

Realizacija sistema datoteka

- **Struktura sistema datoteka**
- **Realizacija sistema datoteka**
- **Realizacija direktorijuma**
- **Alokaciona metoda**
- **Upravljanje slobodnim prostorom**
- **Efikasnost i performanse**
- **Oporavak (*Recovery*)**
- **Log-Structured File Systems**

Struktura sistema datoteka

- **Struktura datoteka:**

- ☞ Logical storage unit

- ☞ Skup povezanih informacija

- **(FS) Sistem datoteka se nalazi na sekundarnoj memoriji (diskovi)**

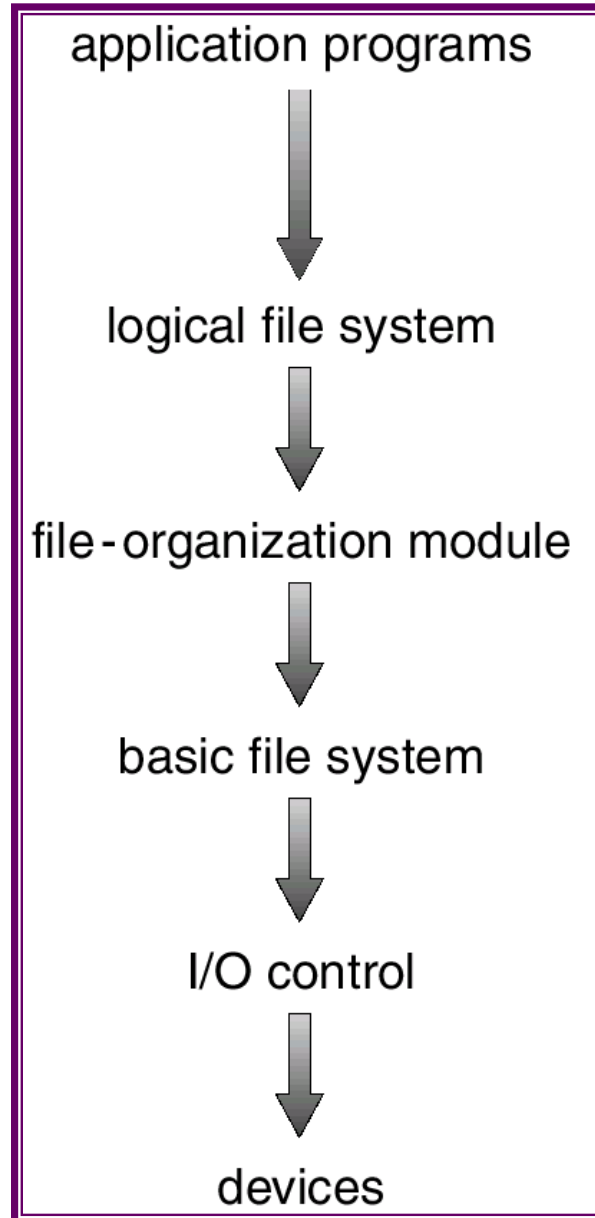
- **FS Sistem datoteka je organizovan u slojevima.**

- **Kontrolni blok datoteke (FCB):**

- ☞ memorijska struktura

- ☞ koja sadrži informacije o datoteci

Sistem datoteka realizovan u slojevima



**Poziva datoteku
po imenu**

**Podrška za strukturu
direktorijuma**

**Veza između datoteka i
blokova na disku**

**Direktan pristup
blokovima na disku**

**I/O control=
drajveri + rutine za
obradu prekida**

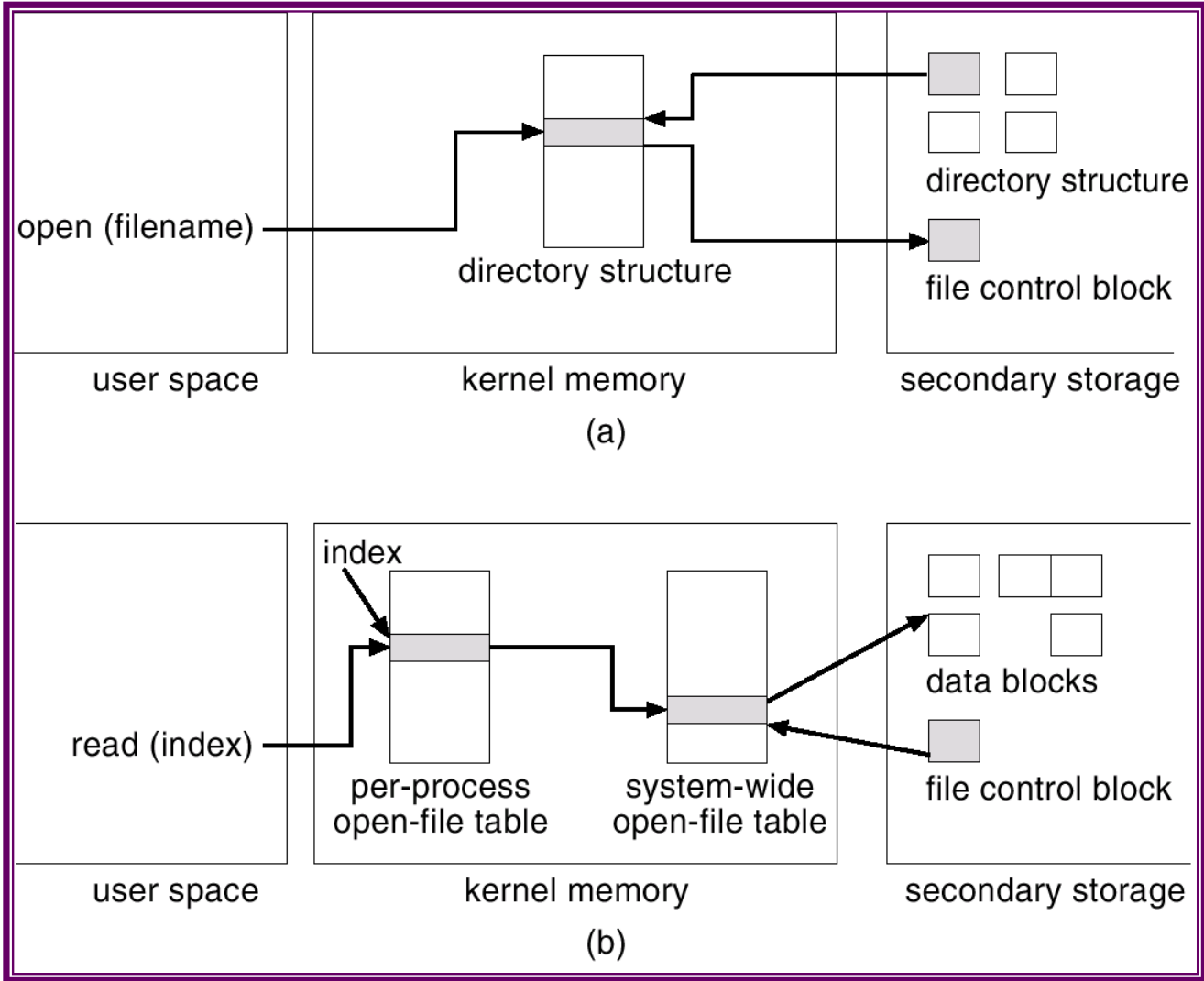
Strukture podataka neophodne za realizaciju sistema datoteka

- **Disk**
- **BCB** (*boot control block, MBR, MPT*)
- **Sistemi datoteka**
- **kontrolni blok particije (superblock, BPB)**
- **kontrolne strukture za alokaciju datoteka (FAT table, inode table, MFT)**
- **direktorijumske strukture**
- **FCB (file-info)**

Memorijske strukture podataka

- **tabele otvorenih datoteka:**
 - ☞ na sistemskom nivou
 - ☞ po procesu
- **keš za direktorijume** = nedavno korišćeni blokovi direktorijuma
- **keš za metapodatke** = nedavno korišćene strukture metapodataka
- **keš za datoteke**
- **strukture za upravljanje slobodnim prostorom**

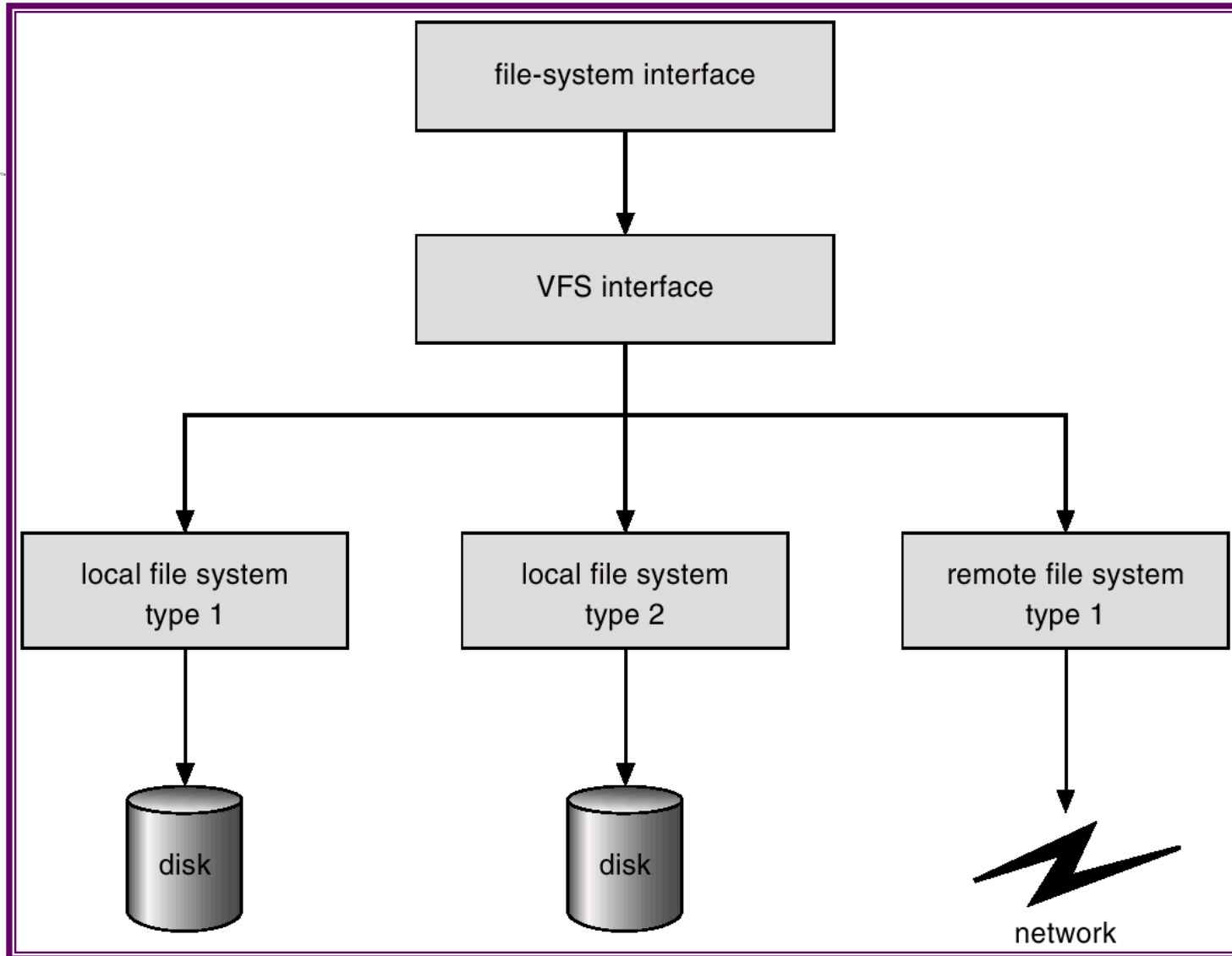
Memorijske strukture podataka



Virtuelni sistem datoteka

- **Virtuelni sistem datoteka (VFS)**
- predstavlja
 - ☞ jedan **objektno orjentisani** način
 - ☞ **realizacije sistema datoteka**
- **VFS omogućava**
 - ☞ da isti **systemski poziv (API)**
 - ☞ **bude korišćen**
 - ☞ **za različite tipove sistema datoteka**
- **API je u VFS interfejsu,**
 - ☞ a ne u bilo kom drugom sistemu datoteka

Virtuelni sistem datoteka – šematski prikaz



Realizacija direktorijuma

■ 1. Linearna lista

■ linearna lista imena datoteka sa pokazivačem na blokove podataka

- ☞ jednostavna šema

- ☞ **dugo pretraživanje**

■ 2. Hash tabela:

■ linearna lista sa hash tabelom

- ☞ smanjuje vreme pretraživanja direktorijuma

- ☞ kolizije

- ☞ fiksne veličine

■ 3. B-trees (B+ or B-)

Alokaciona metoda

■ Alokaciona metoda

- ☞ predstavlja način
- ☞ na koji se blokovi diska
- ☞ dodeljuju datoteci:

■ Kontinualna alokacija

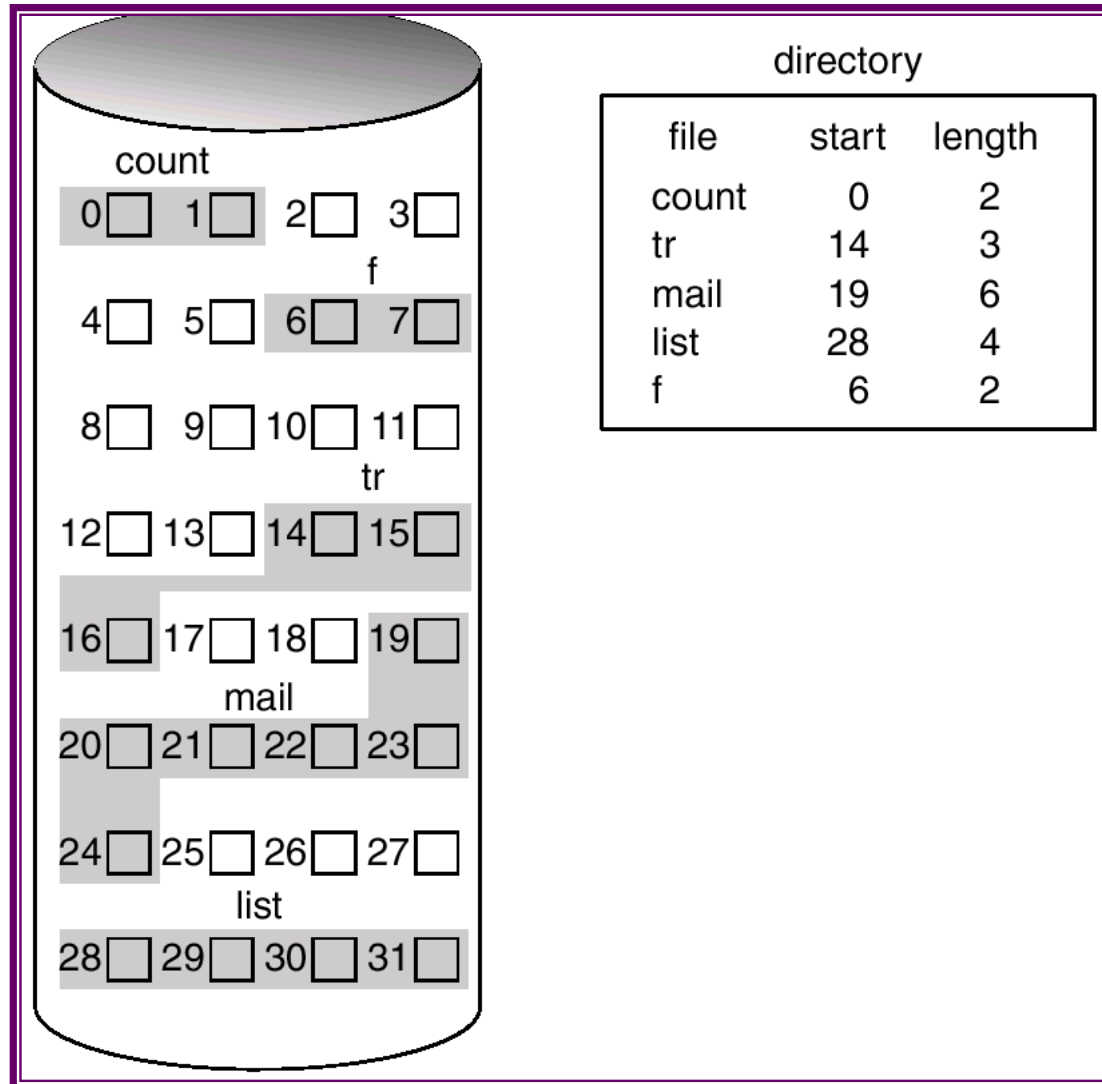
■ Povezana alokacija

■ Indeksna alokacija

Kontinualna alokacija

- Svaka datoteka zauzima
 - ☞ **kontinualne blokove**
 - ☞ **na disku**
- **Prosta metoda, FCB (file-info) jednostavan**, sastoji se od:
 - ☞ **početne adrese (blok #)**
 - ☞ **dužine (broj blokova)**
- **Proizvoljan pristup (random access)**
 - ☞ (direktan pristup bilo kom delu datoteke)
- **Rasipanje memorijskog prostora**
 - ☞ (za datoteku odmah mora dodeliti ukupni adresni prostor)
- **Datoteka ne može da raste**

Kontinualna alokacija prostora na disku



Problemi kod kontinualne alokacije

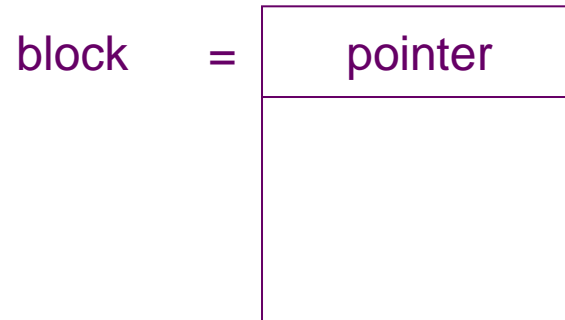
- Eksterna fragmentacija i kompakcija
- Dinamička alokacija memorije:
 - ☞ **First fit** (prva šupljina koja je dovoljno velika)
 - ☞ **Best fit** (najmanja šupljina)
 - ☞ **Worst fit** (najveća moguća šupljina)
- Glavni problem=creation:
 - ☞ proces mora da zna veličinu datoteke koja će biti formirana
 - ☞ to veoma je teško proceniti u trenutku formiranja
- Ako je alociran **suviše mali prostor** =>a datoteka **treba da raste**
- Postoje 2 mogućnosti:
 - ☞ 01. Proces će se završiti sa porukom o grešci
 - ☞ 02. Kopiraće datoteku u veću šupljinu i osloboditi prošlu

Extent metoda

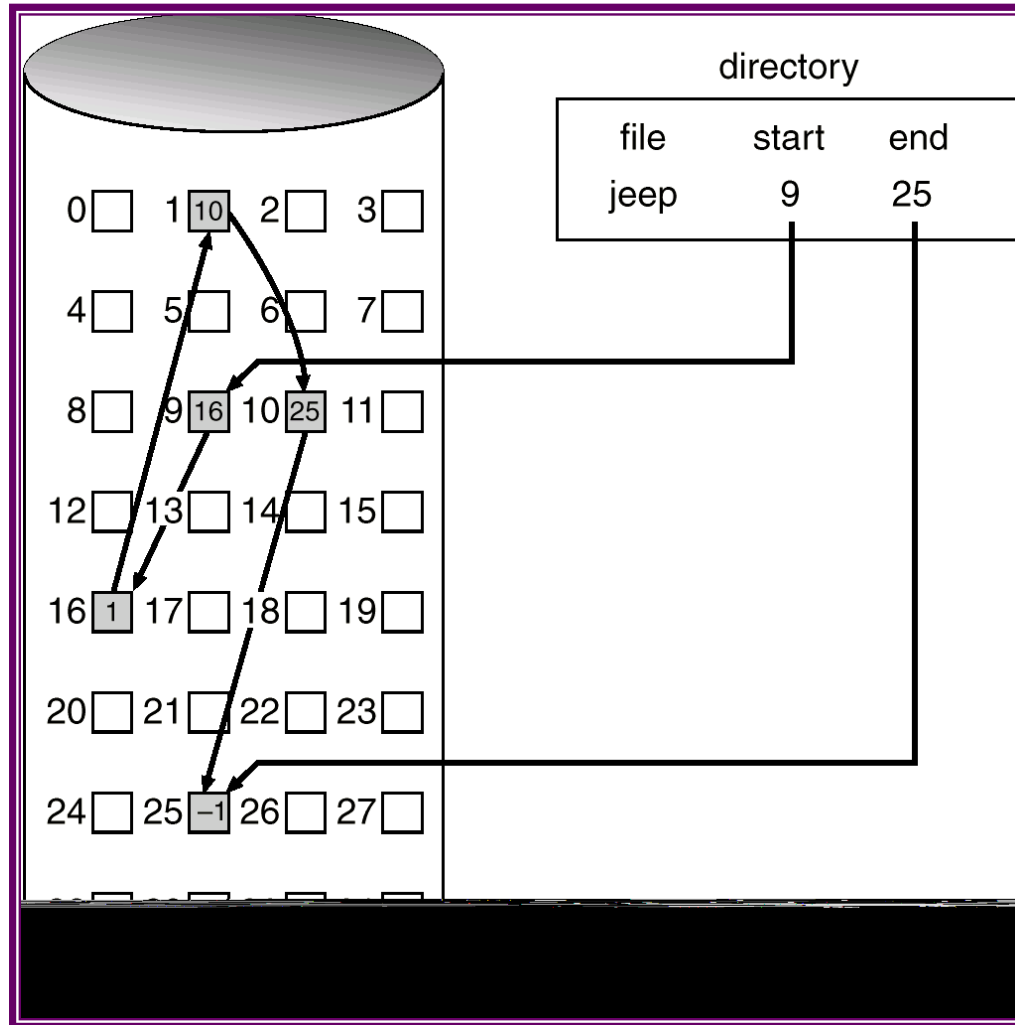
- **Mnogi noviji sistemi datoteka (npr. VFS)**
 - ☞ koriste **modifikovanu**
 - ☞ **metodu kontinualne alokacije**
- **Sistem datoteka baziran na extent metodi**
 - ☞ **alocira blokove na disku**
 - ☞ **u celinama-extents**
- **Extent je celina sastavljena od kontinualnih blokova na disku**
 - ☞ **extent je jedinica promenljive veličine**
 - ☞ **extent se koristi za alokaciju datoteke**
 - ☞ **datoteka se sastoji od jednog ili više extent-a**

Povezana alokacija

- Svaka datoteka dobija povezanu listu blokova
- blokovi mogu biti razbacani svugde po disku



Povezana alokacija



Povezana alokacija (2)

- Prosta metoda, **FCB** zahteva samo početnu adresu
- Sistem za upravljanje slobodnim prostorom:
 - ☞ nema rasipanja memorijskog prostora
- Ne postoji **random** pristup
 - ☞ nemože direktno pristupiti
 - ☞ bilo kom delu datoteke
 - ☞ **bez prethodne analize**

Prednosti i mane povezane alokacije

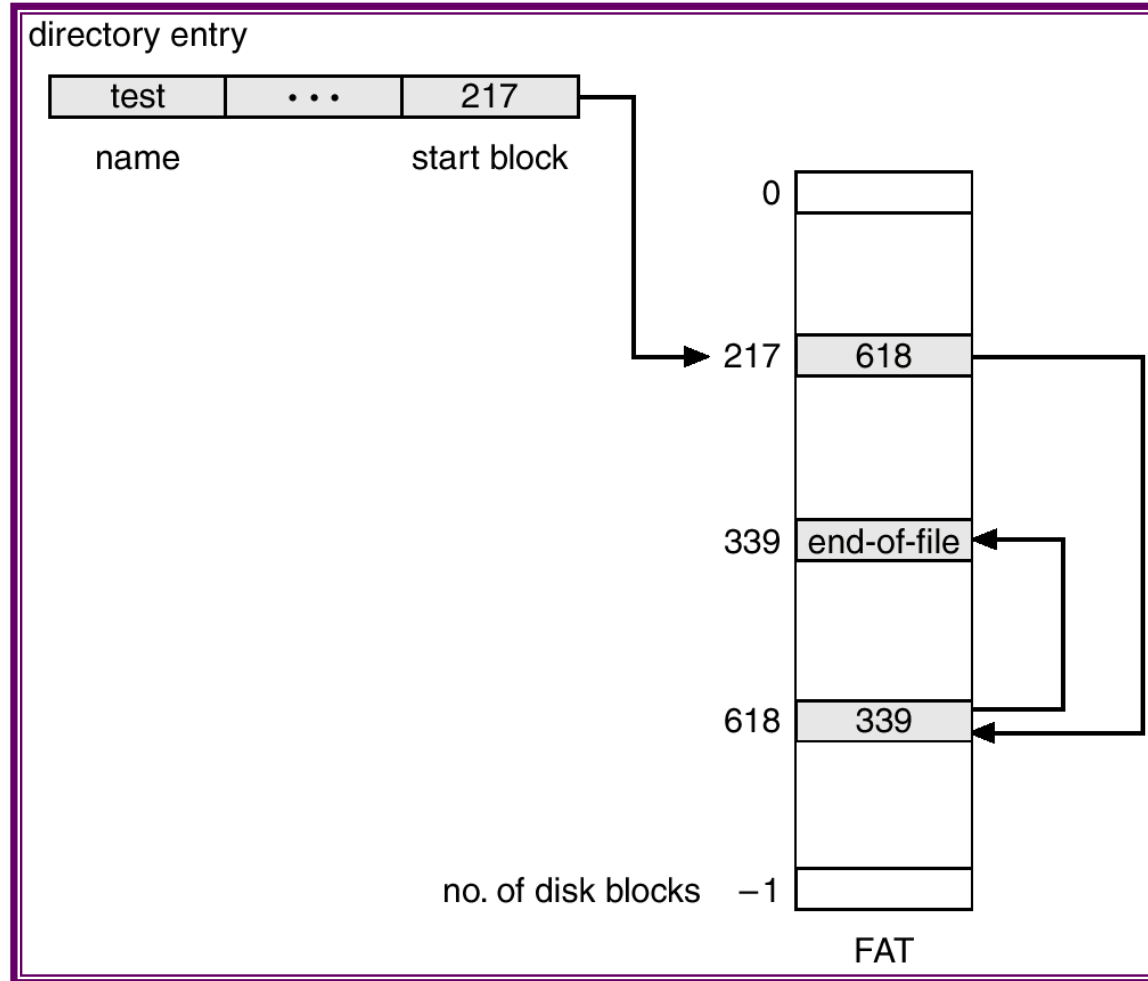
■ Prednosti:

- ☞ Nema problema sa eksternom fragmentacijom
- ☞ Kreiranje datoteke: nije potrebno definisati veličinu datoteke
- ☞ Datoteka može da raste dok ima slobodnih blokova
- ☞ Nikada nije neophodan **kompaktan** prostor na disku

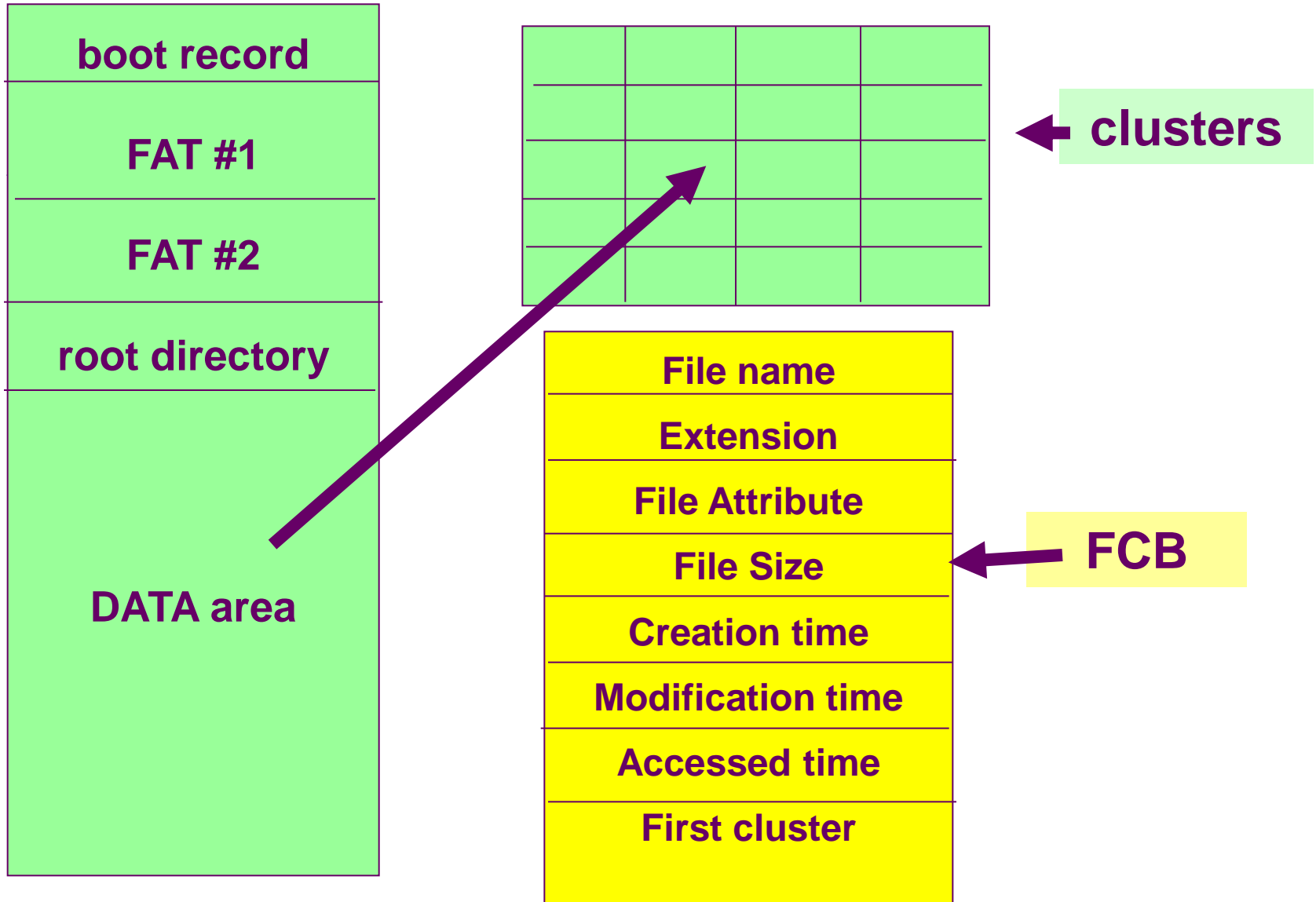
■ Nedostaci:

- ☞ Datoteke su razbacane, ali su i pokazivači na **blokove razbacani**
- ☞ Usporeno pretraživanje zbog sekvencijalnog pristupa
- ☞ Loše za proizvoljno pristupanje datoteci
- ☞ Loše za pristupanje velikim datotekama
- ☞ Pokazivači => gubitak prostora
- ☞ Pouzdanost = loši pokazivači mogu stvoriti mnoge greške FS

File-Allocation Table

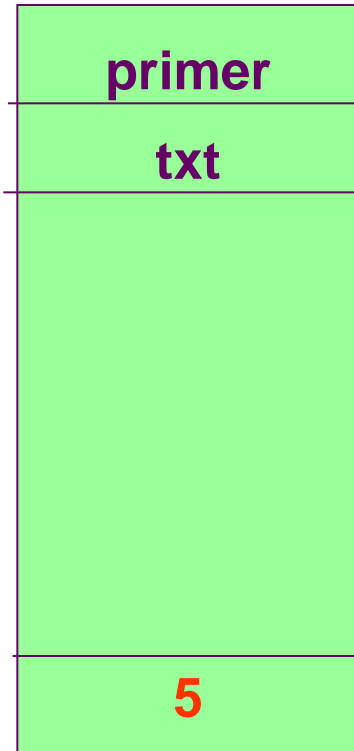


Primer FAT tabelle

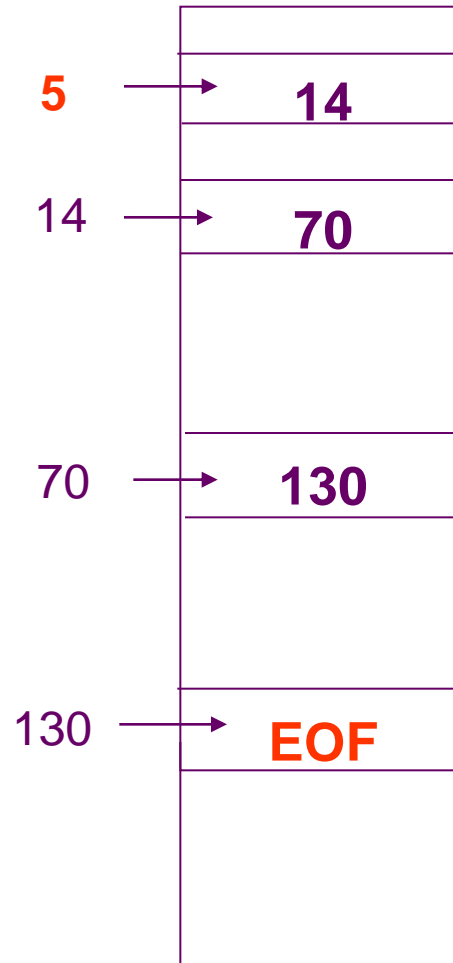


Primer: primer.txt = {5, 14, 70, 130}

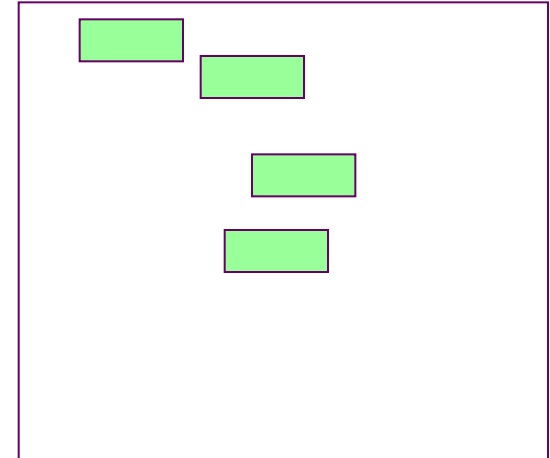
FCB



FAT

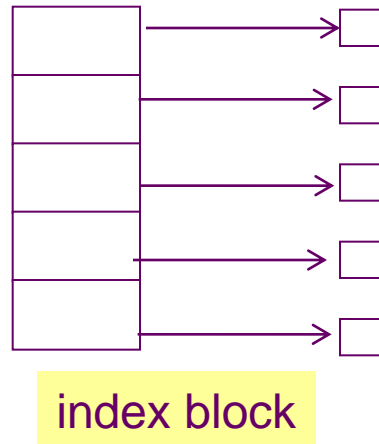


Data area

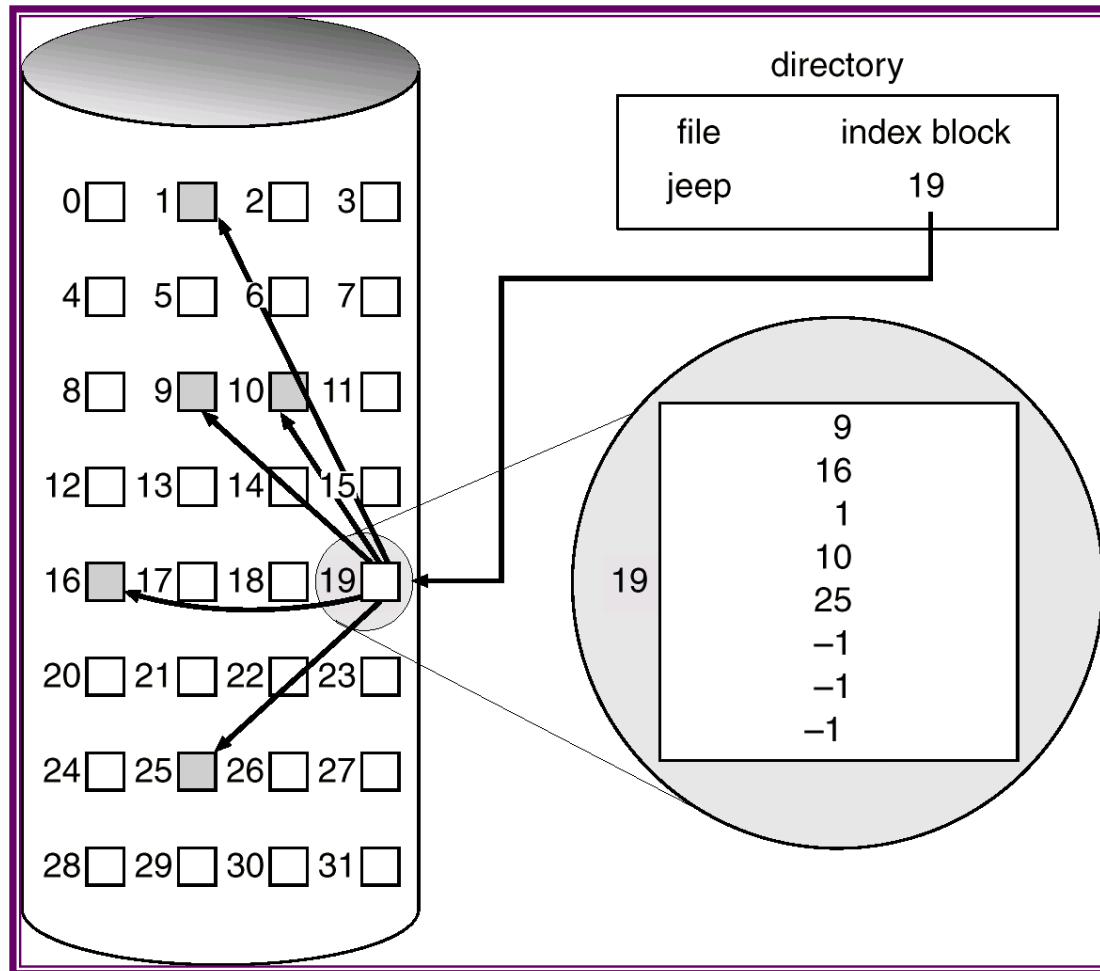


Indeksna alokacija

- Svakoj datoteci se dodeljuje **indeksni blok**
- koji **sadrži sve informacije o prostornom rasporedu datoteke**
- Logički prikaz



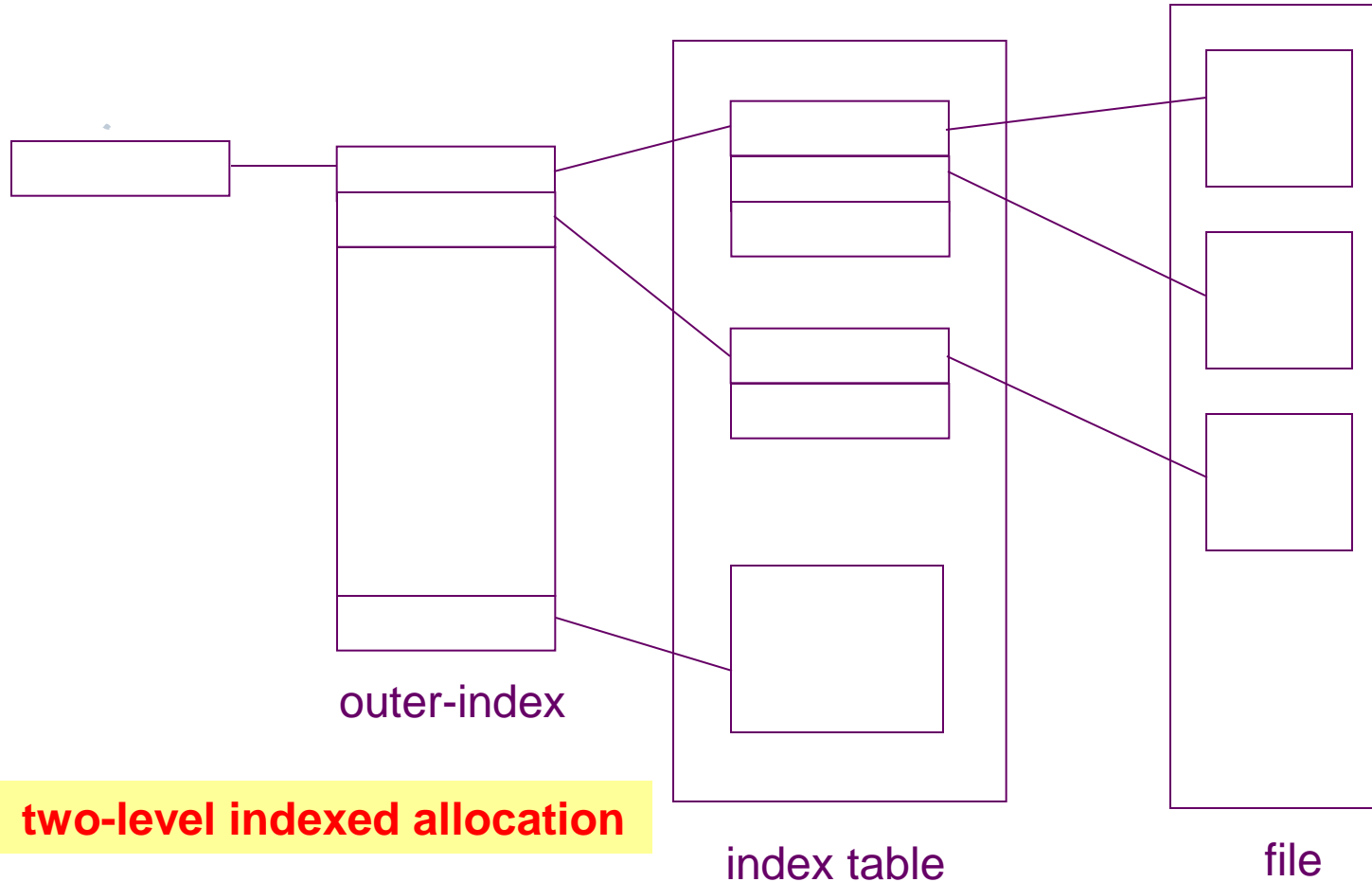
Primer indeksne alokacije



Indeksna alokacija (2)

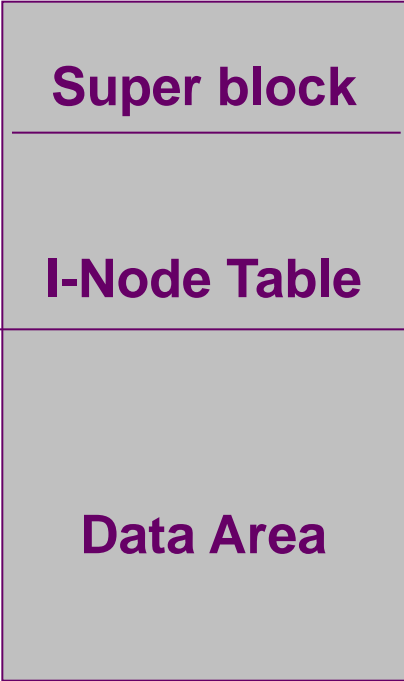
- **Potrebna je indeksna tabela**
- **Random pristup**
- **Dinamički pristup**
 - ☞ nema eksternu fragmentaciju
 - ☞ ali ima
 - ☞ **gubitak prostora (index blocks)**

Indeksna alokacija – mapiranje

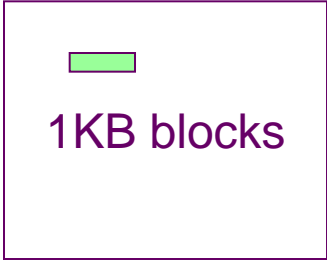


UNIX FS

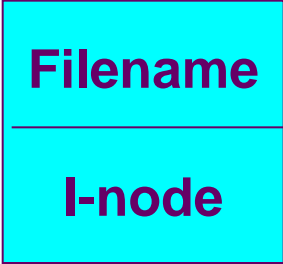
■ FS Layout



DATA area



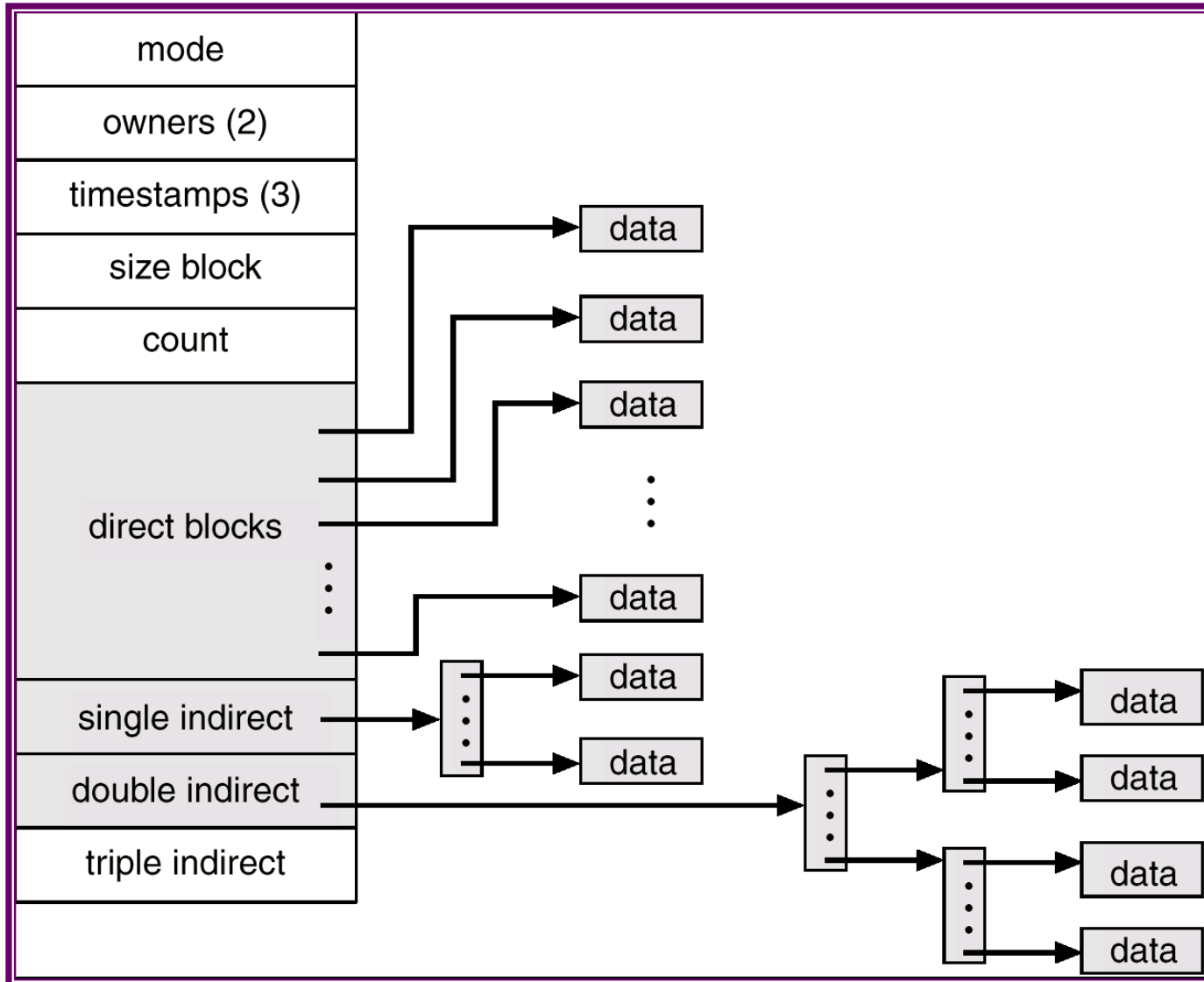
FCB



Inode

File Type & AR
Number of links
Owner's ID
Group's ID
File size
Array of disk block pointers
Accessed time
Modification time
Creation time

UNIX (4KB po bloku)



Poređenje

■ Kontinualna alokacija

- ☞ najbrža jer posle čitanja FCB,
- ☞ prelazi se na direktno čitanje datoteke

■ Linkovana alokacija

- ☞ dobra za sekvencijalan pristup
 - 📄 čita blok i čita sledeći pokazivač
 - 📄 dva puta pristupa disku za jedan blok datoteke
- ☞ veoma loša za direktan pristup
 - 📄 za i -ti blok može zahtevati i čitanja diska
- ☞ Kao rezultat, neki sistemi podržavaju
 - 📄 direktan pristup korišćenjem CA
 - 📄 sekvencijalan pristup korišćenjem LA

Poređenje

- **Indeksna alokacija je najkompleksnija**

- **Performanse IA zavise od:**

- ☞ **strukture indeksa i strukture memorije**

- ☞ **veliĉine datoteke**

- ☞ **pozicije potrebnih blokova**

- 📄 **direktni blok je brži**

- 📄 **indirektni blok je sporiji**

- **Kombinacija CA+IA**

- ☞ **za manje datoteke = CA**

- ☞ **za veće datoteke = IA**

Upravljanje slobodnim prostorom

■ Bit vector (n blocks)



bit[i] =

0 \Rightarrow block[i] blok je slobodan

1 \Rightarrow block[i] blok je zauzet

Bit mape **zauzimaju prostor na disku** (N blokova= N bitova)

Primer: blok veličine = 2^{12} bytes=4K

veličina diska = 2^{30} bytes (1 gigabyte)

$n = 2^{30}/2^{12} = 2^{18}$ blokova

= 2^{18} bits (ili 32K bytes)

Jednostavno je ako su datoteke kontinualne

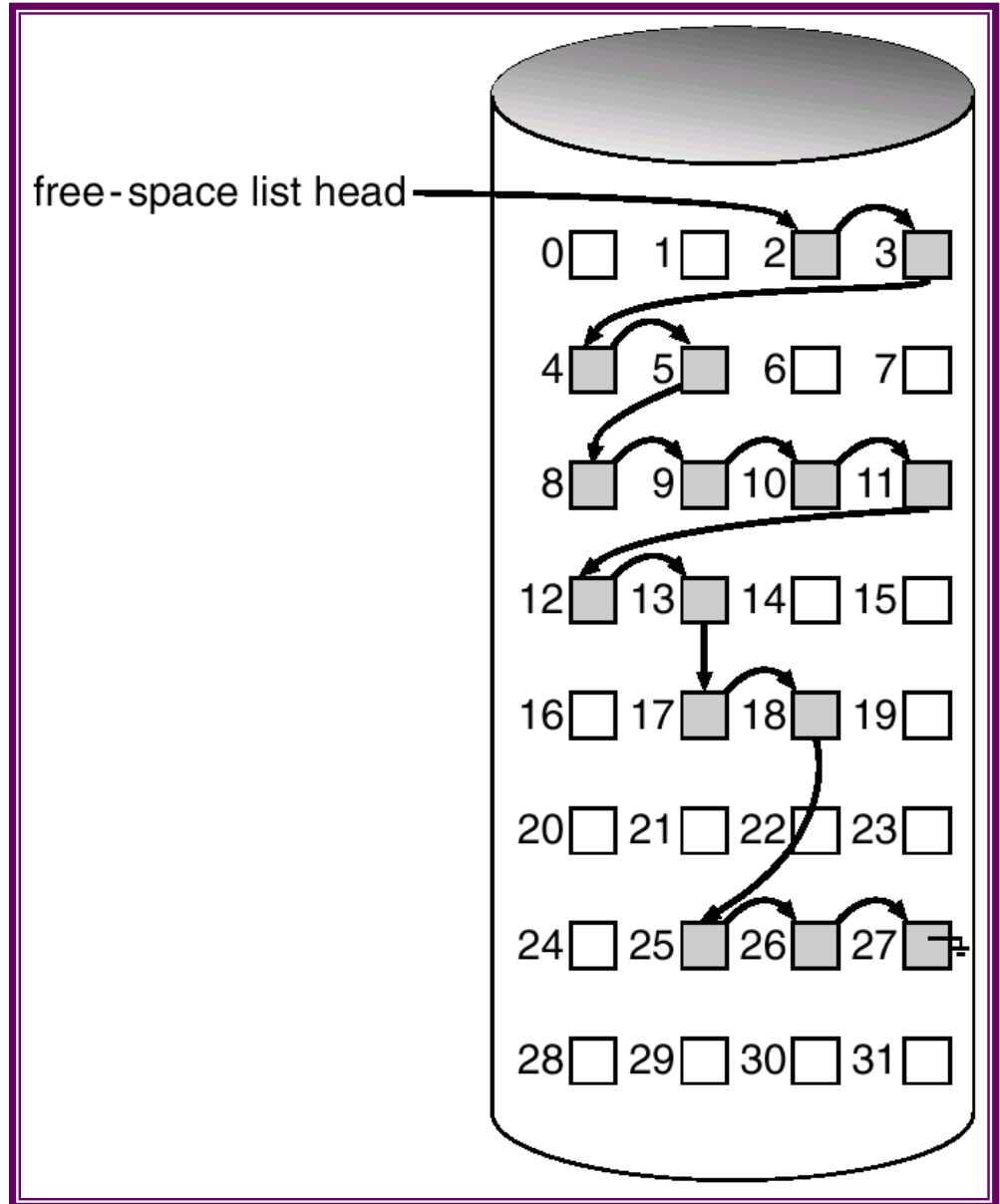
Linkovane liste

Linkovane liste

Nema dopunske strukture
za vođenje evidencije o
listi slobodnih blokova

Nema gubitka prostora

Performanse su
katastrofalno loše



Upravljanje slobodnim prostorom(2)

■ Modifikacije:

👉 1. Grupisanje

📄 U prvom bloku se čuvju pokazivači

📄 na sledećih N-1 slobodnih blokova

📄 Dok N-ti na sledeći free information block

👉 2. Brojanje (*Counting*)

📄 adresa slobodnih blokova (pointer)

📄 +

📄 broj slobodnih blokova u blizini

Efikasnost i performanse

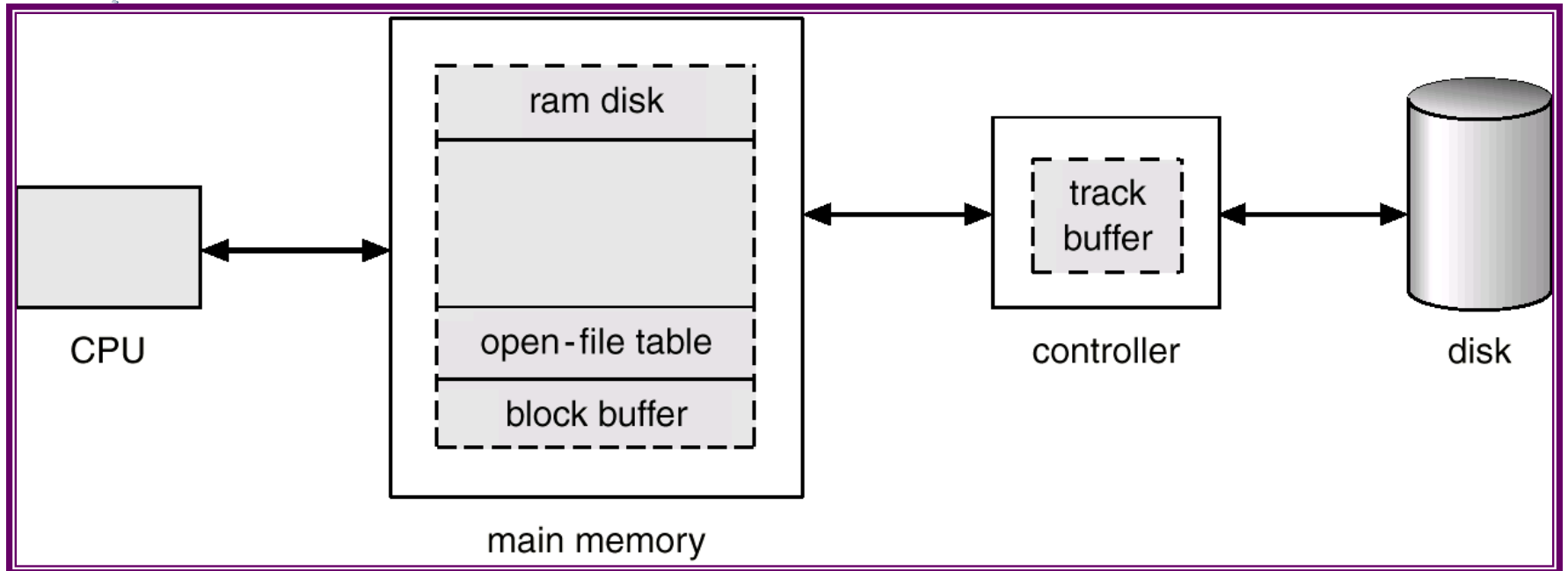
■ Efikasnost zavisi od:

- ☞ Disk-alokacije i algoritama direktorijuma
- ☞ tipova podataka koji se čuvaju u direktorijumskim ulazima

■ Performanse:

- ☞ disk keš:
 - 📄 posebni delovi u glavnoj memoriji za blokove koji se često koriste
- ☞ free-behind and read-ahead:
 - 📄 tehnike za optimizaciju sekvencijalnog pristupa
- ☞ unapređenje performansi PC-ja
 - 📄 tako što se odvoji deo u memoriji kao virtualni disk,
 - 📄 ili RAM disk

Razne komponente keša



Keš za stranice

■ Keš za stranice

- ➡ kešira stranice
- ➡ a ne blokove na disku
- ➡ korišćenjem tehnike virtualne memorije

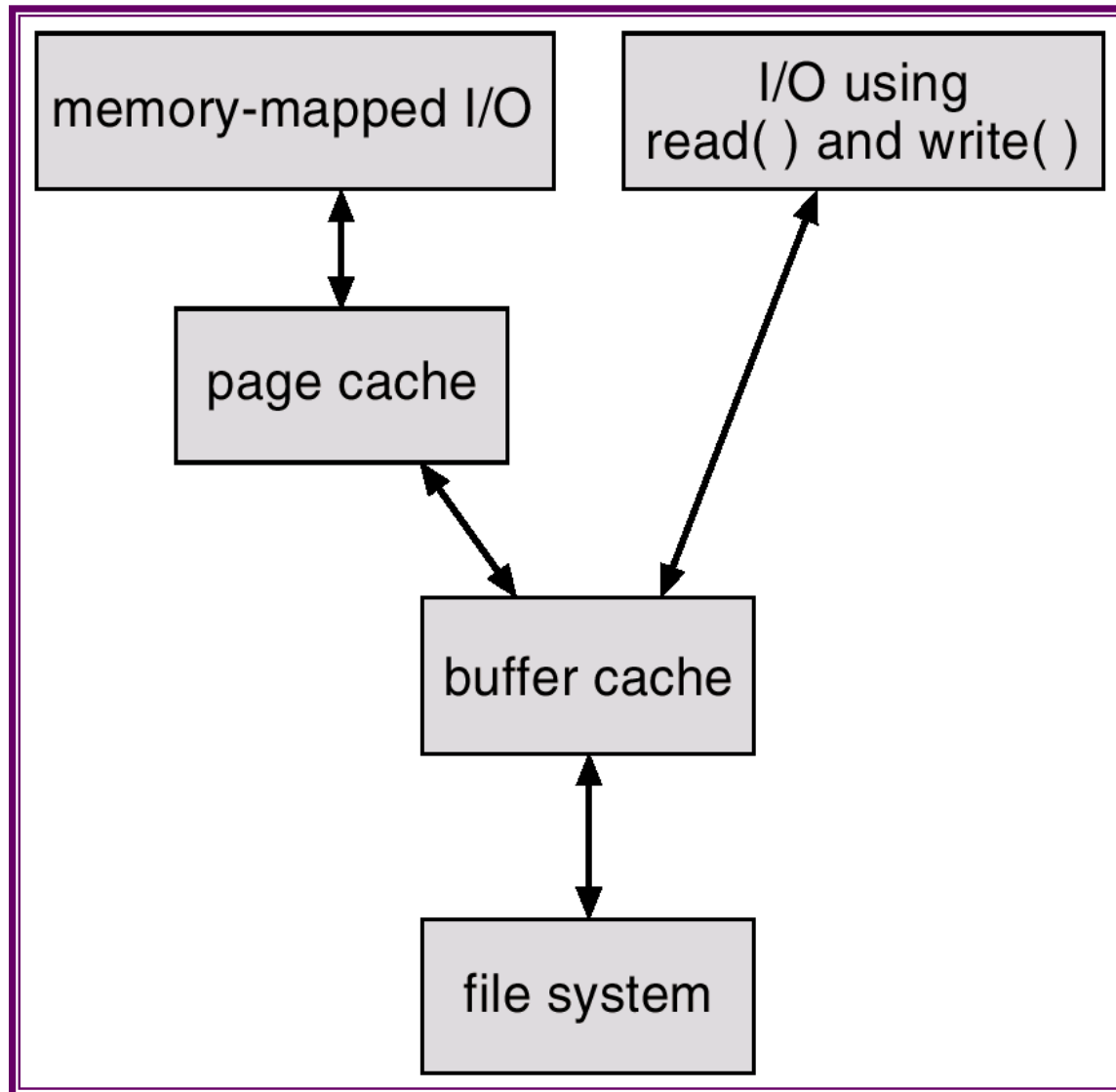
■ Memorijski mapirane datoteke koriste stranični keš

■ Datoteke: I/O Dražveri

- ➡ kroz sistem datoteka
- ➡ koriste **baferski (disk) keš**

■ Ovo je predstavljeno na sledećoj slici:

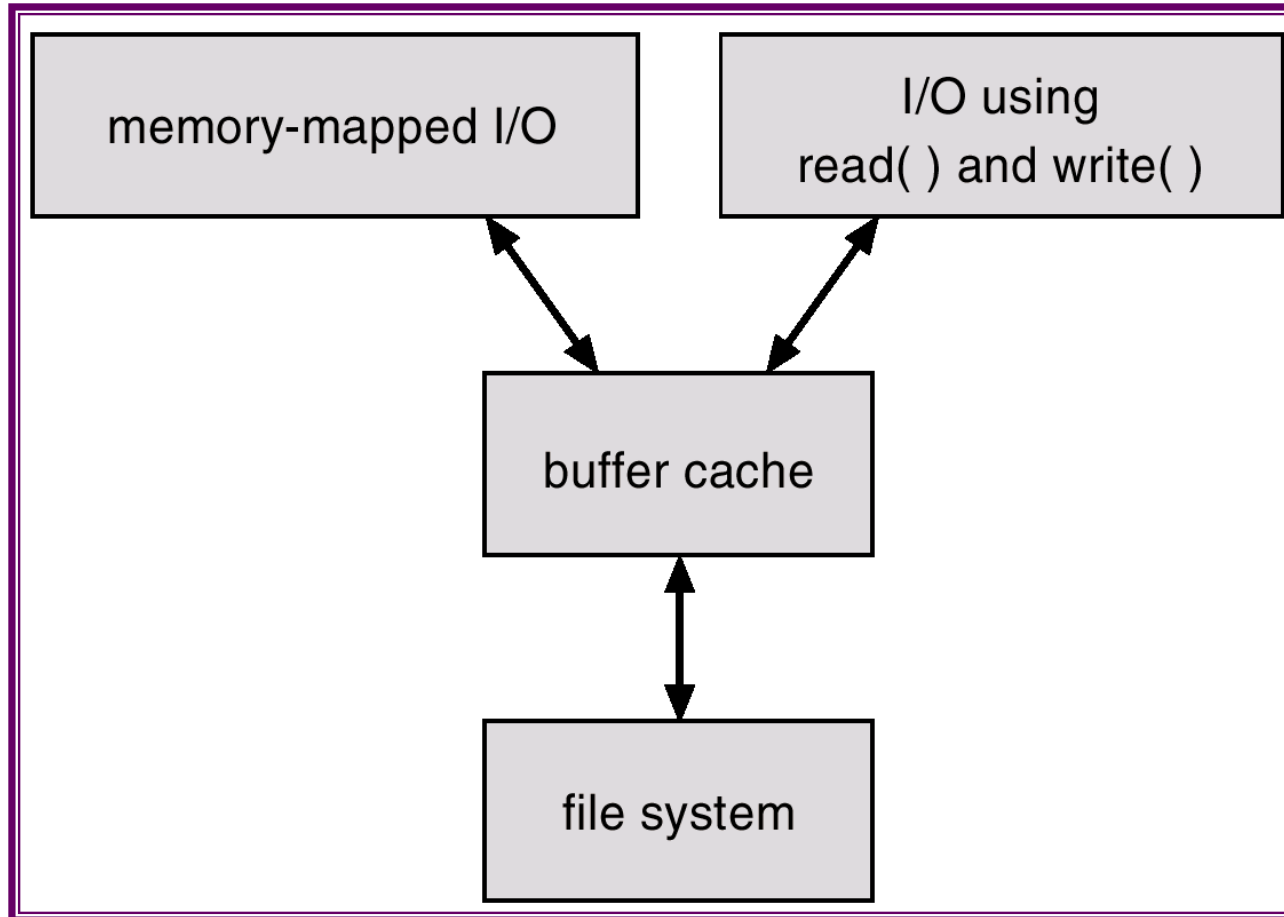
I/O Without a Unified Buffer Cache



Unified Buffer Cache

- **Unified buffer cache**
- **koristi isti keš za stranice**
 - ☞ **da kešira obe:**
 - ☞ **memorijski mapirane stranice**
 - ☞ **i**
 - ☞ **datoteke (običan FS I/O)**

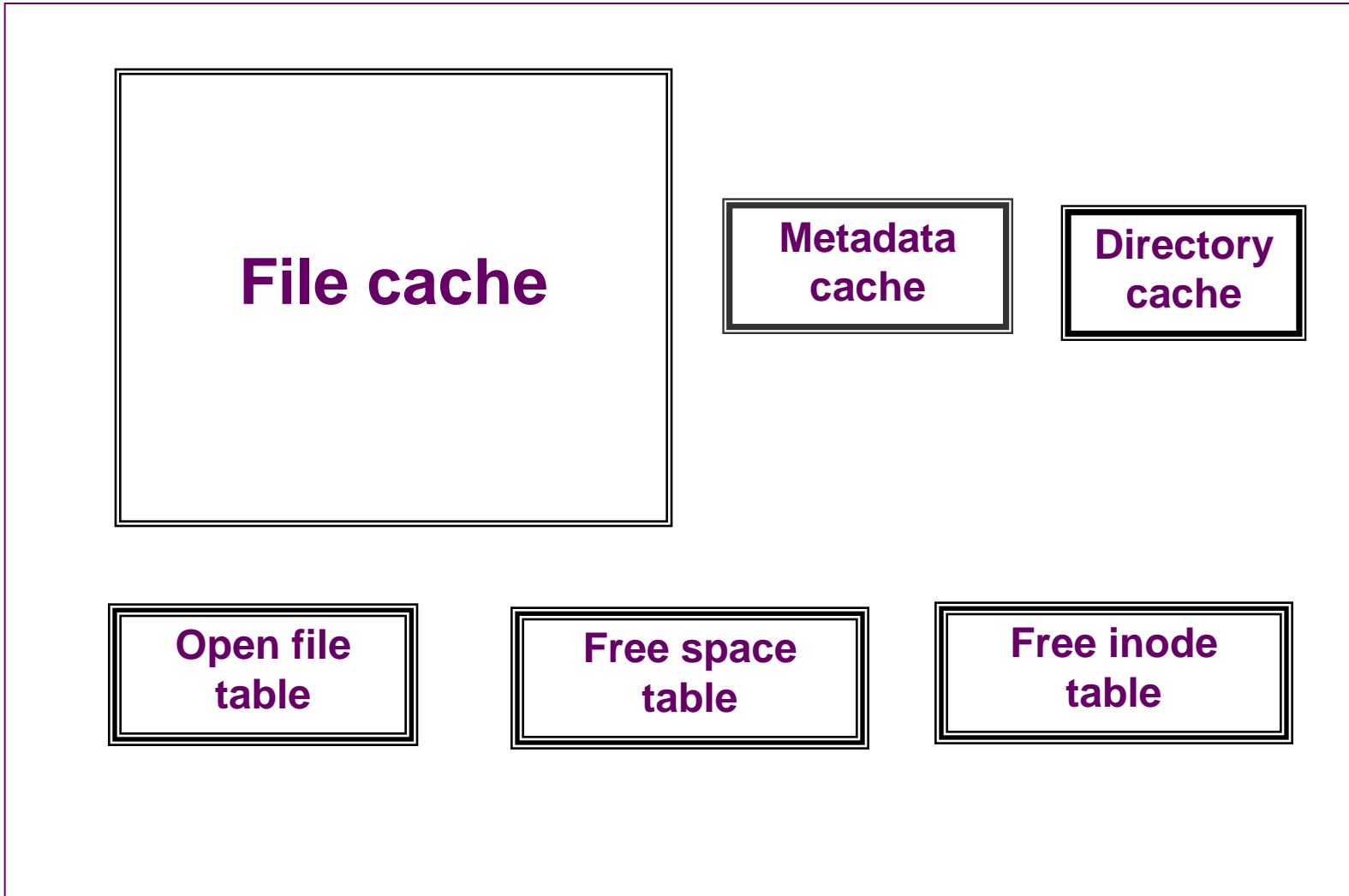
I/O Using a Unified Buffer Cache



Buffer Cache

- Podela keša
- Alokacija keša (*Cache allocation*)
- Razmena podataka u kešu (*Cache replacement*)
- Tehnika upisa podataka (*Write techniques*)
- Cache coherency for DFS

Podela keša



Organizacija keša

- Svaki deo keša ima svoju organizaciju
- **Keš direktorijuma** =
 - ☞ nedavno korišćeni direktorijumski blokovi
 - ☞ sortirani po imenu
- **Keš metapodataka** =
 - ☞ nedavno korišćeni metapodaci
 - ☞ sortirani po broju metapodatka (broj inoda)
- **Keš za datoteke**
 - ☞ Keš za definiciju blokova ($2^{**}N$ blokova diska)
 - ☞ Algoritam za alokaciju/pretraživanje
 - ☞ Standardna šema

Razmenjivanje podataka u kešu

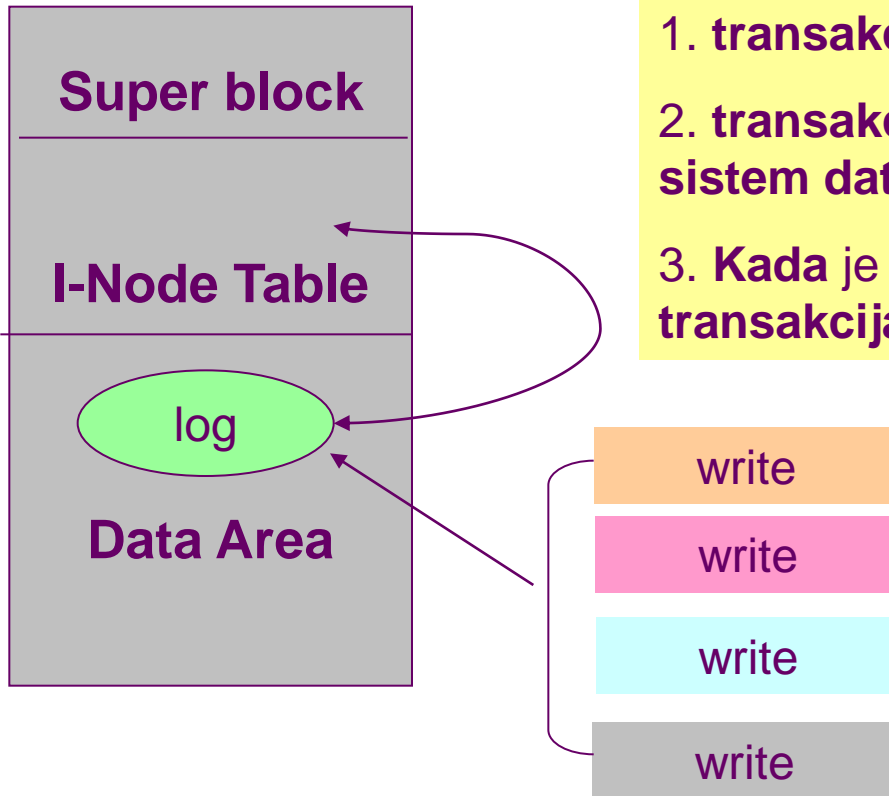
- Svi delovi keša su ograničene veličine
- Razmenjivanje podataka u kešu je neophodno
- Tehnike za razmenjivanje podataka u kešu
 - ☞ LRU
 - ☞ LFU
 - ☞ LRFU
- Veoma složen za DFS, Internet, proxy caching ...
- Kombinacija vrednosti (prostor x starost, vreme uzimanja podataka)

Tehnike upisa

- **Keš bez mogućnosti upisa = niske performanse**
- **Dve metode**
 - ☞ **Write-through (sinhronizovani upis)**
 - 📄 Siguran sistem, bez gubitka podataka, **ali sa niskim performansama**
 - ☞ **Write-back (odloženi upis)**
 - 📄 Visoke performanse, ali postoji gubitak podataka u slučaju da sistem bude oboren
- **Tehnike odloženog upisa**
 - ☞ **Upis nakon zatvaranja (*Write on eject*)**
 - ☞ **Pražnjenje bafera (*Flushing*)**
 - ☞ **Log-approach**
 - ☞ **Log for meta data, only = JFS**
 - ☞ **Total log = LFS**

Journaling –LOG approach

- svi metapodacii se ažuriraju (t) → u jednom velikom logu



1. transakcije se upisuju u log

2. transakcije u logu se asinhrono upisuju u sistem datoteka.

3. Kada je sistem datoteka modifikovan, transakcija se briše iz loga

JFS - Log Structured File Systems

■ Log structured (or journaling) file systems

- ☞ snimaju svako ažuriranje sistema datoteka
- ☞ kao transakciju

■ Sve transakcije se upisuju u log

- ☞ transakcija se izvršava
- ☞ samo kada je upisana u log
- ☞ Međutim, sistem datoteka se možda ne može odmah ažurirati

■ Transakcije u log

- ☞ se asinhrono upisuju u sistem datoteka.
- ☞ kada je sistem datoteka modifikovan,
- ☞ transakcija se briše iz loga.

■ U slučaju obaranja sistema,

- ☞ sve postojeće transakcije u logu
- ☞ trebaju da nastave da funkcionišu

Oporavak (*Recovery*)

- **Keširanje i obaranje sistema**
 - ☞ mogu prouzrokovati
 - ☞ oštećenje podataka
- **Provera konzistencije:**
 - ☞ upoređuje podatke u strukturi direktorijuma
 - ☞ sa blokovima podataka na disku
 - ☞ pokušava da locira nekonzistentne podatke
- **[cache data] must be = [disk data]**
- **U slučaju obaranja sistema podaci mogu biti oštećeni**
- **Koristiti sistemske programe**
 - ☞ za **pravljenje kopije podataka** sa diska na drugi memorijski uređaj
 - ☞ (flopi disk, magnetna traka).
- **Oporavak izgubljenih datoteka**
 - ☞ prebacivanjem
 - ☞ **sa bekapa**