- în shell

```
echo "mesaj" > file_out.txt
grep "cuvant" < file_in.txt
```

- programatic

```
fd = open ("file_out");
close (STDOUT_FILENO);
dup (fd);
close (fd);
```

Redirectare (2)



Ștergere fișier

- eliminarea fișierului din sistemul de fișiere
- shell

```
rm file.txt
```

- ISO/C

```
remove("file.txt");
```

- UNIX

```
unlink("file.txt");
```

- Windows

```
DeleteFile("file.txt");
```

Directoare

- colecție de intrări în sistemul de fișiere
  - folder/catalog
- de ce directoare?
  - organizarea fișierelor
  - acces facil al utilizatorului
- organizarea cea mai întâlnită este cea ierarhică
  - generalizat – graf orientat aciclic (DAG)

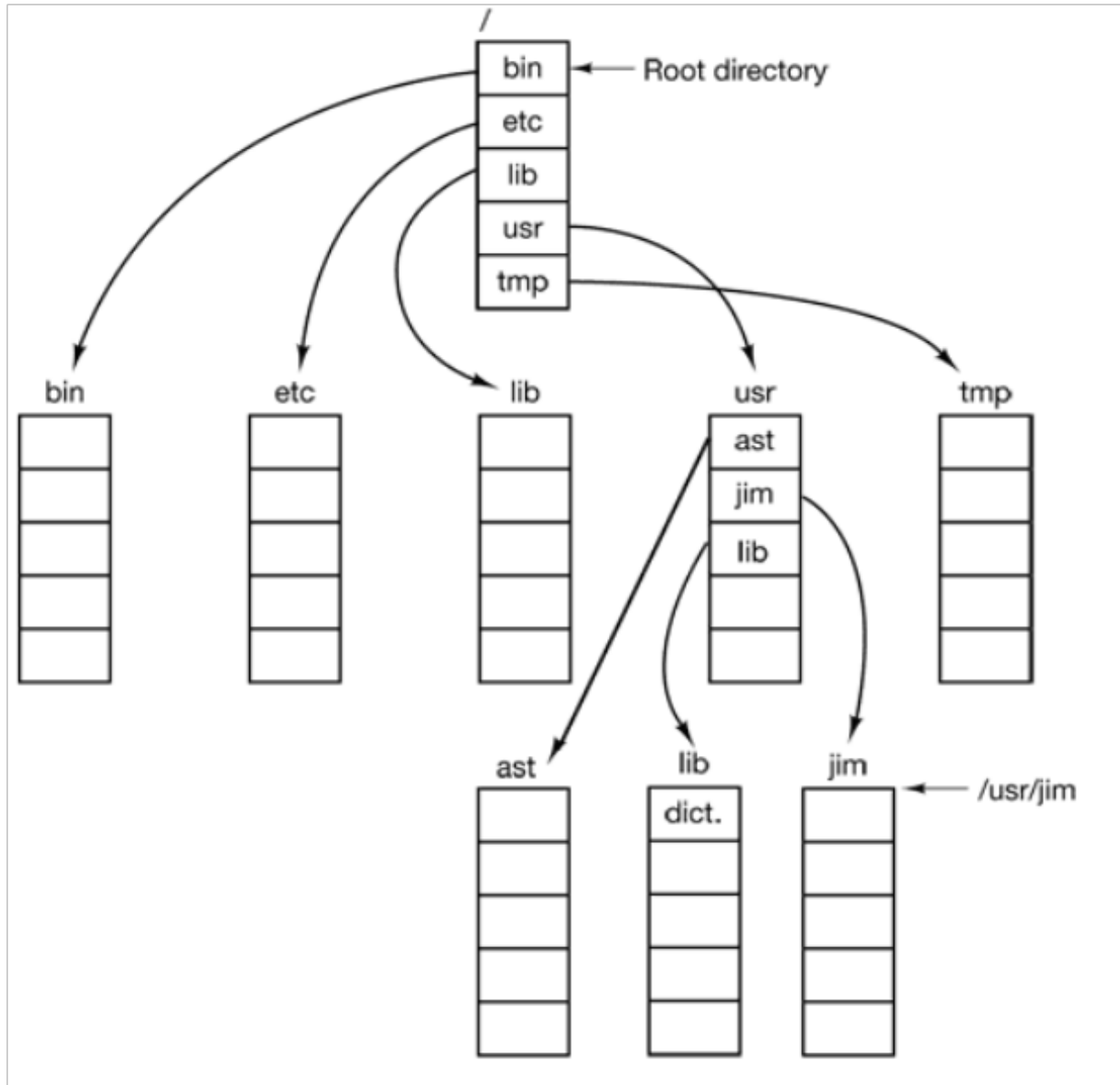
Cale (path)

- înlănțuire de intrări în sistemul de fișiere
- orice sistem de fișiere are un director rădăcină (/ pe Unix)
- separator de cale (/ pe Unix, \ pe Windows)
- cale absolută
- cale relativă
- intrările . și .. sunt speciale

  . (dot) - referință către directorul curent

  .. (dot-dot) - referință către directorul părinte

Organizarea directoarelor



Operații cu directoare

- crearea/ștergerea unui director (mkdir, rmdir)
- crearea unei intrări (touch, mkdir, ln, mknod)
- eliminarea unei intrări (rm, rmdir, unlink)
- listarea conținutului (ls)
- parcurgerea unei căi (cd)
- redenumirea unei intrări (mv)

Operații cu directoare (2)

- UNIX
  - DIR *, struct dirent
  - mkdir, chdir, rmdir
  - opendir, closedir, readdir, rewinddir

- Windows
  - HANDLE
  - CreateDirectory, RemoveDirectory, SetCurrentDirectory
  - FindFirstFile, FindNextFile, FindClose

Fișiere speciale

- legături (links) (ln, link)
- char device (mknod, mknod)
- block device (mknod, mknod)
- FIFO (mknod/mkfifo, mknod/mkfifo)
- socketi UNIX (N/A, socket/mknod)

Legături (link-uri)

- folosite pentru referirea unui fișier deja existent
- symlink (soft link)
- hard link
- soft link pot referi intrări din alt sistem de fișiere

```
$ ls -l /usr/bin/cc

lrwxrwxrwx 1 root root 20 Oct 6 2005 /usr/bin/cc -> /etc/alternatives/cc

lrwxrwxrwx 1 root root 12 Apr 22 2007 /etc/alternatives/cc ->
/usr/bin/gcc
```

Fișiere dispozitiv

- de tip caracter (char device)
  - parcurgere secvențială
  - asociate cu dispozitive seriale, mouse, tastatură

```
$ ls -l /dev/ttyS0

crw-rw---- 1 root dialout 4, 64 Feb 26 2005 /dev/ttyS0
```

- de tip bloc (block device)
  - acces aleator
  - citire în blocuri; discuri, CD-ROM, USB Flash Drive

```
$ ls -l /dev/sda

brw-rw---- 1 root disk 8, 0 Feb 26 2005 /dev/sda
```

Socketi Unix și FIFO-uri

- socketii Unix sunt o formă de comunicare inter-proces – comunicare locală
- au asociată intrare în sistemul de fișiere

```
# ls -l /var/run/cups/cups.sock
```

srwxrwxrwx 1 root root 0 2008-03-03 11:10 /var/run/cups/cups.sock

- FIFO (pipe cu nume) sunt mecanisme de bază de comunicare inter-proces
- un proces scrie, alt proces citește

```
$ ls -l mypipe
prw-r--r-- 1 razvan razvan 0 2008-03-03 21:14 mypipe
```

Standard I/O vs. Low-Level I/O

- Standard I/O – fopen, fread, fwrite, fclose
  - buffered I/O
  - double-buffering problem
  - acces high-level pentru lucrul cu șiruri

- Low Level I/O – open/CreateFile, read/ReadFile
  - wrappere [9] peste apeluri de sistem
  - cea mai mare flexibilitate
  - nu sunt portabile

Drepturi de acces

- la crearea fișierului
- la deschiderea fișierului

- în Unix
  - user, group, others
  - read, write, execute

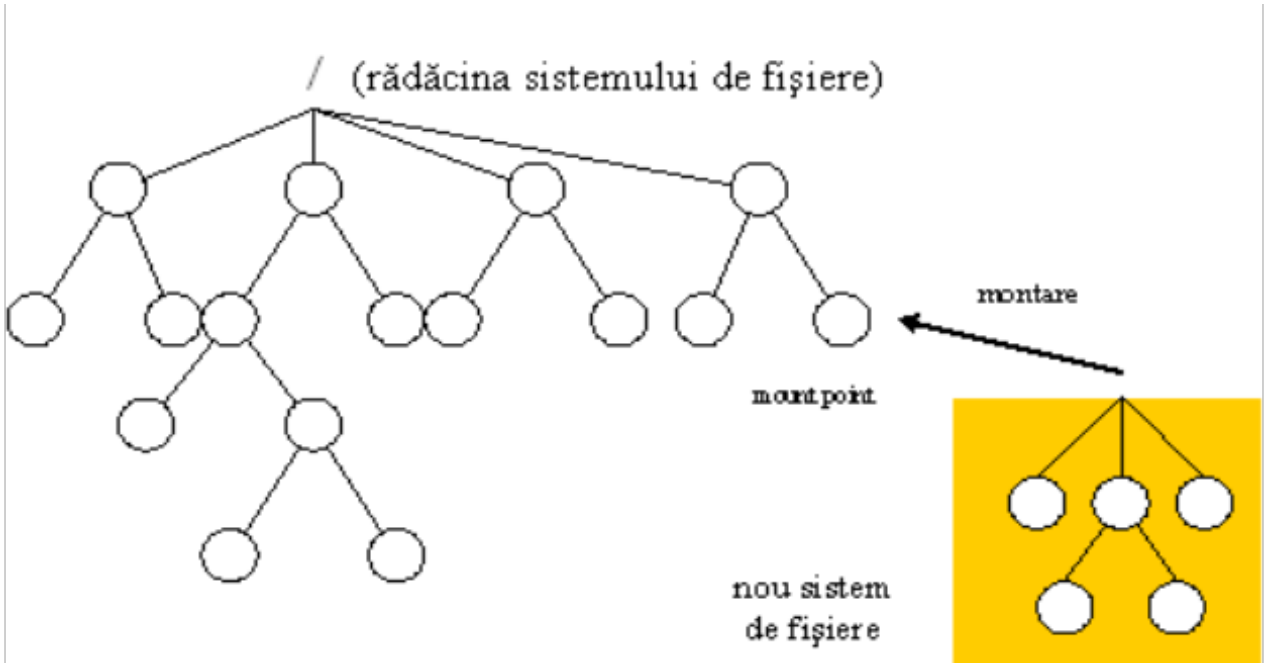
```
chmod("/path/to/file", 0644); /* drepturi în octal */
```

- în Windows
  - citire, scriere, listare, traversare, creare sub-fișier, etc.

```
SetNamedSecurityInfo(...);
```

- atașarea unui punct de intrare în sistemul local de fișiere (mount point)

```
mount("/dev/hda5", "/mnt/hda5", "ntfs", ...)
SetVolumeMountPoint("C:\\mycd\\", "D:\\");
```



SF virtuale; fișiere speciale

- sistemele de fișiere virtuale nu au un suport fizic
  - procfs montat în /proc
  - devfs montat în /dev
  - ce suport au?
- există și intrări de fișiere specializate (fișiere fără suport fizic – de obicei char device-uri)
  - **/dev/null**
  - **/dev/zero**
  - **/dev/urandom**

Cuvinte cheie

- sistem de fișiere
- fișier
- tipuri de fișiere
- byte-stream
- descriptor de fișier
- cursor de fișier
- tabelă de descriptori
- redirectare
- director
- cale
- ierarhie
- rădăcină
- fișiere speciale
- standard I/O
- low-level I/O
- drepturi de acces
- montare SF

Exercițiu 1

- Ce afișează următoarea secvență de cod?

```
        int fd1 = open("file.txt", O_CREAT | O_RDWR, 0644);
int fd2 = open("file.txt", O_CREAT | O_RDWR, 0644);
int fd3 = dup(fd1);
int off;

lseek(fd1, 100, SEEK_CUR);
lseek(fd2, 100, SEEK_CUR);
off = lseek(fd3, 100, SEEK_CUR);
printf("%d\n", off);
```

Exercițiu 2

- Care sunt asocierile corecte (unu la unu, unu la mai multe, mai multe la unul, mai multe la mai multe) între următoarele concepte?

- descriptor de fișier – fișier (pe disc)
- descriptor de fișier - cursor de fișier
- proces - descriptor de fișier
- proces - tabelă de descriptori

Exercițiu 3

- Pentru fiecare dintre opțiunile de creare/deschidere a fișierelor în Windows, care este formatul echivalent Linux?

Windows	Linux
OPEN_ALWAYS	O_CREAT
OPEN_EXISTING	O_TRUNC
CREATE_ALWAYS	O_EXCL
CREATE_NEW	
TRUNCATE_EXISTING	

Exercitiu 3 – Windows creation flags

CREATE_ALWAYS 2	Creates a new file, always. If the specified file exists and is writable, the function overwrites the file, the function succeeds, and last-error code is set to ERROR_ALREADY_EXISTS (183). If the specified file does not exist and is a valid path, a new file is created, the function succeeds, and the last-error code is set to zero. For more information, see the Remarks section of this topic.
CREATE_NEW 1	Creates a new file, only if it does not already exist. If the specified file exists, the function fails and the last-error code is set to ERROR_FILE_EXISTS (80). If the specified file does not exist and is a valid path to a writable location, a new file is created.
OPEN_ALWAYS 4	Opens a file, always. If the specified file exists, the function succeeds and the last-error code is set to ERROR_ALREADY_EXISTS (183). If the specified file does not exist and is a valid path to a writable location, the function creates a file and the last-error code is set to zero.
OPEN_EXISTING 3	Opens a file or device, only if it exists. If the specified file or device does not exist, the function fails and the last-error code is set to ERROR_FILE_NOT_FOUND (2). For more information about devices, see the Remarks section.
TRUNCATE_EXISTING 5	Opens a file and truncates it so that its size is zero bytes, only if it exists. If the specified file does not exist, the function fails and the last-error code is set to ERROR_FILE_NOT_FOUND (2). The calling process must open the file with the GENERIC_WRITE bit set as part of the *dwDesiredAccess* parameter.

Întrebări