SO Curs 2

Sistemul de fisiere – interfata de user space

Fișiere

Ce este un fișier?
Tipuri de fișiere
Fișiere
Directoare
Atribute ale fișierelor

Fisiere deschise

fișiere și fișiere deschise
procese și fișiere
handle de fișier
turaze de fișier
creare/deschidere fișier
citire/scriere
poziționare
închidere

Redirectari

Ce înseamnă redirectare?
De ce redirectare?
Implementare redirectare

Sistemul de fisiere

De ce?
Definiție
Exemple

API de lucru cu fisiere

shell
ANSI C
Unix (POSIX)
Windows (Win32)
Java
Python

Buffered I/O vs. system I/O

buffered I/O
system I/O
versiuni

Cuprins

Sistemul de fișiere
Fișiere. Atribute ale fișierelor
Operații cu fișiere
API pentru lucrul cu fișiere
Directoare

Suport de curs

Cuvinte cheie

sistem de fișiere
fișiere
bytes streams
director
fișier deschis
handle de fișier

cursor de fișier
tabela de descriptori
API pentru fișiere
redirectare
buffered I/O
system I/O

SO Curs 2

Sistemul de fisiere – interfata de user space

Fișiere

Ce este un fisier?
Tipuri de fisiere
Fisiere
Directoare
Atribute ale fisierelor

Sistemul de fisiere

De ce?
Definitie
Exemple

Fisiere deschise

fisiere si fisiere deschise
procese si fisiere
handle de fisier
cursor de fisier
creare/deschidere fisier
citire/scriere
pozitionare
inchidere

API de lucru cu fisiere

shell
ANSI C
Unix (POSIX)
Windows (Win32)
Java
Python

Redirectari

Ce inseamna redirectare?
De ce redirectare?
implementare redirectare

Buffered I/O vs. system I/O

buffered I/O
system I/O
versiuni

Cuprins

Sistemul de fisiere
Fisiere. Atributele ale fisierelor.
Operatii cu fisiere.
API pentru lucrul cu fisiere.
Directoare.

Cuvinte cheie

scrieri de fisiere
fisiere
byte-stream
director
fisier deschis
handle de fisier

cursor de fisier
tabela de descriptori
API pentru fisiere
redirectare
buffered I/O
system I/O

Suport de curs

OSC
- Capitolul 10 – File-System Interface
- Capitolul 11, sectiunea 11.1 – File-System Structure

MOS
- Capitolul 6 – File System Implementation
  - sectiunile 1 si 2

LSP – Capitolul 2 (parte programatică)

WSP – Capitolul 2 (parte programatică)

Cuprins

Sistemul de fisiere

De ce?
Definitie
Exemple

Ce este un sistem de fisiere?

mod de organizare persistenta a informatiei

suport pe disc

utilizator: ierarhie de fisiere si directoare

SO: algoritmi si structuri de date

performanta, scalabilitate, usurinta in utilizare

De ce sisteme de fisiere?

memoria este

- volatila
- de dimensiune mica

suport persistent

structurat, organizat

Exemple de sisteme de fisiere

NTFS

ext4

FAT32

ISO9660

HFS+

despre internele SF in cursul 10
Implementarea Sistemelor de Fisiere

Operatii cu SF

formatare
verificarea consistentei
montare
demontare
tuning (optiuni specifice)

Fișiere

Ce este un fisier?
Tipuri de fisiere
Fisiere
Directoare
Atribute ale fisierelor

Ce este un fisier?

unitatea de baza in sistemul de fisiere
abstractizeaza informatia, datele

informatia structurata

identificat de utilizator prin nume

identificat in SF printr-un numar
in Unix - inode number

Tipuri de fisiere

fisiere obisnuite (regular files)
directoare
link-uri
dispozitive de tip caracter
dispozitive de tip bloc
FIFO-uri
socketi Unix

pentru a determina tipul unui fisier in Unix

- ls -l
- file
- stat

Fisiere obisnuite

regular files

de obicei se numesc doar fisiere

byte stream - flux de octeti
se citeste si se scrie octet cu octet

organizarea datelor tine de tipul fisierului
si de procesul care il foloseste

Directoare

colectii de alte fisiere

organizate

- un vector de alte intrari

Atribute ale fisierelor

comanda stat

nume
identificator pe disc
dimensiune
drepturi de acces
owner
timpi de acces

Fisiere deschise

fisiere si fisiere deschise

procese si fisiere

handle de fisier

cursor de fisier

creare/deschidere fisier

citire/scriere

pozitionare

inchidere

Fisiere si fisiere deschise

fisierele sunt folosite de procese

in momentul folosirii un fisier este "deschis"

un fisier pe disc este static si referit prin nume

un fisier deschis este dinamic si referit prin handle

Procese si fisiere

fisierele sunt deschise de procese pentru a fi folosite

fisierele deschise sunt gestionate prin handle

un handle este o referinta la un fisier deschis

un proces poate deschide mai multe fisiere

un fisier poate fi deschis de mai multe procese
(e necesara sincronizare)

un proces are o tabela de handle-uri

- sunt retinute referintele la toate fisierele deschise

Cursorul de fisier

file cursor, file pointer, file offset

pozitia curenta pentru citire/scriere de un proces

incrementat la citire si scriere

poate fi repozitionat

Unde se gaseste cursorul de fisier la deschiderea fisierului?

Ce se intampla cu cursorul de fisier daca doua procese deschid acelasi fisier?

Procese, fisiere, handle-uri

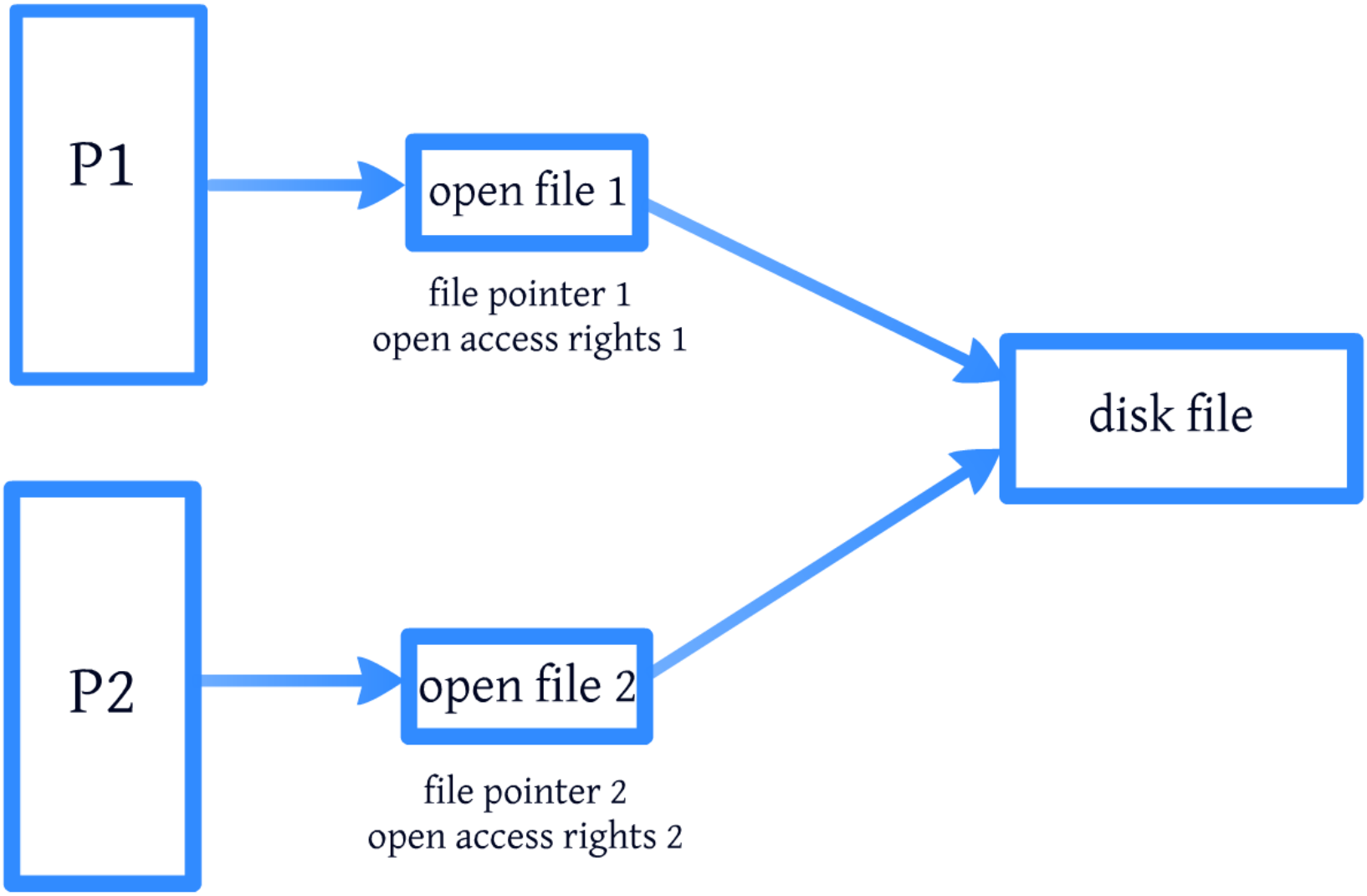
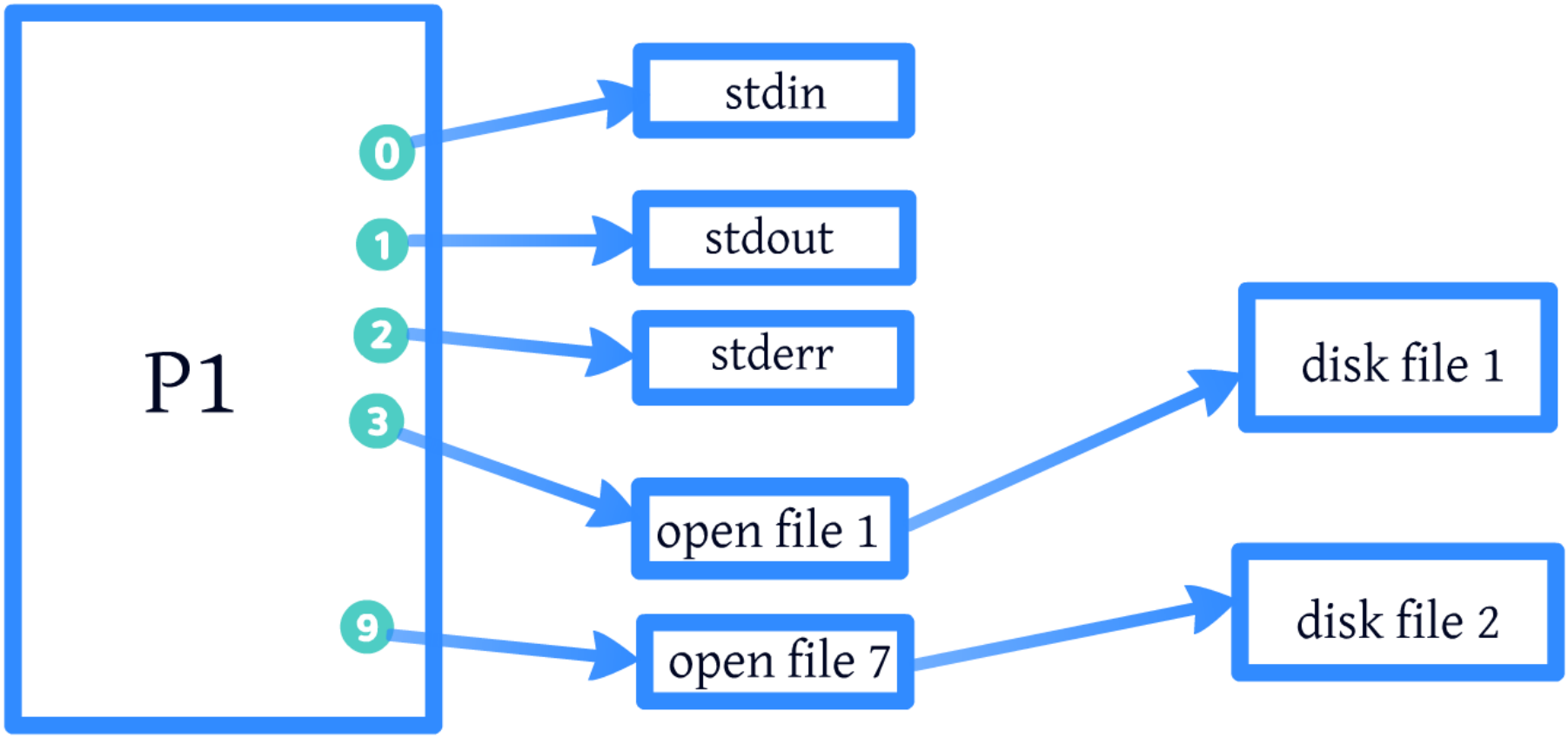


Tabela de descriptori

file descriptor table
file descriptor (Unix) = handle



Deschiderea unui fisier

open & close

realizata de un proces

intrare: nume de fisier, drepturi de deschidere

iesire: handle de fisier

se aloca o structura de fisier deschis

la inchiderea fisierului, se invalideaza handle-ul

se dezaloca structura de fisier deschis

Citire si scriere

read & write

pe un fisier deschis, pe un handle

la intrare: handle, cat se scrie/citeste, unde/de unde

la iesire: cat s-a scris/citit

dupa citire, scriere, se avanseaza cursorul de fisier

Pozitionare

seek

se comanda plasarea cursorului de fisier

poate fi dat inapoi (rewind)

input: handle, cu cat se muta, in raport cu ce

output: noua pozitie

API de lucru cu fisiere

shell
ANSI C
Unix (POSIX)
Windows (Win32)
Java
Python

shell

```
touch file.txt
cat file.txt
echo "abc" > file.txt
rm file.txt
```

ANSI C

```
FILE *f = fopen("file.txt", "r+");

n = fread(f, sizeof(char), 100, buffer);

m = fwrite(f, sizeof(char), 200, buffer2);

fseek(f, 100, SEEK_SET);

fclose(f);
```

Unix (POSIX)

```
int fd = open("file.txt", O_RDWR);

n = read(fd, buffer, 100);

m = write(f, buffer2, 200);

lseek(fd, 100, SEEK_SET);

close(fd);
```

Windows (Win32)

```
HANDLE f = CreateFile("file.txt", GENERIC_WRITE, ..., OPEN_EXISTING, ...);

ret = ReadFile(f, buffer, 100, &bytesRead, NULL);

ret = ReadFile(f, buffer, 100, &bytesRead, NULL);

SetFilePointer(f, 100, NULL, FILE_BEGIN);

CloseHandle(f);
```

Java

```
FileReader fr = new FileReader("f.txt");
fr.read(buffer);
fr.close();

FileWriter fw = new FileWriter("f.txt");
fw.write(buffer);
fw.close();

FileChannel fc = FileChannel.open("f.txt", READ, WRITE);
fc.position(100);
fc.close()
```

Python

```
f = open("f.txt", "r+")

str = f.read(10)

f.write(buffer)

f.seek(0, 0)  # place at beginning

f.close()
```

Redirectari

Ce inseamna redirectare?
De ce redirectare?
implementare redirectare

Ce inseamna redirectare?

iesirea standard este transmisa intr-un fisier
sau intrarea standard este transmisa dintr-un fisier

```
echo "ana" > file.txt
grep "ana" < /etc/passwd
```

sau este transmisa intr-un/dintr-un pipe

```
grep "ana" /etc/passwd | wc -l
```

De ce redirectare?

pentru a retine iesirea unei comenzi
pentru a transmite intrarea unei comenzi dintr-un fisier
pentru comunicare inter-proces (pipe), filtre
pentru a anula iesirea unei comenzi (/dev/null)

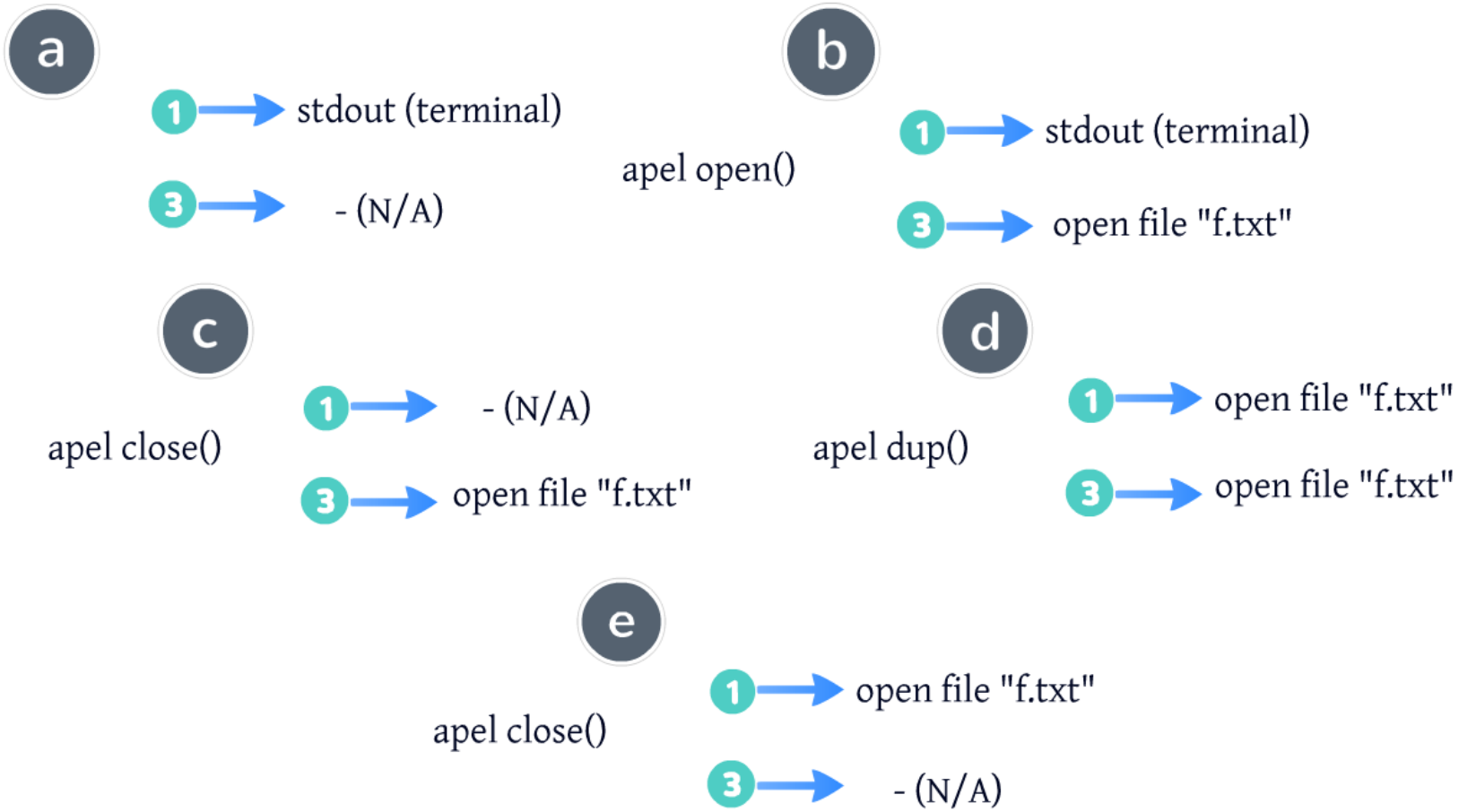
Implementare redirectare

echo "ana" > f.txt

```
fd = open("f.txt", O_WRONLY);
close(STDOUT_FILENO);
dup(fd);
close(fd);
```

dup - duplica un descriptor in primul descriptor liber
primul descriptor liber de mai sus este STDOUT_FILENO (1)

Implementare redirectare (cont.)




```
puts("ana are mere"); /* se va scrie in fisierul "f.txt" */
```

e au fost transmise

Buffered I/O vs. system I/O

buffered I/O

system I/O

versus

I/O

Buffered I/O

ANSI C I/O

biblioteca standard C face buffering la informatii
nu sunt transmise instant catre sistemul de operare

printf, fprintf, puts, fputs, fwrite

Cand sunt transmise sistemului de operare?

- la fflush()
- la fclose()
- line buffered (standard output)
- fully buffered (fisiere): cand s-a umplut buffer-ul

System I/O

low-level I/O

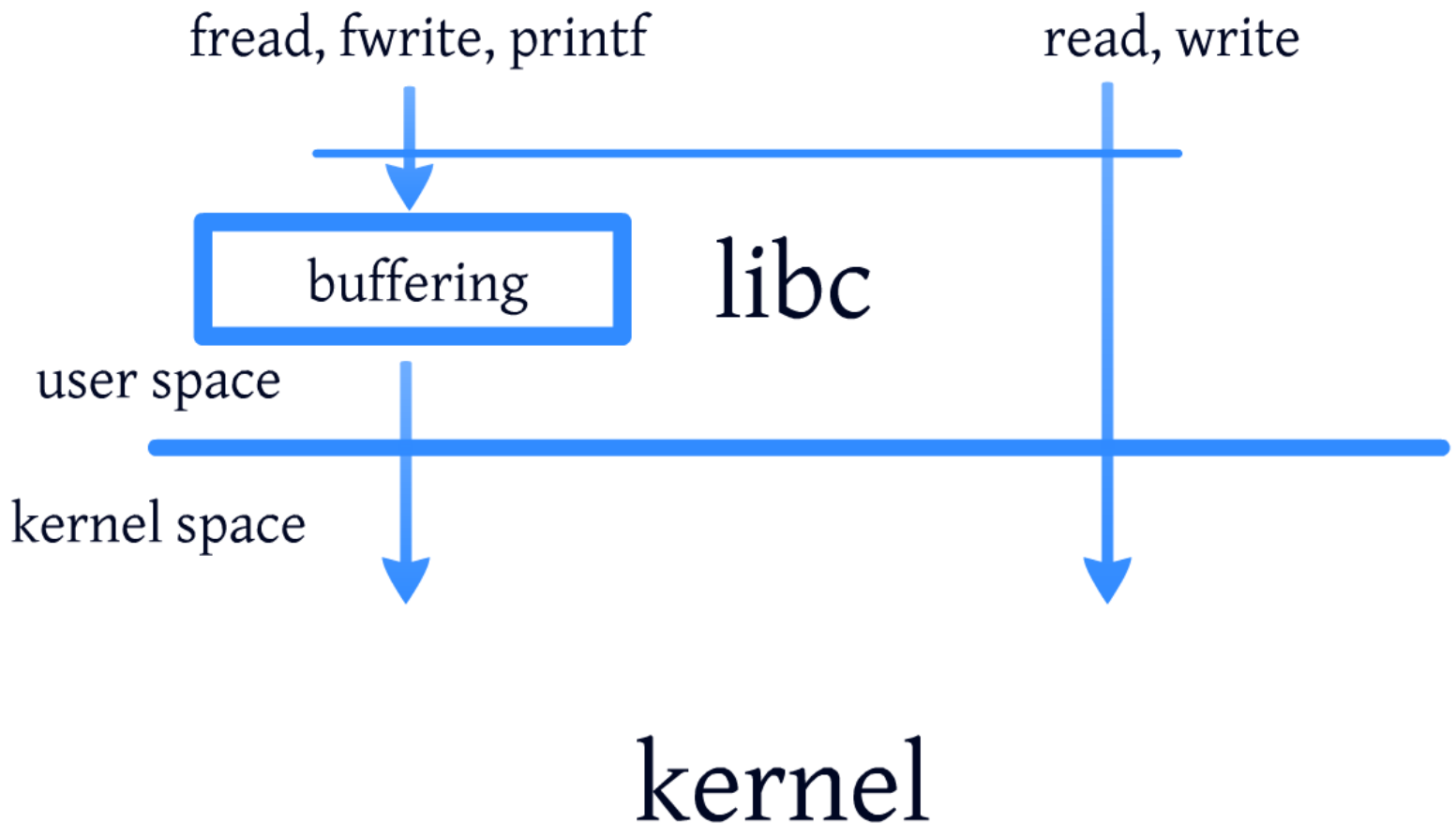
datele sunt transmise de la/catre SO

nu exista buffering

apelul se intoarce cand o parte din date au fost transmise

read, write

Buffered I/O & System I/O



Buffered I/O vs. system I/O

buffered I/O

buffered

latenta in propagare

memorie ocupata

mai putine apeluri de sistem

system I/O

sincron

transmitere directa

doar bufferele programului

overhead de apel de sistem

Cuvinte cheie

sistem de fisiere
fisiere
byte-stream
director
fisier deschis
handle de fisier

cursor de fisier
tabela de descriptori
API pentru fisiere
redirectare
buffered I/O
system I/O