Sistem datoteka

Vladimir Filipović

Sistem datoteka

- Trajno čuvanja podataka je mogućnost koja se podrazumeva i bez koje se ne mogu zamisliti savremeni računarski sistemi.
  - Radna (primarna) memorija nije pogodna za skladištenje podataka jer njena veličina nije dovoljno velika.
  - Problem predstavlja i činjenica da se sadržaj radne memorije dodeljen procesu gubi po prestanku rada sistema, ali i samog procesa.
- Za potrebe skladištenja podataka se koriste sekundarne (spoljašnje, trajne) memorije.

Sistem datoteka (2)

- Rešenje za trajno čuvanje podataka treba da zadovolji sledeće uslove:
  1. Sačuvani podaci mogu biti trajno zapisani, tj. ne gube se po prestanku izvršavanja procesa koji ih koristi ili po prestanku rada sistema.
  2. Velika količina podataka može biti sačuvana.
  3. Podaci su nezavisni od procesa, tj. više procesa im može pristupiti a ne samo onaj koji ih je kreirao.

Sistem datoteka (3)

- Datoteka (fajl) je apstrakcija informacije zapisane na sekundarnoj memoriji.
- Korisniku ovu apstrakciju pruža sistem datoteka i to tako da korisnik „vidi“ samo određeni interfejs pogodan za korišćenje, ali ne i detalje implementacije.
- Datoteka predstavlja kolekciju informacija, zapisanu u memoriji, kojoj je pridruženo ime.
- Sa korisničke strane, to je najmanja struktura informacije koja se može upisati u memoriju.

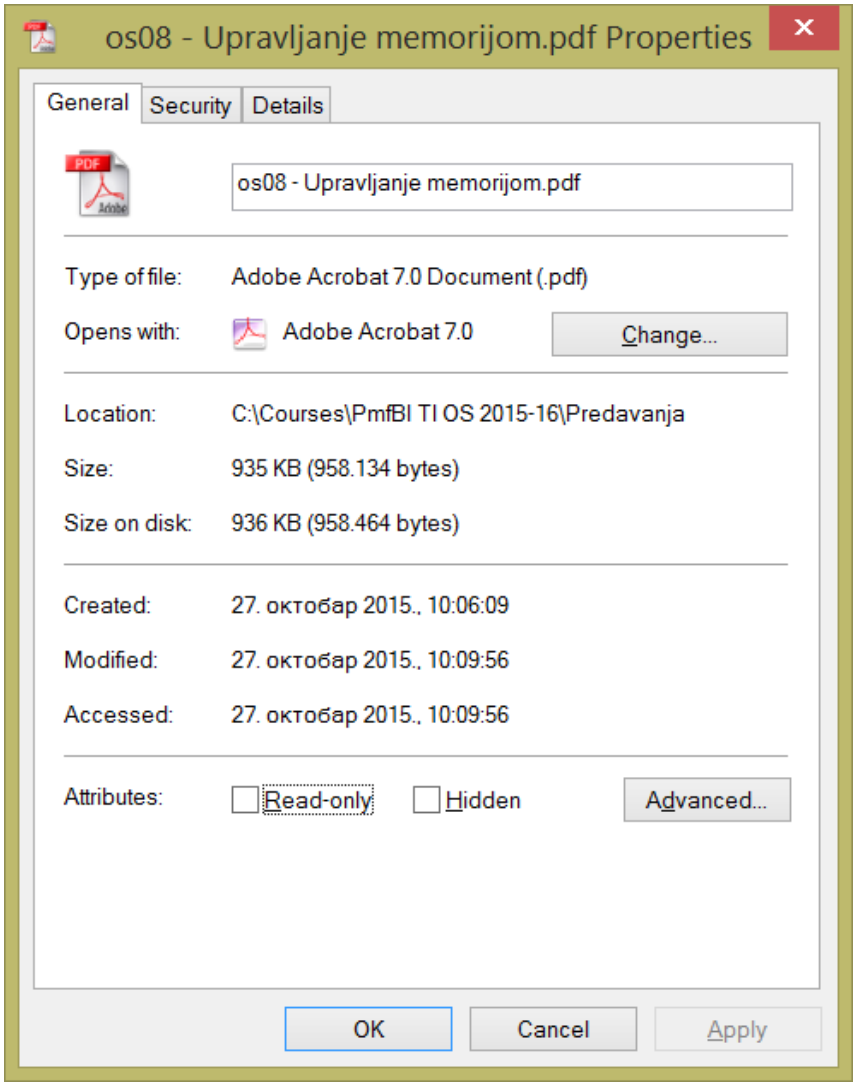
Sistem datoteka (4)

- Datoteka može sadržati podatke, ali i programe.
- Podaci u datotekama mogu biti zapisani u obliku teksta, ali i u drugim formama koje program za koji su namenjeni ume da pročita.
- Sadržaj datoteke je u opštem slučaju niz bajtova, a interpretacija tog niza zavisi od namene, preciznije programa koji ima zadatak da ga pročita.
- Sistem datoteka daje korisniku mogućnost imenovanja datoteka. Štaviše, svaka datoteka mora imati ime, koje postaje njena referenca. Tako datoteka prestaje da bude vezana za proces koji ju je kreirao i za sistem gde je nastala.

Atributi datoteka

- Skup atributa (osobina) koje sistem datoteka čuva o svakoj datoteci razlikuje se od sistema do sistema, ali su obično podržani sledeći:
  - Ime;
  - Lokacija;
  - Veličina;
  - Vreme kreiranja, poslednje modifikacije i poslednjeg pristupa;
  - Identifikator vlasnika fajla;
  - Prava pristupa;

Atributi datoteka (2)



Atributi datoteka (3)

- Neki od atributa su obavezni, kao što su ime fajla, ili njegova lokacija.
- Drugi atributi, kao što su vremena kreiranja ili modifikacije, nisu obavezni, pa od konkretnog sistema zavisi da li će biti podržani.
- Među podržanim atributima obično se nalazi i skup raznih flegova (zastavica).
- Fleg je obično jedan bit, i njegova vrednost se interpretira kao „uključeno“ ili „isključeno“.
  - Na primer, sistem datoteka može postojanje određenih fajlova sakriti od korisnika, a za to se obično koristi fleg sakrivenosti.

Atributi datoteka (4)

- Na osnovu atributa koji definišu prava pristupa, operativni sistem određuje ko može da pristupi datoteci i modifikuje je, a ko ne.
- Moguće su razne sheme, tako da se nekim korisnicima odredi samo pravo čitanja, drugima i čitanja i pisanja, ali ne izvršavanja, itd.

Atributi datoteka (5)

Atribut	Značenje
Kreator	Korisnik koji je kreirao fajl
Vlasnik	Aktuelni vlasnik fajla
Vreme kreiranja	Tačan datum i vreme kreiranja fajla
Vreme poslednjeg pristupa	Datum i vreme poslednjeg pristupa fajlu
Vreme poslednje promene	Datum i vreme poslednje promene fajla
Trenutna veličina	Broj bajtova fajla
Šifra	Lozinka za pristup fajlu
Sistemski fleg	0 ako je fajl običan a 1 ako je sistemski
ASCII fleg	0 ako je ASCII fajl a 1 ako je binarni
Fleg direktnog pristupa	0 ako je sekvencijalni a 1 ako je direktan pristup
Fleg za čitanje	0 ako se može i pisati a 1 ako je samo za čitanje
....	

Pregled atributa

Operacije nad datotekama

- Kroz sistemske pozive, sistem datoteka može pružiti razne funkcije, ali većina sistema datoteka podržava sledeće operacije:
  - Kreiranje – upisivanje nove datoteke;
  - Brisanje – uklanjanje postojeće datoteke;
  - Čitanje – čitanje informacije iz datoteke;
  - Pisanje – ažuriranje datoteke, bilo menjanjem već postojećih podataka, ili dodavanjem novih;
  - Repozicioniranje – pomeranje pokazivača na određeni deo datoteke kako bi se isti ažurirao;
  - Skraćivanje – brisanje sadržaja datoteke, tako da svi atributi ostaju

Operacije nad datotekama (2)

- Druge operacije se mogu izvesti kombinovanjem ovih osnovnih.
  - Na primer, datoteka se može iskopirati kreiranjem nove datoteke i kopiranjem sadržaja stare datoteke.
- Pre korišćenja datoteke, proces je mora otvoriti. Za ovo se koristi poseban sistemski poziv koji služi da se datotka označi kao otvorena, inicijalizuju potrebne strukture, učitaju atributi i adrese na kojima se čuvaju podaci.

Operacije nad datotekama (3)

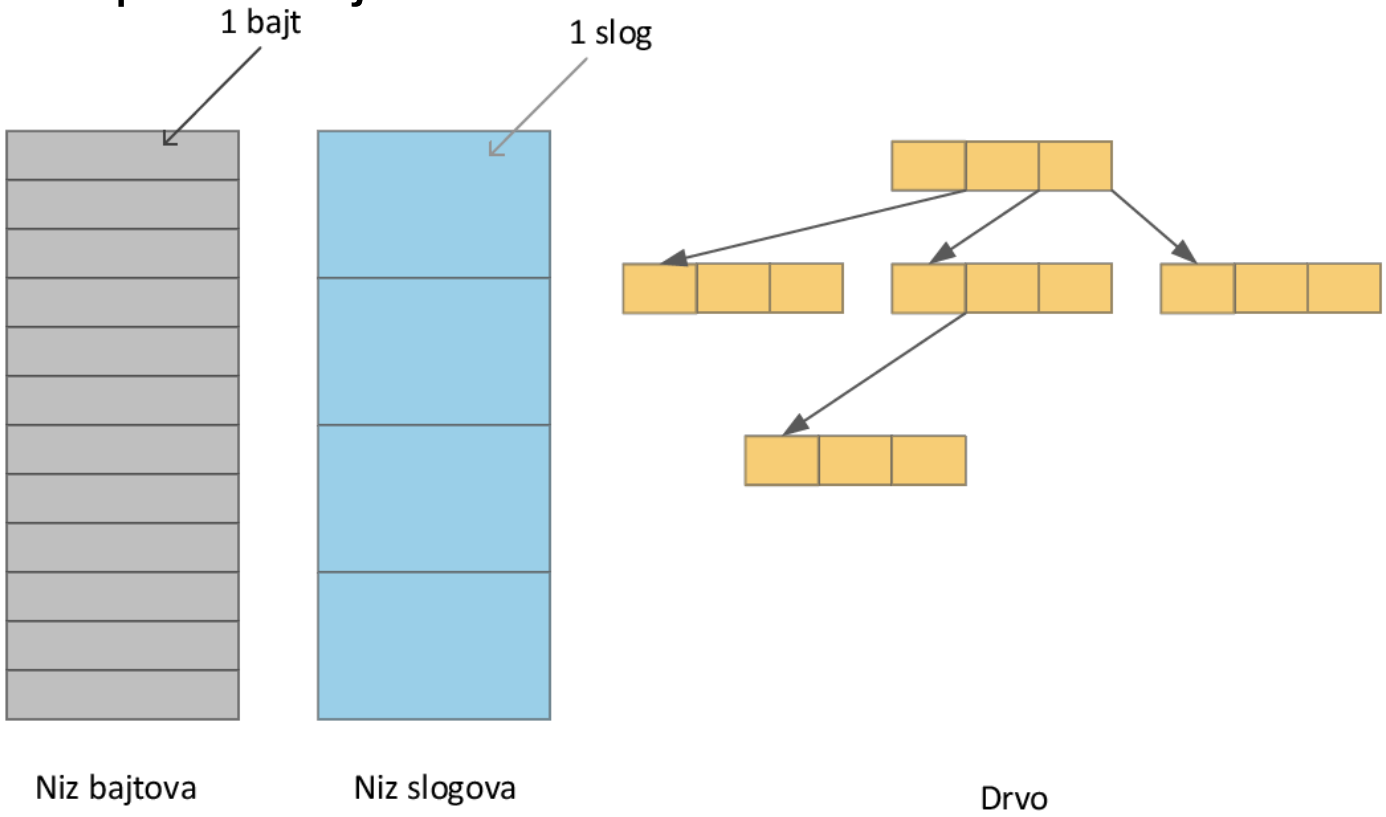
- Kada više nije potrebna procesu, datoteka se zatvara korišćenjem sistemskog poziva za tu namenu.
  Datoteka se zatvaranjem briše iz tabele otvorenih i oslobađaju se strukture koje više nisu potrebne.
- Procesima je potrebno obezbediti mogućnost da čitaju, upisuju i menjaju atribute datoteka.
  - Na primer, da bi program sortirao datotke po veličini, on mora biti u stanju da pročita taj atribut.
  - S druge strane, program može imati potrebu da neke datoteke označi kao skrivene, pa mu se mora pružiti mogućnost da modifikuje atribute.

Struktura datoteka

- Operativni sistemi datoteke mogu struktuirati na više načina:
    1. Niz bajtova je jedna od čestih organizacija. Sva kontrola nad sadržajem datoteke prepuštena je programu koji sa njime manipuliše.
       Operativni sistem vidi samo niz bajtova i ne razume unutrašnju strukturu. Ovakav pristup je veoma fleksibilan.
    2. U ovoj organizaciji datoteke se podele u slogove fiksirane veličine sa tačno određenom strukturom. Na ovaj način datoteka postaje sekvenca slogova. Čitanje i pisanje se obavlja na nivou sloga.

Struktura datoteka (2)

3. U ovoj organizaciji se datoteka implementira kao stablo.
   Ovo ubrzava pretragu podataka, ali je implementacija komplikovanija.



Tipovi datoteka

- Pri dizajniranju operativnog sistema, postavlja se pitanje da li će sistem biti napravljen tako da prepoznaje i podržava tipove fajlova.
  - Ako sistem prepoznaje određeni tip, onda će znati kako da rukuje određenim fajlom, tj. na koji način da interpretira sadržane informacije.
  - Na primer, ako bi se fajl koji sadrži video informacije otvorio u tekst editoru, dobio bi se besmislen tekst, jer editor čita informacije iz fajla i interpretira ih kao tekst.
  - Ovo je moguće, jer je fajl niz bajtova. Kako će se oni interpretirati i da li će se interpretirati na predviđen način ili ne zavisi od načina na koji ih korisnik koristi.

Tipovi datoteka (2)

- Često se primenjuje konvencija da se pored imena, na kraj fajla doda tačka i za njom kratka ekstenzija (nastavak) koja govori o tipu.
  - Ovo je korisno i korisniku i operativnom sistemu, jer se na osnovu ekstenzije „zna“ koje operacije mogu da se izvrše nad datotekom i na koji način se podaci čitaju.
  - Operativni sistem može na osnovu ekstenzije da definiše podrazumevani program za otvaranje određenog tipa datoteke.
  - Na primer, može se definisati da tekst editor otvara sve datoteke sa ekstenzijom .txt
  - Ekstenzija je samo nagoveštaj koja ne garantuje tip datoteke. Nije obavezno da se ekstenzija sadrži u imenu, ali, ako je prisutna i tačno navedena, olakšava rad.

Tipovi datoteka (3)

Ekstenzija	Vrsta fajla
.html , .htm, .xml, .tex	Etiketirani dokumenti
.txt	Tekstualni dokument
.jpg, .png, .tif,	Slika
.c, .cpp, .cs, .java, .asm	Kod programa
.exe, .jar, .com, .bin	Izvršni fajlovi
.o	Objektni fajlovi
.lib, .a, .so, .dll	Biblioteke za programe
.mpg, .avi, .flv, .mpeg, .mov	Video - Multimedija
.mp3	Muzika
.pdf, .ps,	Dokumenti koji se mogu pregledati nezavisno od sistema
.zip, .rar, .tar	Komprimovane arhive
.doc, .docx, .rtf	Formati jezičkih procesora
.bat, .sh	Beč (batch) fajlovi

Standardne ekstenzije

Tipovi datoteka (4)

- U nekim sistemima vrsta datoteke se određuje na osnovu niza bitova koji se nalazi na početku datoteke i koji se naziva se magični broj.

Izvršni fajl (Microsoft DOS, Windows, OS/2)	.exe	4d 5a
Izvršni fajl (Unix)		7f 45 4c 46
Arhivirani fajl (ZIP, JAR, ODF, OOXML)	.zip, .jar, .odt, .ods	50 4b 03 04 ili 50 4b 07 08
Arhivirani fajl (RAR)	.rar	52 61 72 21 1A 07 00
TAR (POSIX)	.tar	75 73 74 61 72
JPEG File Interchange Format	.jpg	ff d8 ff e0
Bit mapa	.bmp	42 4d
GIF format slike	.gif	53 49 4d 50 4c 45
PNG format	.png	89 50 4e 47

Magični brojevi (Heksadekadni zapis)

Tipovi datoteka (5)

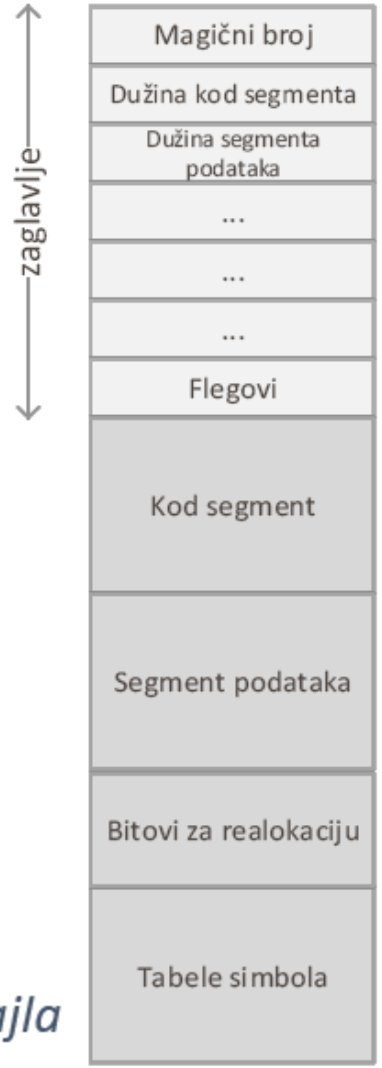
- Operativni sistemi obično podržavaju nekoliko tipova datoteka:
  1. Skoro na svim sistemima postoje regularne datoteke koje čuvaju informacije korisnika.
  2. Direktorijumi (folderi) su sistemske datoteke koje odražavaju strukturu datotečkog sistema.
  3. Na nekim sistemima postoje i interfejsi drajvera uređaja koje operativni sistem vidi kao obične datoteke, radi lakšeg rukovanja - time se postiže da program manipuliše uređajem koristeći iste sistemske pozive kao da se radi o datoteci.
     To su karakter-specijalne datoteke za serijske uređaje (štampač, mreža, itd.) i blok-specijalne datoteke, sa mogućnošću čitanja/upisa bloka date veličine.

Regularne datoteke

- Regularna datoteka može biti ASCII ili binarna.
  - ASCII datoteka se sastoji od niza znakova. Oni se ne moraju posebno interpretirati, jer univerzalnost ASCII formata omogućava da se mogu obrađivati u većini editora i da ih bez modifikacije mogu koristiti različiti programi.
  - Ostale tipovi datoteka se nazivaju binarnim. Binarne datoteke se ne mogu interpretirati bajt po bajt i imaju strukturu poznatu programu koji njima rukuje.
  - Tipična binarna izvršna datoteka sastoji se od zaglavlja, kod segmenta, segmenta podataka, bitova za relokaciju i tabele simbola koja se koristi pri debagiranju (otklanjanju grešaka).

Regularne datoteke (2)

- Magični broj je prvi podatak u zaglavlju i on govori operativnom sistemu da se radi o izvršnom fajlu.
- Zatim slede podaci o dužini raznih delova fajla, i na kraju određeni flegovi.
- Nakon zaglavlja slede podaci samog programa. U slučaju izvršne datoteke, OS poznaje njenu unutrašnju strukturu, jer je upravo on i odgovoran za njeno pokretanje.



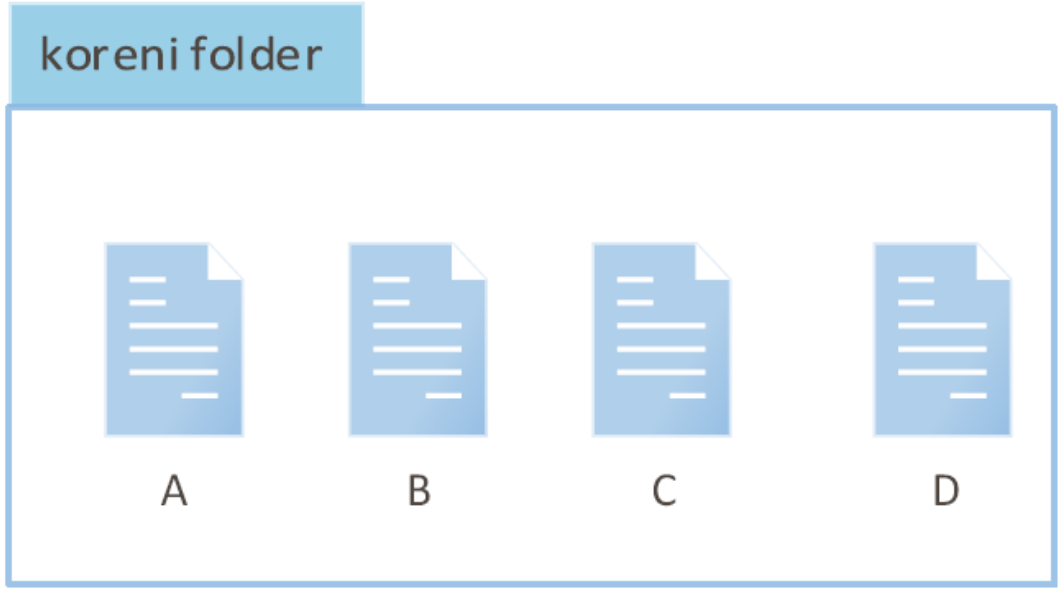
Sastavni delovi izvršnog fajla

Direktorijumi

- Direktorijum (folder, katralog) se može pojednostavljeno zamisliti kao tabela sa spiskom datoteka koje „sadrži“.
- I sam direktorijum je datoteka.
- Datotečki sistem bi trebao da omogući nekoliko osnovnih operacija nad direktorijumom:
  - Kreiranje;
  - Brisanje;
  - Izlistavanje svih datoteka u direktorijumu;
  - Dodavanje nove datoteke;
  - Brisanje datoteke.

Direktorijumi (2)

- Najjednostavnija organizacija podrazumeva da postoji jedan koreni (root) direktorijum u kom se nalaze sve datoteke.
- Na prvim računarima sa malo datoteka i sa jednim korisnikom ovo je bilo prihvatljivo.



Organizacija sa jednim korenim folderom

Direktorijumi (3)

- Karakteristike jednonivoske organizacije:
  - Jednostavna je za implementaciju.
  - Vrlo je ograničena, jer dve datoteke ne mogu imati isto ime.
  - Svaki korisnik, ili aplikacija, bi morao pre kreiranja datoteke da proveri da li datoteka sa tim imenom već postoji i da, ako je to slučaj, odabere drugo ime.

Direktorijumi (4)

- Organizacija na dva nivoa je unapređenje, u kojem se svakom korisniku dodeli poseban folder.
- Dakle, postoje dva nivoa, prvi direktorijum je koreni, a unutar njega postoji direktorijum za svakog od korisnika.

Organizacija foldera u dva nivoa

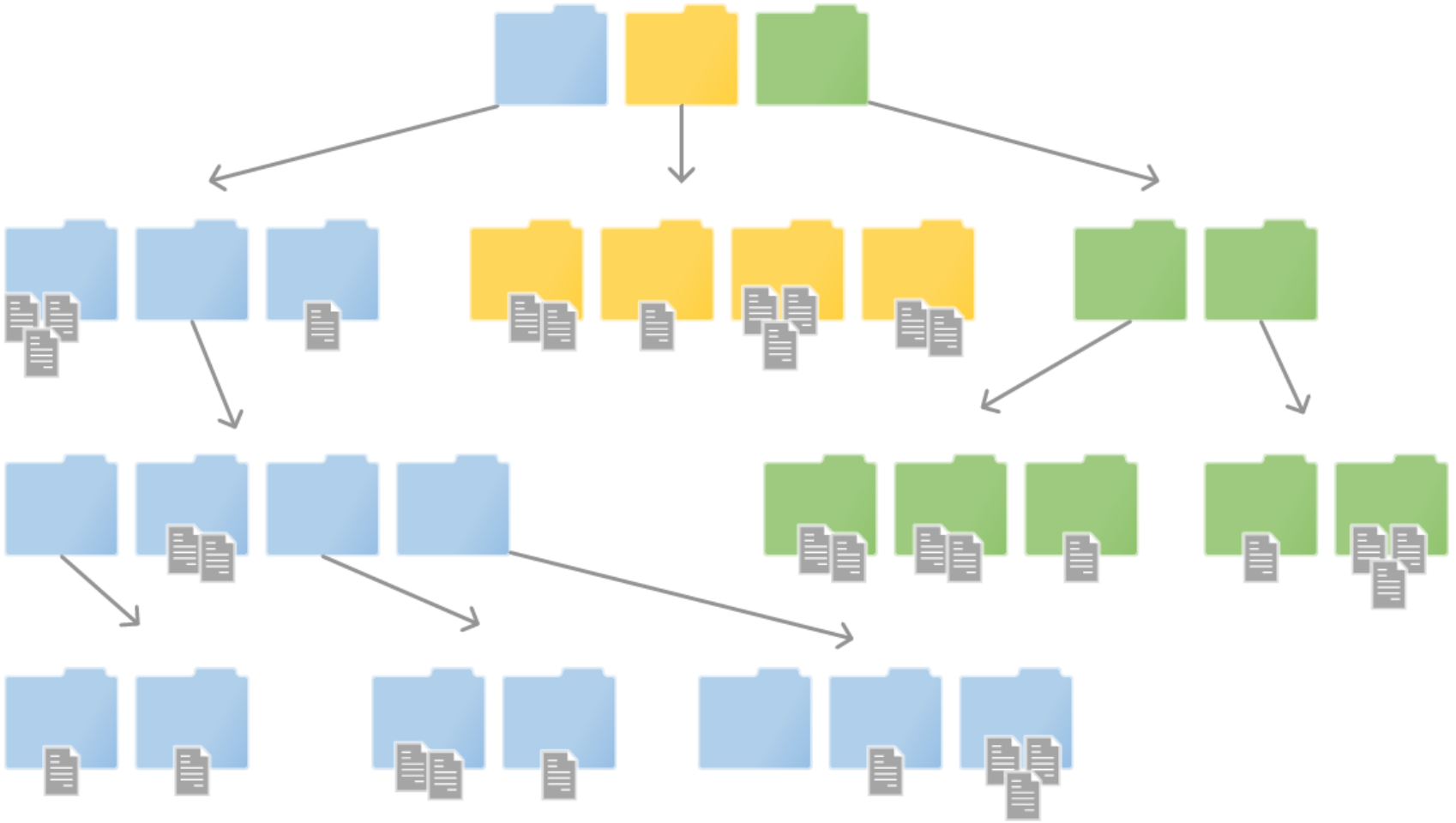
Direktorijumi (5)

- Karakteristike dvonivoske organizacije:
  - Iako je ovo bolje rešenje od organizacije sa jednim nivoom.
  - Međutim, problem iz prethodne organizacije je nasleđen - jedan korisnik može imati veliki broj fajlova i traženje novog imena za svaki novi je neprihvatljivo.
  - Sa druge strane, svaki korisnik je izolovan ako mu OS ne dozvoli da pristupa folderima drugih korisnika.

Direktorijumi (6)

- Generalizacija prethodne ideje je pristup da se direktorijumi organizuju u strukturu drveta (stabla).
- Svaki direktorijum može sadržati datoteke ali i druge direktorijume, bez ograničenja dubine.
- Pri tome, svaka datoteka ima jedinstvenu putanju, od korenog direktorijuma do same te datoteke i mora imati jedinstveno ime samo u poslednjem folderu kojem pripada.
- Većina modernih sistema je organizovana na prethodno opisani način.

Direktorijumi (7)



Organizacija foldera strukturom stabla

Direktorijumi (8)

- U mnogim operativnim sistemima procesi imaju radni (trenutni) direktorijum u kom se izvršavaju.
- Tu se, uglavnom nalaze datoteke potrebne za izvršavanje procesa.
- Radni direktorijum se može promeniti i za to je obično obezbeđen sistemski poziv koji kao parametar prima putanju do novog.
- Ako je potrebno pristupiti datoteci koji se ne nalazi u radnom direktorijumu, jedna opcija bila bi da se radni direktorijum promeni u onaj direktorijum gde se tražena datoteka nalazi, pa da se potom pristupi traženoj datoteci.

Direktorijumi (9)

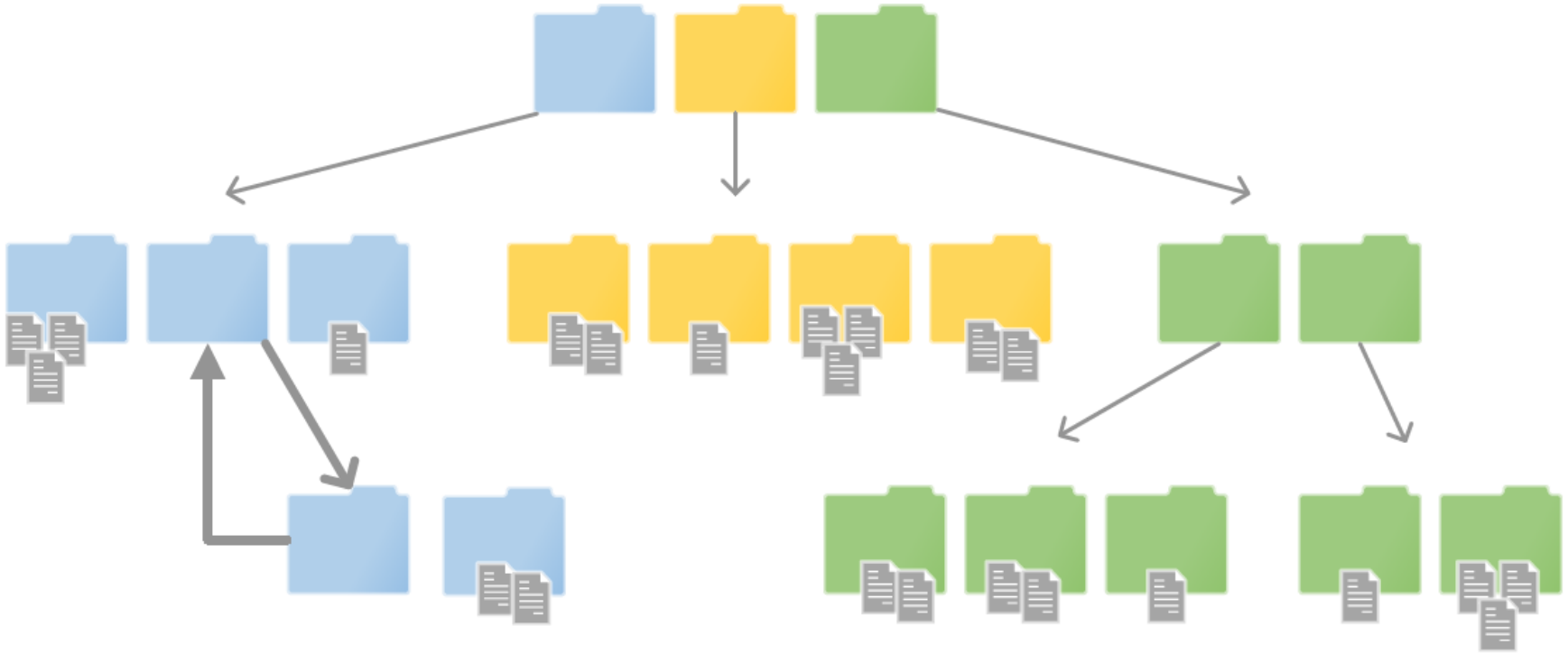
- Druga opcija je da se navede apsolutna putanja do tražene datoteke, tj. putanja koja počinje korenim direktorijumom sistema i završava se imenom datoteke, navodeći sve poddirektorijume koji čine putanju.
- Da bi korisnik „došao“ do tražene datoteke, on mora otvoriti redom sve direktorijume od korenog do krajnjeg u kom se nalazi datoteka.
- Apsolutna putanja je, niz imena direktorijuma koji se moraju proći da bi se došlo do određene datoteke, počevši od korenog direktorijuma, i koji se završava imenom tražene datoteke.

Direktorijumi (10)

- Relativna putanja je deo apsolutne putanje koja počinje radnim direktorijumom.
- Ako se ona navede, operativni sistem sam uradi nastavljanje do apsolutne putanje kako bi našao tražene datoteke.
- Da bi korisnik mogao lakše da referiše na datoteke i direktorijume korišćenjem relativnih adresi, uvedena su dva specijalna direktorijuma koja se implicitno nalaze u svakom direktorijumu. Njihova imena su „.“ i „..“ (tačka i tačka-tačka).
- Tačka je referenca za sam direktorjum, dok je tačka-tačka referenca za roditeljski direktorijum.

Direktorijumi (11)

- Uvođenjem specijalnih direktorijuma „.“ i „..“, struktura direktorijuma više nije drvo, već postaje graf koji može imati cikluse.



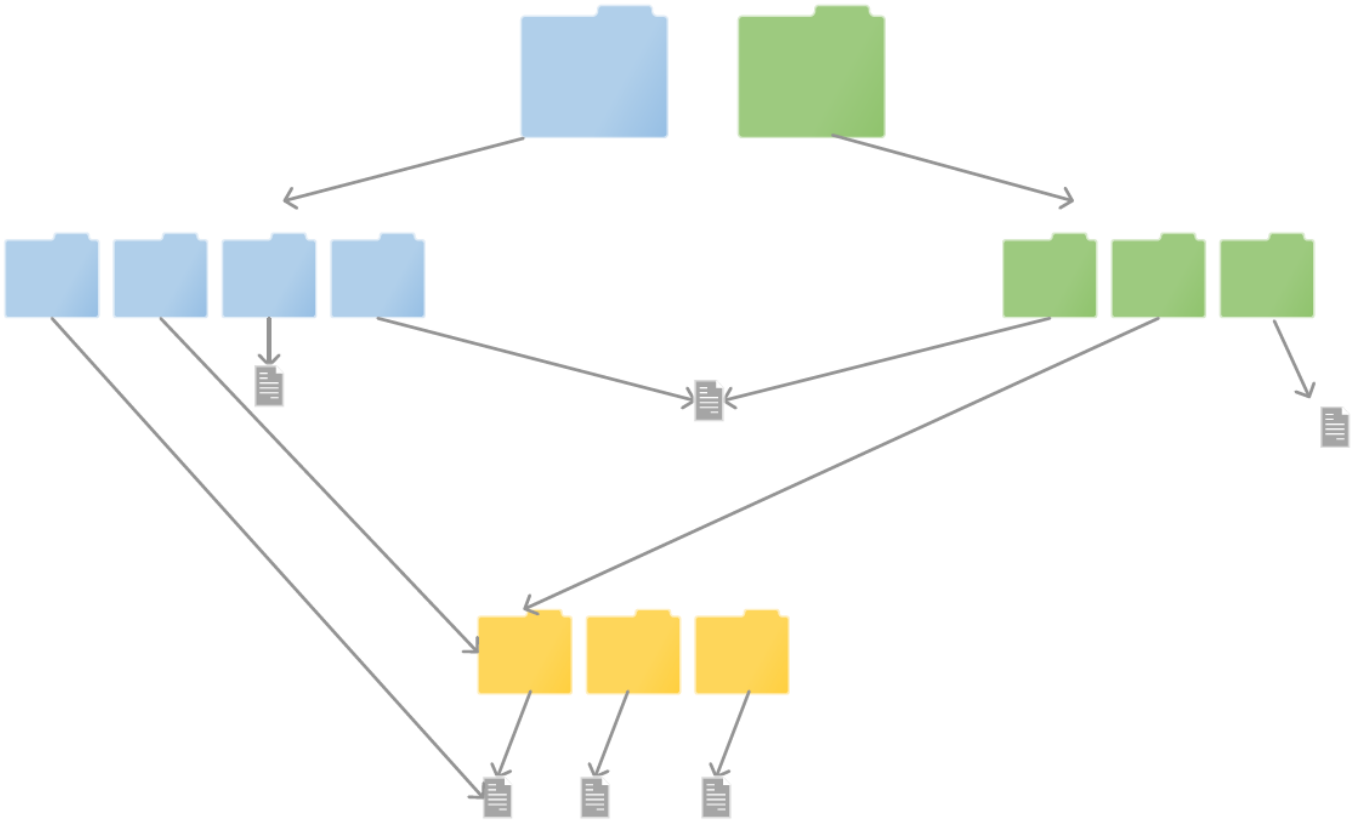
Apsolutne i relativne putanje

Deljenje datoteka

- Često više korisnika treba da deli datoteke.
  - Na primer, ako korisnici rade na istom projektu, zgodno je da datoteke budu u direktorijumima svakog od korisnika, a da se, ako neko izmeni datoteku, te promene vide i kod drugih korisnika.
- Problem se može rešiti uvođenjem linkova (veza). Link je niska unutar datoteke koja sadrži putanju do neke druge datoteke.
  - Rešenje za deljenje datoteka korišćenjem linkova je da samo jedan korisnik poseduje originalnu datoteku, a svi ostali veze prema njoj. Tako svi korisnici, osim jednog, sadrže samo link prema datoteci koja fizički postoji samo u jednom direktorijumu. Pri otvaranju linka, OS ne otvara njegov sadržaj, već prati putanju do originalne datoteke.

Deljenje datoteka (2)

- Brisanje linkova ne utiče na originalnu datoteku. Brisanje datoteke čini da svi linkovi ka njoj postanu nevalidni, jer pokazuju na fajl koji više ne postoji.



Prava pristupa

- Prava pristupa (access permissions) predstavljaju prava koja korisnik može da ostvari nad datotekom.
- Zaštita podataka na većini sistema zasniva se na kontrolisanom pristupu podacima koji se ostvaruje kroz prava pristupa.
  - Kod jednokorisničkih sistema, mehanizam zaštite se može realizovati veoma jednostavno, jer se ceo sistem može zaključati šifrom - čime je sigurnost obezbeđena.
  - Kod višekorisničkih sistema, stvar je komplikovanija, jer najčešće nije cilj izolovati korisnike time što bi svako imao pristup samo svojim datotekama, već treba omogućiti fleksibilniji mehanizam koji bi određenim korisnicima odobravao pristup a drugima ne.

Prava pristupa (2)

- Nad datotekama i direktorijumima najčešće se definišu sledeća tri tipa pristupnih dozvola:
  - Dozvola za čitanje;
  - Dozvola za pisanje;
  - Dozvola za izvršavanje.

Prava pristupa (3)

- Prethodne dozvole nemaju isto značenje kada se primenjuju na datoteke i na direktorijume.
  - Dozvola za čitanje u slučaju datoteke znači da korisnik može da pristupa podacima datoteke i da ih čita dok kod direktorijuma ova dozvola omogućava izlistavanje stavki koje on sadrži.
  - Dozvola za pisanje kada su datoteke u pitanju dozvoljava dodavanje novih podataka, a kod direktorijuma se odnosi na dodavanje i brisanje datoteka i podirektorijuma koji se u njemu nalaze.
  - Dozvola za izvršavanje omogućava korisniku da izvrši datoteku, dok je kod direktorijuma ta dozvola neophodna za druge dozvole odnosno, u zavisnosti od sistema, bez nje se ne mogu primeniti neke dozvole.

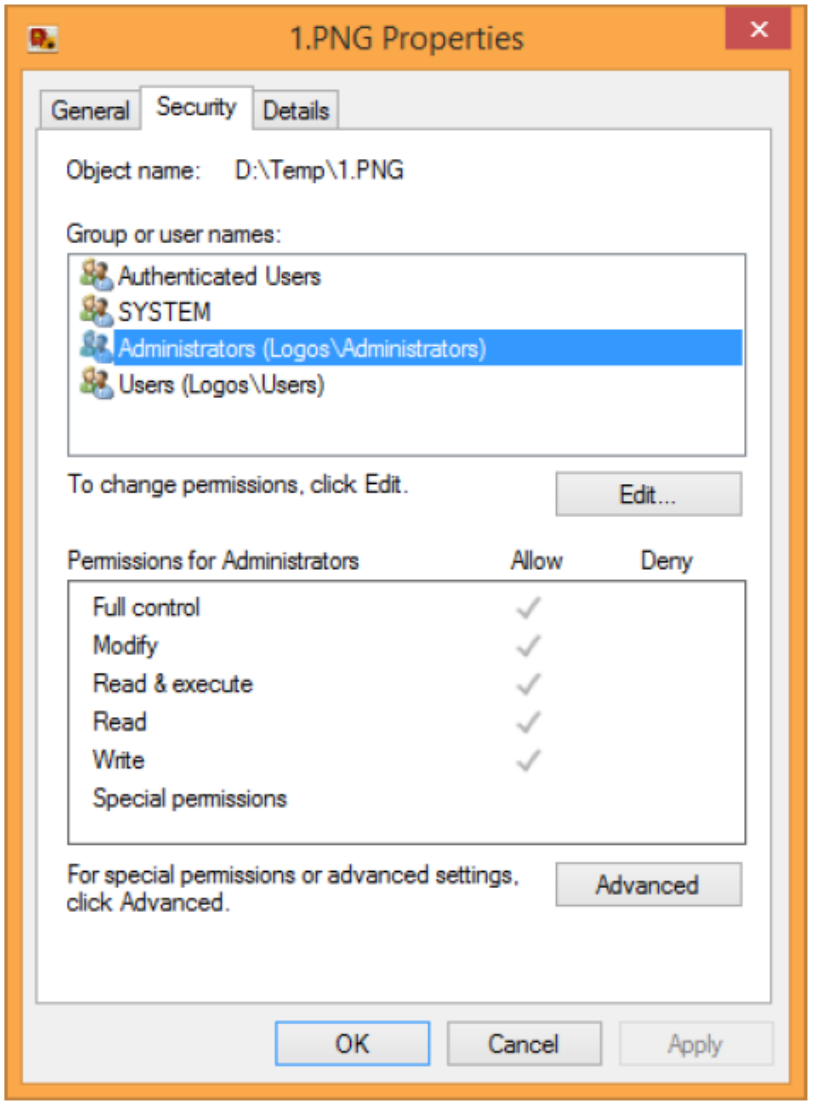
Prava pristupa (4)

- Problem pristupa se može rešiti tako što će se za svaku datoteku održavati lista svih korisnika sa njihovim dozvolama.
  - Time se postiže najveća fleksibilnost, jer se za svakog ponaosob mogu definisati prava.
  - Međutim, korisnika može biti puno, i održavati njihovu listu za svaki fajl nije pogodno.
- Drugi pristup podrazumeva da se svi korisnici podele u grupe na osnovu zadataka koje izvršavaju.
    - Grupe omogućavaju definisanje posebnih pristupnih dozvola za različite korisnike.

Prava pristupa (5)

- Često se primenjuje pristup u kom se (iz ugla datoteke) korisnici svrstati u tri grupe: vlasnik, članovi grupe i ostali.
  - Vlasnik je korisnik koji je kreirao datoteku i on ima prava da drugima određuje prava pristupa.
  - Članovi grupe su korisnici koji pripadaju grupi u kojoj je vlasnik datoteke
  - Ostali su svi ostali korisnici koji nisu u grupi sa vlasnikom.
  - Obično vlasnik fajla ima sva prava kao i administrator, grupe imaju restriktivnija prava, a ostali još manje privilegija. Ovako je postignut kompromis, gde je kompleksnost implementacije daleko manja, a fleksibilnost je velikoj meri zadržana.

Prava pristupa (6)



Prava pristupa (7)

```
drwxr-xr-x 2 root    root         120 Aug 16 19:35 ConsoleKit
-rw-r--r-- 1 root    root       33315 Nov 10 15:05 Xorg.0.log
-rw-r--r-- 1 root    root       18069 Oct 24 23:00 Xorg.0.log.old
drwxr-x--- 2 apache  apache       256 Oct 16 00:44 apache2
drwxr-xr-x 2 couchdb couchdb       48 Apr  4  2012 couchdb
drwxr-xr-x 2 root    root         176 Aug  4 10:12 cups
-rw-r----- 1 root    root      146132 Oct 24 23:01 dmesg
-rw-rw---- 1 portage portage     1600 Oct 25 00:04 emerge-fetch.log
-rw-rw---- 1 portage portage  1745844 Nov 17 04:16 emerge.log
drwxr-xr-x 2 root    root          88 Sep 29 04:11 gdm
-rw-r--r-- 1 root    root       71103 Oct 24 23:02 kdm.log
-rw-r--r-- 1 root    root      292876 Oct 24 23:02 lastlog
-rw------- 1 root    root    14226778 Nov 17 14:40 messages
drwxr-xr-x 2 mysql   mysql         80 Jul  6 00:25 mysql
-rw-r--r-- 1 root    root       58703 Oct 24 23:03 pm-powersave.log
```

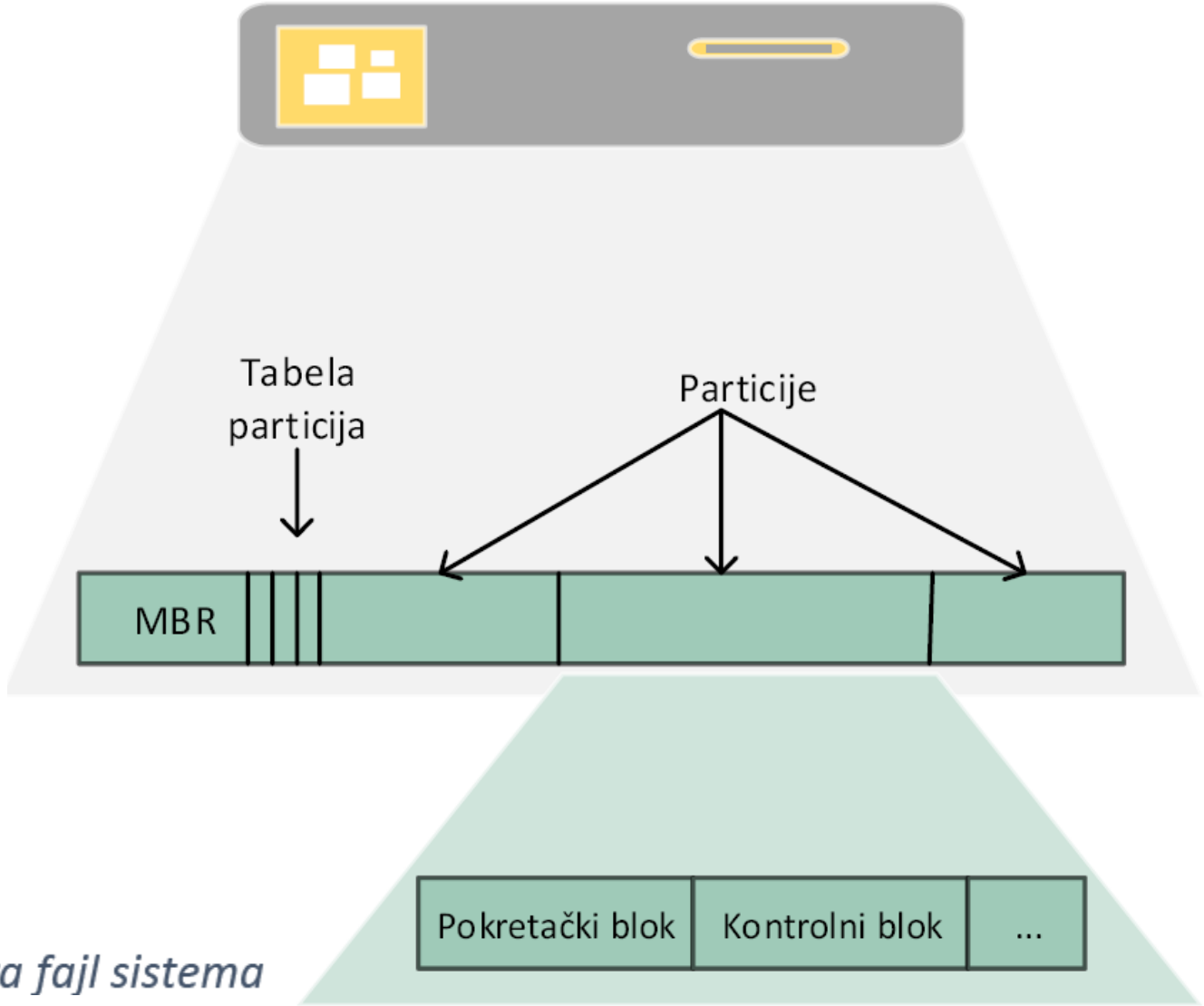
Implementacija sistema datoteka

- Datotečni sistemi su smešteni na neki vid sekundarne memorije, koja se obično naziva disk, bilo da je reč o magnetnom disku ili nekom drugom uređaju.
- Većina diskova se može podeliti na particije, celine koje su logički predstavljene kao posebni diskovi i koje omogućavaju da se na svakoj koristi različit sistem datoteka.
- Prvi sektor diska se naziva MBR (Master Boot Record) i podaci koji su tu zapisani (MBR program) su prvi koji se učitavaju i izvršavaju pri pokretanju računara.

Implementacija sistema datoteka (2)

- MBR sadrži i tabelu particija u kojoj je zapisan početak i kraj svake od njih.
- Jedna particija je u tabeli označena kao aktivna, i prvo što MBR program izvršava jeste nalaženje aktivne particije i učitavanje njenog prvog bloka, koji se naziva pokretački blok (boot block).
- Nakon toga, izvršava se pokretački blok aktivne particije tj. učitavanje i pokretanje OS.
- Posle pokretačkog bloka, sledi kontrolni blok koji sadrži informacije vezane za particiju kao što su ukupan broj blokova, veličina blokova, broj slobodnih blokova, itd.

Implementacija sistema datoteka (3)



Struktura fajl sistema

Implementacija datoteke

- Svaka memorija ima definisanu najmanju jedinicu koja može biti alocirana i adresirana. Takve jedinice zovu se blokovi.
  - Na primer, magnetni diskovi obično koriste blokove veličine 512 bajta ili 4 kilobajta.
- Svaka datoteka koja se kreira predstavljena je kao sekvenca blokova koji su joj dodeljeni.
- Ključni problem, pri implementaciji datoteka, predstavlja način na koji se alociraju slobodni blokovi i kako se čuva informacija koji blokovi pripadaju kojoj datoteci.

Neprekidna alokacija

- Neprekidna alokacija je najjednostavniji pristup pri kojem se svaka datoteka čuva kao neprekidni niz blokova.
  - Na primer, ako je datoteka veličine n blokova i ako počinje na lokaciji b, onda ta datoteka zauzima blokove b, b+1, b+2, ..., b+n-1.
- Sledeća datoteka se smešta odmah iza poslednjeg bloka prethodnog fajla.
- Ovakva organizacija ubrzava sekvencijalno čitanje sa uređaja kao što su magnetni diskovi, jer zahteva minimalno pomeranje glave za čitanje.

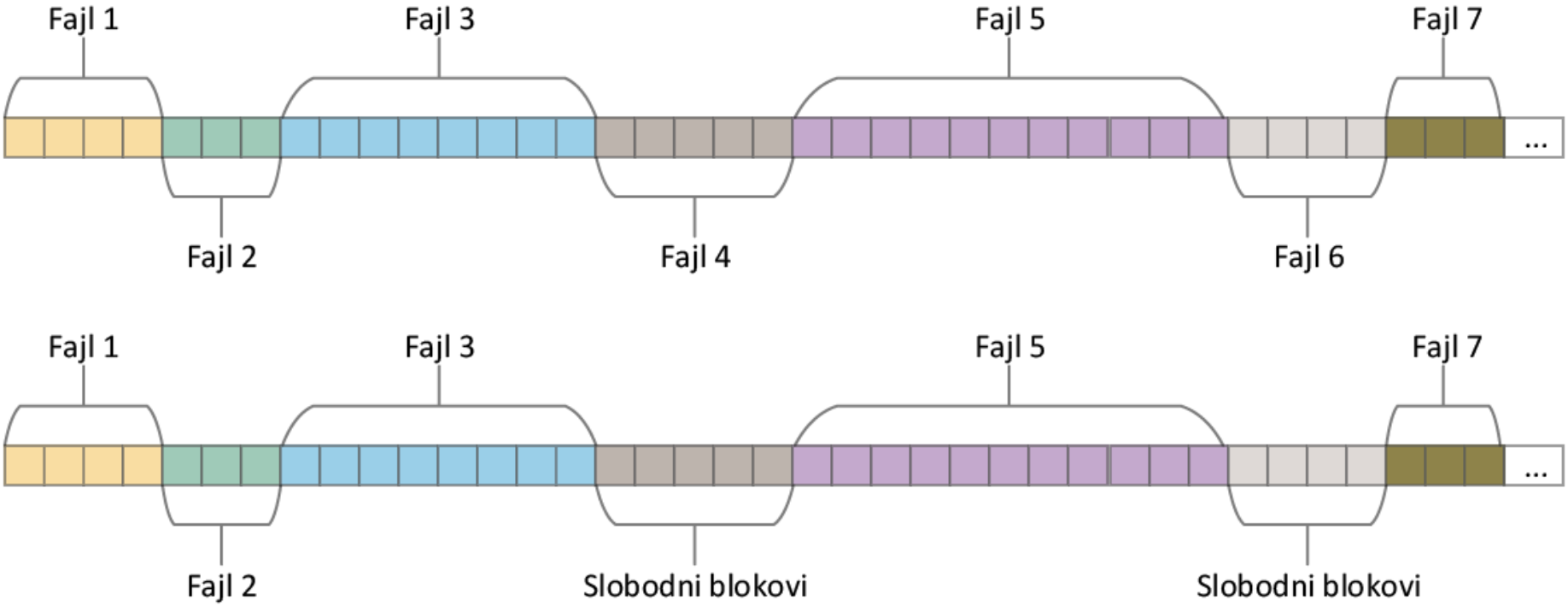
Neprekidna alokacija (2)

- U ovakvoj organizaciji je takođe olakšan i direktan pristup delu datoteke, jer ako fajl počinje blokom b, pristup i-tom bloku fajla je pristup bloku b+i u sistemu.
- U okviru direktorijuma će se za svaku datoteku čuvati samo početni blok i broj blokova koje ta datoteka zauzima.

Neprekidna alokacija (3)

- Ovakva organizacija ima ozbiljne probleme pri brisanju datoteka, jer dolazi do fragmentacije.
  - Ako se obriše datoteka, na njenom mestu ostaje „rupa“ tj. slobodni prostor. Te rupe se popunjavaju novim datotekama, ali se može desiti da sistem ima dovoljno slobodnog prostora a da ne postoji nijedan neprekidni niz slobodnih blokova dovoljan za smeštanje nove datoteke. Ovaj problem se naziva eksterna fragmentacija.
  - Drugi problem predstavlja interna fragmentacija. Naime, pošto datoteka mora zauzeti ceo broj blokova, poslednji blok je često neiskorišćen do kraja.

Neprekidna alokacija (4)



Neprekidna alokacija

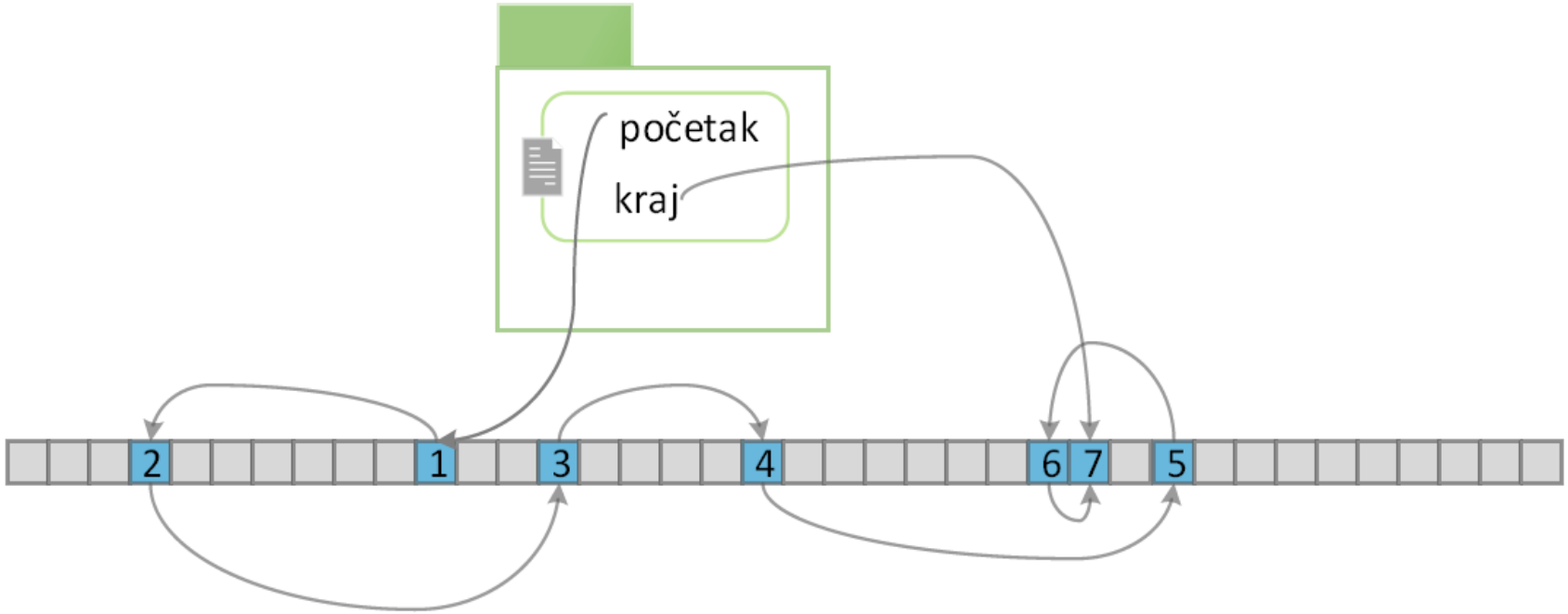
Neprekidna alokacija (5)

- Veći problem leži u činjenici da nije moguće pretpostaviti koliko će datoteka da raste u budućnosti a samim tim i alocirati dovoljno slobodnog prostora na kraju datoteke.

Alokacija preko povezanih listi

- Problemi neprekidne alokacije se rešavaju preko povezanih listi.
- Svaka datoteka je predstavljena kao povezana lista blokova.
- Blok osim informacije, sadrži i pokazivač na sledeći blok, tj. polje sa njegovom adresom. Poslednji blok datoteke ima kao pokazivač na sledeći blok neku nevalidnu vrednost (obično null).
- U direktorijumu se za svaku datoteku tog direktorijuma čuva samo adresa prvog bloka datoteke.

Alokacija preko povezanih listi (2)



Povezane liste

Alokacija preko povezanih listi (3)

- Realizacija operacija kreiranja datoteke i upisa u datoteku:
  - Nova datoteka se kreira tako što se u direktorijum doda novi pokazivač na prvi blok novokreirane datoteke. Taj pokazivač na početku može sadržati nevalidnu vrednost, kako bi se označilo da je novokreirana datoteka prazna.
  - Upis u (dosad praznu) datoteku se vrši tako što sistem nađe slobodni blok, u taj blok upiše podatke, pokayivač na sledeći u okviru bloka postavi na invalidnu vrednost, markira blok da je zauzet i adresu tog bloka upisuje u pokazivač i za početak i za kraj koji se nalazi u direktorijumu.
  - Sledeći upisi bi se realizovali sa slobodnim blokovima nadovezanim na kraj liste itd.

Alokacija preko povezanih listi (4)

- Prednosti ovog pristupa su:
  - Novi blok se može nadovezati u datoteku bilo gde da je fizički smešten - blokovi jedne datoteke mogu biti razbacani svuda po disku.
  - Datoteka se čita blok po blok, uvek čitajući adresu narednog bloka iz pokazivača trenutnog, prolazeći kroz listu, čime se u potpunosti rešava problem eksterne fragmentacije.
  - Datoteka može rasti sve dok postoji slobodnih blokova, tako da veličina fajla ne mora biti deklarisana na početku. Na ovaj način je rešen i najveći problem interne fragmentacije.

Alokacija preko povezanih listi (5)

- Međutim, postoje i mane ovog pristupa:
  - Povezana lista nije efikasna kada je potreban direktan pristup delu fajla.
    - Da bi se došlo do i-tog bloka, mora se krenuti od početnog i pratiti svaki pokazivač, jedan po jedan, kroz svaki blok, sve dok se ne dostigne i-ti.
    - Magnetni diskovi su pri ovakvoj organizaciji neefikasni i zbog činjenice da je, u opštem slučaju, potrebno pomerati glavu za svaki blok. Novije vrste sekundarne memorije, kao što je SSD, nemaju ovakav problem.

Alokacija preko povezanih listi (6)

- Mane povezanih listi:
  - Osim same informacije, u svakom bloku mora smestiti i pokazivač, što dodatno zauzima prostor.
    - Ovo se donekle mogu rešiti grupisanjem blokova (clustering) i alociranjem grupa umesto pojedinih blokova.
    - Na primer, fajl sistem može definisati grupu kao četiri bloka i odrediti nju kao najmanju jedinicu prostora koja se alocira. Svaka grupa bi se mogla shvatiti kao neprekidna alokacija četiri bloka.
    - Ovakvim kompromisom štedi prostor – manje je polazivača, ali se povećava interna fragmentacija.

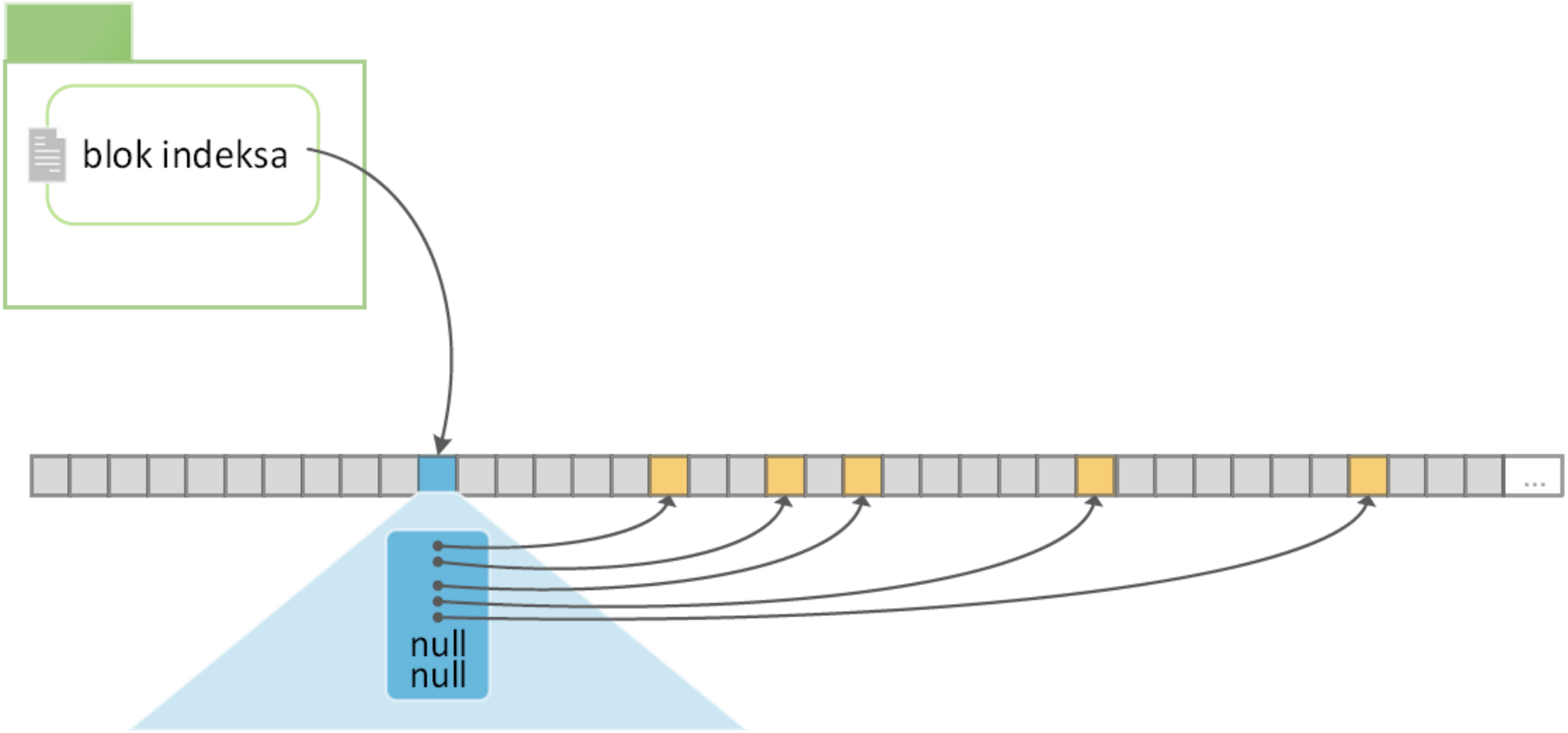
Alokacija preko povezanih listi (7)

- Mane povezanih listi:
  - Pouzdanost je veliki problem kod povezanih listi
    - Može se dogoditi da greška pri upisu, prestanak napajanja, hardverska greška, itd. prouzrkuje da jedan pokazivač sadrži pogrešnu vrednost.
    - Ovakva greška bi uzrokovala da se fajl nadovezuje na slobodan prostor, ili na neki drugi fajl, narušavajući konzistentnost celog fajl sistema.
    - Ovo se može delimično rešiti dvostruko povezanim listama, ili dodavanjem redundantnosti na drugi način, ali to je prostorno još neefikasnije.

Indeksirana alokacija

- Povezane liste rešavaju važne probleme, ali uvode novi problem jer ne pružaju efikasan direktan pristup, pošto su pokazivači sadržani u samim blokovima.
- Ovaj problem se rešava uz pomoć indeksirane alokacije, koja sve pokazivače smešta na jedno mesto, koje se naziva blok indeksa.
- Svaka datoteka sadrži blok indeksa, a on sadrži niz adresa blokova u koji su smešteni podaci date datoteke.
- U direktorijumu se čuva adresa bloka indeksa za svaku datoteku tog direktorijuma.

Indeksirana alokacija (2)



Indeksirana alokacija

Indeksirana alokacija (3)

- Kod ovakvog pristupa nema eksterne fragmentacije, ali se zato više prostora troši na pokazivače.
  - Na primer, ako fajl alocira samo nekoliko blokova, da bi se sačuvalo tih nekoliko adresa, mora se alocirati ceo blok indeksa. U njemu će samo nekoliko polja biti popunjeno adresama, a ostala će ostati neiskorišćena.
- Veoma je važno odrediti koja je veličina najpogodnija za veličinu bloka indeksa.
  - Ako se izabere preveliki blok, prostor se neefikasno troši.
  - S druge strane, ako je blok premali, neće biti dovoljno mesta da se sačuvaju svi pokazivači u slučaju velikih datoteka.

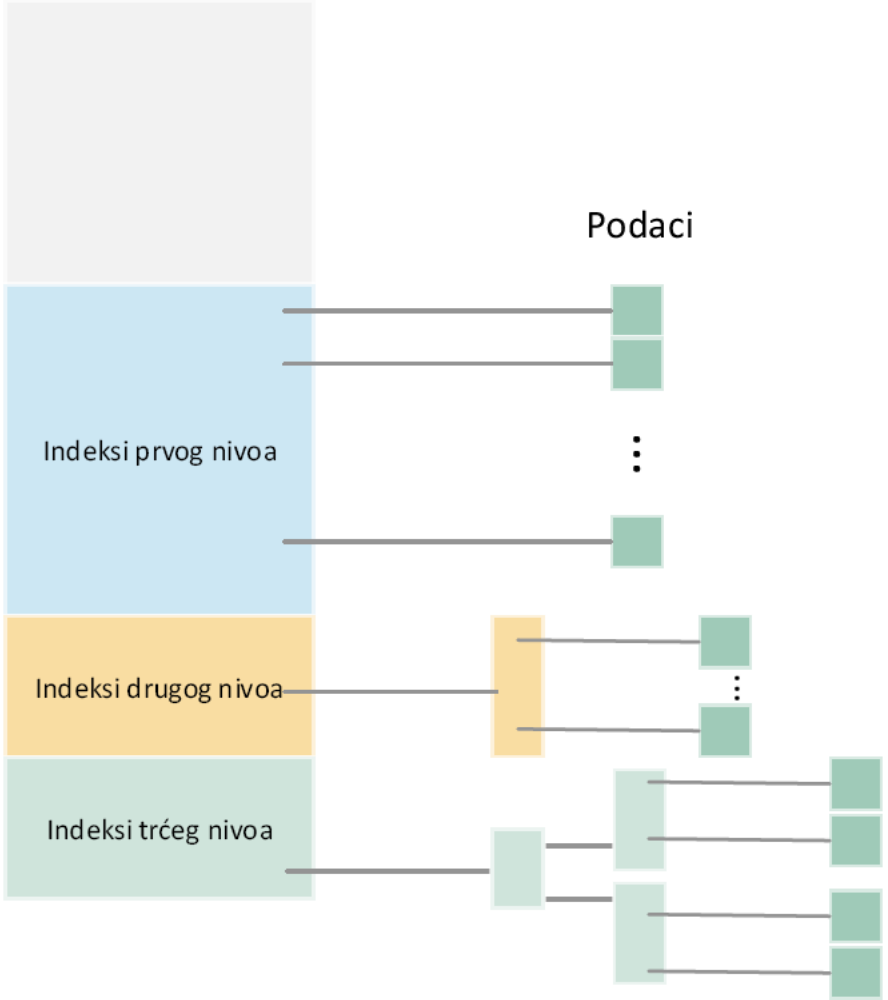
Indeksirana alokacija (4)

- Problem se može prevazići na sledeći način:
  1. Povezana lista blokova indeksa: blokovi indeksa se nižu u povezane liste, tako što poslednja adresa predstavlja pokazivač na sledeći blok indeksa. Pri tome, male datoteke i dalje imaju samo jedan blok indeksa.
  2. Indeksi sa više nivoa: kod dva nivoa, indeks prvog nivoa ne pokazuje direktno na blokove datoteke, već sadrži adrese blokova indeksa drugog nivoa, a oni sadrže adrese blokova. Da bi se pristupilo bloku, sistem u prvom nivou nalazi adresu bloka indeksa i tek u njemu nalazi adresu željenog bloka. Ovo se može proširiti na indekse višeg nivoa.

Indeksirana alokacija (5)

- Problem se može prevazići na sledeći način:
  3. Hibridni pristup: Koristi se povezana lista blokova indeksa. Određen broj blokova indeksa na početku su direktno indeksirani, što znači da adrese koje oni sadrže pokazuju na blokove fajla. Dalje u listi, određen broj blokova indeksa su indeksi sa dva nivoa, čije adrese pokazuju na blokove indeksa. Sledeći u listi su blokovi indeksa trećeg nivoa, itd.

Indeksirana alokacija (6)



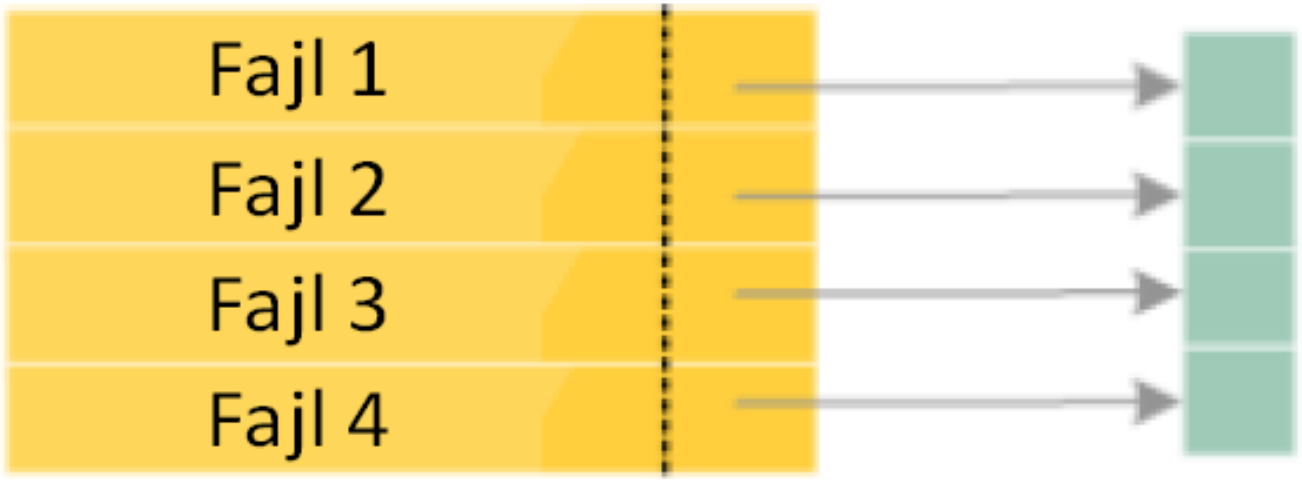
Hibridni pristup

Implementacija direktorijuma

- Da bi operativni sistem mogao da otvori datoteku koji se nalazi na određenoj putanji, u direktorijumu se mora naći elemenat koji opisuje datoteku i sadrži informacije o blokovima koji joj pripadaju.
  - Na primer, neka je data putanja „/user1/doc/text”. Operativni sistem prvo traži elemenat za direktorijum „user1“ u folderu „/“. Kada ga nađe, otvara taj direktorijum i u njemu traži informacije o direktorijumu „doc“. Najzad, u direktorijumu „doc“ se pronalaze informacije o datoteci „text“.
- Štaviše, operativni sistem mora u trenutnom folderu takođe naći elemenat za svaki naredni deo putanje koju je prosledio korisnik.

Implementacija preko liste

- Najjednostavnije je direktorijum predstaviti kao listu imena datoteka i pokazivača ka strukturama koje sadrže atribute i informacije o alociranim blokovima za te datoteke.



Struktura foldera zadata listom fajlova

Implementacija preko liste (2)

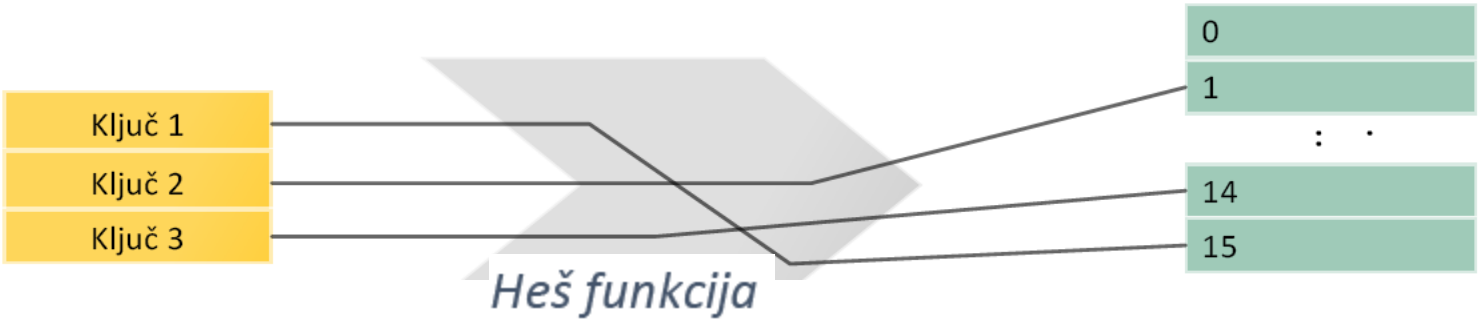
- Realizacija operacija nad direktorijumom:
  - Pri kreiranju nove datotke, pretražuje se direktorijum, tj. lista imena, se kako bi se utvrdilo da ne postoji fajl sa istim imenom.
    Nakon toga se unos za novi fajl dodaje na kraj liste.
  - Pri brisanju se prvo pronalazi datoteka sa datim imenom i onda se oslobađaju alocirani blokovi.
    Posle toga će elemenat u listi datoteka biti nevažeći pa je potrebno obeležiti ga kao takvog - za to se može koristiti jedan bit u strukturi koja čuva atribute (npr. 1 predstavlja da fajl postoji, 0 da je obrisan).
- Bolja ideja je da se koriste povezane liste za svaki elemenat u direktorijumu, čime se omogućava jednostavnije i efikasnije brisanje elemenata.

Implementacija preko liste (3)

- Mana ovog pristupa je to što pronalaženje datoteke nije efikasno.
- S obzirom na to da je pronalaženje datoteke sa zadatim imenom osnovna i najčešća operacija, ovo predstavlja ozbiljan problem.
  - Svaki put kada se datoteka kreira, briše ili otvara, direktorijum se mora pretražiti.
  - Ovaj problem se može rešiti održavanjem sortirane liste prema imenima datoteka. Tada se za pretraživanje koristi binarna pretraga, što je daleko efikasnije.
  - Međutim, održavanje sortirane liste komplikuje dodavanje novih elemenata u direktorijum.

Implementacija preko heš tabele

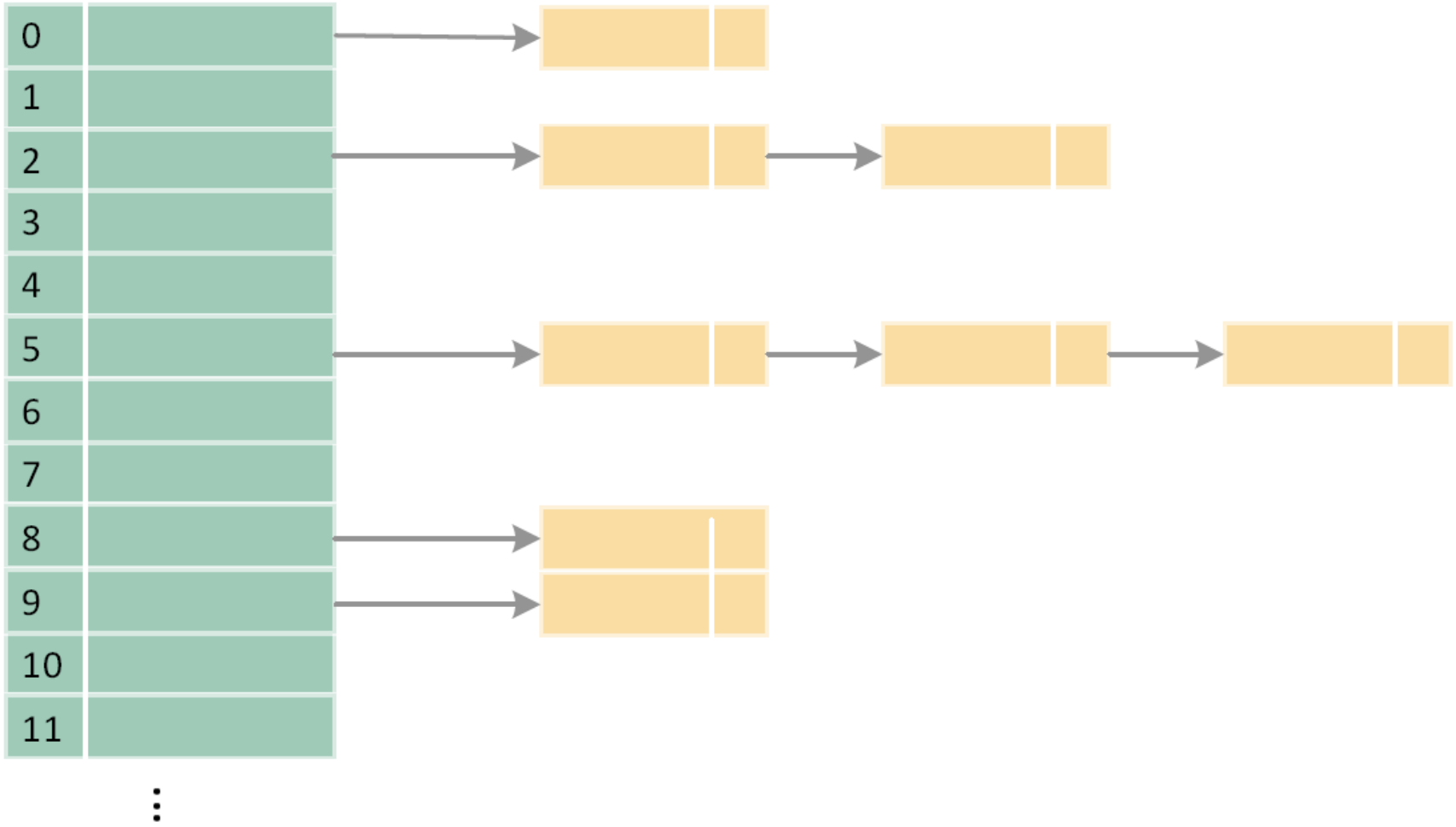
- Za folder se može koristiti i heš tabela.
- Pristup funkcioniše na sledeći način:
  - Tabela sadrži imena datoteka koja su poređana u indeksirani niz. Kada je zadato ime koje treba pronaći (tzv. ključ), to ime se dostavlja heš funkciji koja računa vrednost tj. redni broj iz tabele - indeks. Na osnovu indeksa, datoteka se nalazi direktno, bez pretraživanja, što je vrlo efikasno.



Implementacija preko heš tabele (2)

- Najjednostavnija heš funkcija se može implementirati kao ostatak pri deljenju nekim brojem (npr. 256).
  Od imena datoteke bi se mogao dobiti broj tako što bi se iskombinovale ASCII kodovi znakova imena.
  Heš funkcija bi taj broj podelila sa 256 i ostatak bi vratila kao indeks.
- Problem sa heš tabelama predstavlja situacija gde dva različita imena daju iste ostatke. Ovo se naziva kolizija.
- Problem je i fiksna dužina tabele. U prethodnom primeru, u tabeli bi imalo mesta za ne više od 256 imena fajlova, jer toliko ostataka postoji pri deljenju sa 256.
- Oba problema se mogu rešiti ulančavanjem. Neka svaki elemenat u heš tabeli predstavlja povezanu listu. Tada se novi elemenat nadovezuje na stare sa istim indeksom.

Implementacija preko heš tabele (3)



Heš funkcija - ulančavanje

Manipulisanje slobodnim prostorom

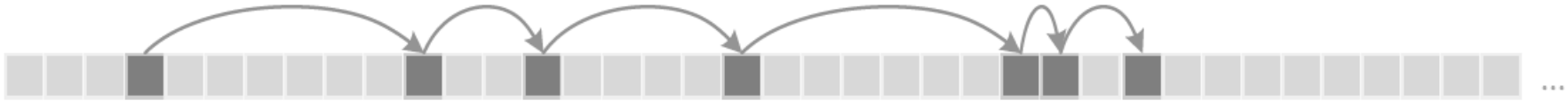
- Kada se datoteka obriše, potrebno je prostor osloboditi i upotrebiti ga za upis novih podataka.
- Da bi to bilo moguće, potrebno je da operativni sistem vodi evidenciju o slobodnom prostoru, tačnije da ima „spisak“ slobodnih blokova.
  - Kada se kreira datoteka, iz spiska se uzima potreban broj blokova koji se alociraju, a njihove adrese se brišu sa spiska.
  - Kada se datoteka briše, adrese blokova koje je zauzimla ta datoteka se dopisuju na spisak.

Manipulisanje slobodnim prostorom (2)

- Ponegde se spisak slobodnih blokova implementira kao bit mapa.
- Svaki blok se predstavlja nulom ili jedinicom, u zavisnosti od toga da li je zauzet ili slobodan.
  - Na primer, ako su blokovi 3, 5, 7, 11, 12, 13, 14, 18, 20, 21 i 22 slobodni, bit mapa bi imala sledeći izgled: 00101010001111000101110000000000 ...
- Ovo je vrlo jednostavno i efikasno rešenje.
- Međutim, ako je disk veliki, problem je u tome što onda i bit mapa takođe mora biti velika.

Manipulisanje slobodnim prostorom (3)

- Drugo rešenje se zasniva na ideji da se koristi povezana lista u kojoj bi se nizali svi slobodni blokovi.



Povezana lista slobodnih blokova

- Rešenje se dalje može unaprediti jer se, uglavnom, više susednih blokova alocira ili oslobađa odjednom.
  Imajući to na umu, u svakom čvoru liste može se čuvati adresa prvog slobodnog bloka i broj koji označava koliko ima slobodnih blokova od njega.

Sigurnost

- Pad sistema može dovesti do nekonzistentnosti podataka.
  - Na primer, kreiranje nove datoteke je operacija koja zahteva više promena nad sistemom datoteka, kao što su alociranje novih blokova, ažuriranje strukture koja čuva informacije o njima, menjanje strukture foldera i sl. Ove promene mogu biti prekinute padom sistema, rezultujući nekonzistentnost.
- Pored pada sistema, iste probleme može izazvati i greška u implementaciji, kontroleru, pa čak i aplikaciji.

Sigurnost (2)

- Da bi otkrio grešku, datotečki sistem mora skenirati i proveriti sve metapodatke. Neke greške je moguće popraviti.
  - Na primer, ako sistem koristi povezanu listu blokova za alokaciju datoteke, onda se cela datoteka može povratiti čitanjem blokova, jer je u svakom upisan pokazivač na sledeći. U dom slučaju se oštećena struktura direktorijuma može vratiti.
- Međutim, gubitak direktorijuma kod indeksirane alokacije može biti nepovratan, jer blokovi nemaju nikakve informacije jedni o drugima.

Sigurnost (3)

- Da bi se obezbedila konzistentnost, često su u upotrebi sistemi datoteka koji koriste dnevnike (journaling file system).
- Ideja je da se svi metapodaci prvo upišu u dnevnik.
- Skup operacija koje su potrebne da bi se neki zadatak obavio, npr. kreiranje datoteke, naziva se transakcija.
- Tek kada su planirane operacije upisane u dnevnik, izvršavaju se stvarne promene nad podacima.

Sigurnost (4)

- Kada se jedna operacija završi, pokazivač u dnevniku se uveća tako da pokazuje na sledeću i tako sve dok se ne izvrši kompletan skup, nakon čega se on uklanja iz dnevnika.
- Ako se dogodi pad sistema, dnevnik će sadržati sve planirane nezavršene operacije koje treba izvršiti nakon ponovnog podizanja sistema.

Zahvalnica

Najveći deo materijala iz ove prezentacije je preuzet iz knjige „Operativni sistemi“ autora prof. dr Miroslava Marića i iz slajdova sa predavanja koje je držao prof. dr Marić.

Hvala prof. dr Mariću na datoj saglasnosti za korišćenje tih materijala.