

Glava 3.

Uloga operativnog sistema u upravljanju podacima

Upravljanje razmenom podataka između aplikativnog programa i datoteke na disku je zadatak operativnog sistema. Na taj način, programeri se oslobadaju potrebe da vode računa o svim kompleksnim aktivnostima koje se moraju realizovati prilikom te razmene podataka. Delegiranjem brige o upravljanju stvarnom razmenom podataka između aplikativnog programa i datoteke na disku se obezbeđuje i određeni nivo zaštite podataka od neovlašćenog korišćenja i zaštita programa od slučajnog, ili namernog oštećenja. Ako bi se dozvolilo programeru da upravlja podacima, postojali bi uslovi da on pristupi svakoj datoteci pa i tduo, ili pak da učita podatke u proizvoljan deo operativne memorije i tako uništi neki od programa.

S obzirom na nivo usluga pri upravljanju razmenom podatka, operativni sistemi se mogu klasifikovati na:

- operativne sisteme koji pružaju samo usluge niskog nivoa i
- operativne sisteme koji pružaju i usluge visokog nivoa.

Operativni sistemi koji pružaju usluge visokog nivoa, pružaju i usluge niskog nivoa. Usluge niskog nivoa:

- pokrivaju širok spektar aktivnosti obezbeđenja memorijskog prostora za datoteku na disku i upravljanja stvarnom razmenom podataka između operativne memorije i diska,
- ali ne vode računa o:
 - logičkoj strukturi podataka datoteke,
 - izgradnji specijalnih pomoćnih struktura podataka za poboljšanje efikasnosti korišćenja datoteke i
 - takvom traženju u datoteci, koje je zasnovano na vrednosti podataka.

Usluge visokog nivoa pokrivaju sve napred pobrojane aktivnosti. U operativne sisteme koji pružaju usluge niskog nivoa spadaju savremeni operativni sistemi, čiji izraziti predstavnik je Unix. U operativne sisteme koji pružaju usluge visokog nivoa spadaju operativni sistemi centralnih (mainframe) računara.

Operativni sistem nije jedan jedinstven program. To je skup programa i mehanizama, namenjenih za upravljanje radom hardverskim i softverskim resursima računarskog sistema. Deo operativnog sistema koji upravlja podacima naziva se ulazno-izlaznim sistemom ili sistemom za upravljanje datotekama. Ovaj sistem sačinjava veći broj programa, odnosno rutina i mehanizma. U ovoj glavi će te rutine i mehanizmi biti grupisani na sledeći način:

- 1º Rutine i mehanizmi *sistemskih poziva*, putem kojih operativni sistem pruža usluge korisničkim programima za izvršavanje osnovnih operacija sa datotekama.
- 2º Rutine za upravljanje *katalogom*. Koriste se za formiranje kataloga sa imenima datoteka. Ovi katalozi omogućuju kasnije lako i brzo pronalaženje datoteka na eksternim memorijskim uređajima na osnovu imena.
- 3º Rutine i mehanizmi za uspostavljanje veze sa datotekom na disku pri izvršavanju sistemskih poziva. U te mehanizme spada određeni broj tabela sa podacima o datotekama.

U glavi 5 će biti obradene:

- 4º Rutine i mehanizmi za upravljanje *prostorom* na eksternim memorijskim uređajima sa direktnim pristupom, koje se koriste za dodelu memoriskog prostora datotekama, kao i za vođenje brige o slobodnom prostoru.
- 5º Rutine i mehanizmi za pripremu i upravljanje stvarnom, fizičkom razmenom podataka između operativne memorije i diska.

U glavi 6 su obradene:

- 6º Metode pristupa. Metode pristupa predstavljaju programe i mehanizme operativnih sistema visokog nivoa, za implementaciju usluga upravljanja i korišćenja datoteka sa kompleksnom organizacijom:

Sistemski pozivi niskog nivoa su, u ovoj glavi, ilustrovani korišćenjem ulazno-izlaznih Posix naredbi niskog nivoa iz programa pisanog u programskom jeziku C. Izlaganje programskih procedura je, u velikoj meri, ograničeno samo na sistemske pozive. Ovaj tekst se ne bavi pitanjima programiranja u programskom jeziku C. Pretpostavlja se da čitalac poseduje odgovarajuće znanje, koje mu omogućava praćenje teksta.

3.1 SISTEMSKI POZIVI

Za izvršavanje osnovnih operacija sa datotekama, operativni sistem pruža usluge korisničkim programima putem takozvanih sistemskih poziva. Sistemski poziv ili makroinstrukcija dovodi do prenosa upravljanja nad radom centralnog procesora sa korisničkog programa na operativni sistem. Pozivom program traži neku uslugu od operativnog sistema. Za izvršavanje sistemskih poziva, operativnom sistemu treba čitav niz podataka o samoj datoteci. Informacije o svakoj datoteci se čuvaju u posebnim datotekama na disku, koje se nazivaju katalog (adresar, direktorijum) i sistemska tabela datoteke.* Ove datoteke su detaljno opisane u narednim delovima ove glave. Za izvršavanje sistemskih poziva, operativnom sistemu su potrebni sledeći podaci o datoteci:

- jedinstveni naziv fizičke datoteke (jedina informacija u ljudski razumljivom obliku),

* engleski: redom *directory* i *index node*

- tip datoteke (na primer .doc, .txt, .html, .c, .cpp, .java i slično),
- vlasnik datoteke (po pravilu, korisnik, koji je kreirao datoteku),
- adresa datoteke (oznaka memorijskog uredaja i pokazivač ka sistemskoj tabeli datoteke na tom uredaju),
- lista adresa ka fragmentima memorijskog prostora dodeljenog datoteci,
- veličina datoteke u jedinicama memorijskog kapaciteta i
- podaci o zaštiti (podaci o imaćima dozvola za čitanje, pisanje i izvršavanje datoteke).

Datoteka se naziva fizičkom, jer ona stvarno postoji na disku i naziv te datoteke je poznat korisniku. Ta ista fizička datoteka dobija i druga, logička imena, kao što će to biti pokazano u narednom tekstu.

U osnovne sistemske pozive spadaju zahtevi:

- `create` (kreiranje datoteke),
- `write` (pisanje novih podataka u datoteku),
- `read` (čitanje podataka iz datoteke),
- `seek` (pozicioniranje na početak određene lokacije u datoteci),
- `delete` (brisanje kompletног sadržaja datoteke, njenih podataka u katalogu i njene sistemske tabele) i
- `truncate` (brisanje kompletног sadržaja datoteke, ali ne i njene sistemske tabele, niti njenih podataka u katalogu).
- `close` (zatvaranje datoteke).

Da bi izvršio upis novih podataka ili čitanje postojećih iz datoteke, operativnom sistemu je potrebna informacija o adresi lokacije unutar memorijskog prostora dodeljenog datoteci, u koju treba da piše, ili iz koje treba da čita. Neki operativni sistemi zato zahtevaju da sistemski pozivi sadrže adresu te lokacije. Međutim, mnogi operativni sistemi tu informaciju izvode iz vrednosti adrese lokacije u koju je poslednji put pisano, ili iz koje je poslednji put čitano. Ta adresa se nalazi u promenljivoj, koja se naziva *tekućim pokazivačem*, ili indikatorom aktualnosti, jer ukazuje na tekući, aktuelni sadržaj datoteke. Vrednost promenljive *tekući_pokazivač* svake aktivne datoteke čuva se u operativnoj memoriji.

Da bi izvršio poziv za kreiranje datoteke, operativni sistem:

- nalazi memorijski prostor na disku za novu datoteku,
- upisuje njeni ime u katalog i
- formira sistemsku tabelu datoteke.

Zahtev za upis podataka u datoteku dovodi do:

- traženja naziva datoteke u katalogu i učitavanje podataka iz sistemske tabele datoteke,
- upis podataka počev od tekućeg pokazivača i
- ažuriranje tekućeg pokazivača na novu vrednost.

Da bi izvršio čitanje, operativni sistem:

- vrši traženja naziva datoteke u katalogu i učitavanje podataka iz sistemske tabele datoteke,
- čita podatke počev od tekućeg pokazivača i
- ažurira tekući pokazivač na novu vrednost.

Pozicioniranje se vrši postavljanjem sadržaja promenljive `tekuci_pokazivač` na novu vrednost. Brisanje kompletne datoteke zahteva:

- traženje naziva datoteke u katalogu i učitavanje podataka iz sistemske tabele datoteke,
- oslobođanje memoriskog prostora dodeljenog datoteci i
- brisanje naziva datoteke iz adresara i brisanje sistemske tabele datoteke.

Brisanje samo sadržaja datoteke inicira:

- traženje naziva datoteke u katalogu i učitavanje podataka iz sistemske tabele datoteke i
- oslobođanje memoriskog prostora dodeljenog datoteci.

U daljem tekstu su opisane aktivnosti sistemskih poziva, ilustrovane efektima njihovog pozivanja iz programa pisanog u programskom jeziku C, putem takozvanih Posix ulazno-izlaznih naredbi, u okruženju operativnog sistema Unix. Posix naredbe spadaju u naredbe niskog nivoa, bliskog funkcijama operativnog sistema. Posix ulazno-izlazne naredbe su odabране, jer omogućavaju uočavanje aktivnosti, koje se dešavaju „iza scene“. Te aktivnosti objašnjavaju šta se stvarno dešava pri formirajući i korišćenju datoteka.

3.1.1 OTVARANJE DATOTEKE

Većina osnovnih operacija sa datotekama zahteva prethodno traženje po katalogu na osnovu imena datoteke i prenos sistemske tabele datoteke u operativnu memoriju. Da bi se izbeglo repetitivno izvršavanje ovih operacija, uveden je sistemski poziv `open()` (otvoriti), koji se mora izvršiti na početku korišćenja svake datoteke. Na osnovu tog poziva, operativni sistem pronalazi podatke o datoteci na disku i prenosi ih u niz tabela u operativnoj memoriji, kako bi ih mogao koristiti pri svakom narednom pozivu. Ovi podaci o datoteci ostaju u operativnoj memoriji tokom celog intervala vremena, u kojem je datoteka aktivna. O tabelama operativnog sistema koje sadrže podatke o datoteci, biće više reči u narednim delovima ove glave. Sistemski poziv `open` ima sledeće zadatke:

- da na osnovu naziva fizičke datoteke, putem kataloga, pronade sistemsu tabelu datoteke i prenese je u operativnu memoriju, ukoliko datoteka već postoji,
- ako datoteka još ne postoji, da prvo formira zapis o datoteci u katalogu, formira njenu sistemsu tabelu datoteke i rezerviše inicijalni memoriski prostor za nju na disku (što odgovara aktivnosti *kreiranja* datoteke), a zatim nastavi sa aktivnostima otvaranja kao za datoteku koja postoji,
- da dodeli otvorenoj datoteci takozvano logičko ime, uspostavi vezu između tog logičkog i fizičkog imena datoteke i vrati logičko ime datoteke programu, koji je zatražio otvaranje datoteke,
- da saglasno specifikaciji, otvor datoteku za aktivnosti, kao što su čitanje, pisanje, ili dodavanje novih podataka na kraj datoteke i
- da postavi vrednost promenljive `tekuci_pokazivač` na određenu vrednost.

Posix naredba za otvaranje datoteke ima sledeći opšti oblik

```
int fd = open(char * ime_dat, int akt [int ovl]);
```

Parametri naredbe `open` imaju sledeće značenje:

- `fd` je logičko ime, ili identifikator datoteke, koji operativni sistem vraća programu. Ako pri otvaranju dođe do greške, `fd` ima vrednost manju od nule.
- `ime_dat` je naziv fizičke datoteke,
- `akt` ukazuje na zadatke, odnosno aktivnosti, koje poziv `open` treba da izvrši. Putem parametra `akt` se specificira da li `open` treba da otvori neku postojeću, ili kreira novu datoteku sa imenom `ime_dat`. Takođe, u okviru parametra `akt` se specificira koje operacije (čitanje, upis) su dozvoljene na otvorenoj datoteci. Parametar `akt` može imati neku kombinaciju sledećih vrednosti:
 - `O_APPEND` – otvara datoteku za takvo pisanje da se novi podaci upisuju na kraj datoteke.
 - `O_CREAT` – kreira novu datoteku za upis. Nema efekta, ako datoteka već postoji.
 - `O_EXCL` – vraća grešku, ako je specificiran i `O_CREAT`, a datoteka već postoji.
 - `O_RDONLY` – otvara datoteku samo za čitanje.
 - `O_RDWR` – otvara datoteku za čitanje i pisanje.
 - `O_TRUNC` – briše sadržaj datoteke, ako datoteka postoji.
 - `O_WRONLY` – otvara datoteku samo za pisanje.

Pri prevođenju programa, kompjajler pretvara svaku od nabrojanih vrednosti parametra `akt` u ceo broj, a zatim primenjuje logičku funkciju \vee (ili) na njihove binarne vrednosti, da bi izračunao rezultatnu vrednost parametra `akt`. Ako `akt` sadrži `O_APPEND`, tada `open` postavlja vrednost promenljive *tekuci pokazivac* na adresu kraja datoteke, inače je postavlja na adresu početka datoteke.

- `ovl` je specifikacija zaštite datoteke. Ovaj parametar je obavezан, ako parametar `akt` sadrži vrednost `O_CREAT`, inače je opcion. Parametar `ovl` definiše ovlašćenja korisnika za korišćenje datoteke, odnosno ukazuje na prava:
 - čitanja,
 - pisanja i
 - izvršavanja

datoteke. Operativni sistem Unix klasificiše korisnike datoteke u tri kategorije. To su: vlasnik datoteke, grupa i ostali. Svaka grupa može imati dodeljenu bilo koju kombinaciju od navedena tri prava korišćenja datoteke. Parametar `ovl` je trocifren oktalni broj. Prva cifra tog broja ukazuje na prava vlasnika, druga na prava grupe, a treća na prava ostalih korisnika datoteke. Pri tome, binarna cifra najveće težine svake oktalne cifre se odnosi na čitanje, naredna na pisanje, a binarna cifra najmanje težine na izvršavanje datoteke. Ako je binarna cifra postavljena na jedan, odgovarajuća akrivnost je dozvoljena, inače nije. Slika 3.1 ilustruje zapis o ovlašćenjima za korišćenja datoteke. Na slici, slovo `r` označava čitanje, slovo `w` pisanje, a slovo `e` izvršavanje datoteke. Mala slova $x \in \{0, 1\}$ označavaju binarne vrednosti oktalnih cifara.

	<code>r w e</code>	<code>r w e</code>	<code>r w e</code>
<code>ovl =</code>	<code>x x x</code>	<code>x x x</code>	<code>x x x</code>
	vlasnik	grupa	ostali

Slika 3.1.

Primer 3.1. Izvršenje sledeće Posix naredbe kreira datoteku pod nazivom student.data, otvara je za upisivanje novih slogova na kraj datoteke i za čitanje.

```
fpt = open(student.data, O_CREAT | O_APPEND | O_RDWR, 644);
```

Nakon uspešnog otvaranja, operativni sistem vraća neki broj, kao što je 3 ili 4, kao vrednost promenljive fd i postavlja tekući pokazivač na adresu kraja datoteke. Prava korišćenja datoteke su sledeća: vlasnik ima pravo čitanja i pisanja, a grupa i ostali samo pravo čitanja datoteke. Slika 3.2 prikazuje sadržaj promenljive ovl datoteke student.data.

Izvršenje sledeće Posix naredbe otvara datoteku pod nazivom student.data samo za čitanje.

```
fpt = open(student.data, O_RDONLY);
```

	r w e	r w e	r w e
ovl	= 1 1 0	1 0 0	1 0 0
	vlasnik	grupa	ostali

Slika 3.2.

Neki programski jezici ne sadrže open instrukciju. Kod programa pisanih u ovim jezicima, sve aktivnosti otvaranja datoteke izvršavaju se prilikom prvog zahteva za čitanje ili upis podataka.

3.1.2 PISANJE I ČITANJE IZ DATOTEKE

Pisanje i čitanje predstavljaju stvarne ulazno-izlazne operacije sa datotekama. Obe operacije se izvršavaju tako što se piše u lokaciju ili čita iz lokacije čija adresa se nalazi u promenljivoj *tekući_pokazivač*. Nakon završene operacije, promenljiva *tekući_pokazivač* sadrži adresu lokacije, koja je fizički naredna u odnosu na lokaciju, u koju je baš pisano, ili iz koje je baš izvršeno čitanje.

PISANJE

Poziv funkcije za pisanje zahteva specificiranje tri parametra. To su:

- odredišna datoteka,
- izvor podataka u operativnoj memoriji i
- broj bajtova, koje treba upisati u datoteku.

U Posix jeziku, naredba za upis podataka u datoteku tačno prati taj zahtev

```
s_size write(int fd, &izvor, int broj),
```

gde je *s_size* specijalni numerički tip podataka, *fd* logičko ime odredišne datoteke, *&izvor* pokazivač na početak promenljive u programu koja sadrži podatke za pisanje u datoteku, a *broj* promenljiva koja sadrži broj bajtova za upis u datoteku. U slučaju uspešnog upisa, funkcija *write()* vraća broj upisanih bajtova, a ako je upis neuspešan, povratna vrednost je -1.

Primer 3.2. Posmatra se datoteka student.data otvorena u primeru 3.1. Naredni fragment koda deklariše potrebne promenljive i upisuje podatke o studentu, smeštene u promenljivu student tipa niz karaktera, u datoteku student.data na disku.

```
int broj;
char student[];
...//ovde je niz student postavljen na neku vrednost
broj = strlen(student);
(3.1) write(fd, &student, broj);
```

Funkcija strlen() vraća aktuelni broj karaktera u nizu student. Pošto je datoteka otvorena za upis na kraj datoteke, novi podaci će biti upisani na kraj datoteke, bez obzira na prethodnu vrednost promenljive *tekuci_pokazivač*. Nakon uspešnog izvršenja naredbe (3.1), promenljiva *tekuci_pokazivač* će sadržati adresu kraja datoteke. □

ČITANJE

Poziv funkcije za čitanje zahteva specificiranje sledeća tri parametra:

- izvorna datoteka,
- odredišna promenljiva u operativnoj memoriji i
- broj bajtova, koje treba pročitati.

U Posix jeziku, naredba za čitanje podataka u datoteku tačno prati taj zahtev

```
s_size read(int fd, &odred, int broj),
```

gde je s_size specijalni numerički tip podataka, fd logičko ime izvorne datoteke, &odred pokazivač na početak promenljive u programu koja treba da primi podatke iz datoteke, a broj promenljiva, koja sadrži broj bajtova, koje treba pročitati iz datoteke. Sadržaj promenljive broj je veoma važan, jer predstavlja jedinu indikaciju sistemskom pozivu koliko podataka treba da pročita iz datoteke. U slučaju uspešnog čitanja, funkcija read() vraća broj pročitanih bajtova, a ako je čitanje neuspešno, povratna vrednost je -1.

Primer 3.3. Prepostavlja se da je *tekuci_pokazivač* datoteke student.data postavljen na početak podataka upisanih putem naredbe (3.1), da promenljiva broj sadrži vrednost, na koju je postavljena u primeru 3.2 i da važi student[0] == '\0'. Nakon uspešnog izvršenja naredbe

```
(3.2) read(fd, &student, broj);
```

promenljiva student[] će ponovo sadržati podatke koji su naredbom (3.1) upisani u datoteku. □

3.1.3. POZICIONIRANJE

Često se pokazuje pogodnom mogućnost upravljanja postavljanjem sadržaja promenljive *tekuci_pokazivač*. U te slučajeve spada:

- direktno pristupanje lokaciji sa poznatom adresom, umesto sekvencijalnog kretanja kroz datoteku,
- povratak na početak datoteke,
- skok na kraj datoteke da bi se tamo upisali novi podaci.

Da bi ove potrebe bile zadovoljene, mora se obezbediti mogućnost upravljanja izmenama sadržaja promenljive *tekući_pokazivač*. Aktivnost postavljanja sadržaja *tekući_pokazivač* na neku vrednost naziva se *pozicioniranjem*. Pozicioniranje zahteva tri podatka. To su:

- naziv datoteke,
- referentna tačka i
- pomak u odnosu na referentnu tačku.

Referentna tačka može biti:

- početak datoteke,
- tekuća pozicija i
- kraj datoteke.

Poziv funkcije za pozicioniranje ima, u Posix jeziku, sledeći opšti oblik

```
off_t lseek(int fd, long bajt_pomak, int ref),
```

gde je *off_t* numerički tip podataka, *lseek* naziv funkcije, a *bajt_pomak* broj bajtova za koliko treba povećati ili smanjiti sadržaj promenljive *tekući_pokazivač* u odnosu na promenljivu *ref*. Promenljiva *ref* može imati jednu od sledeće tri vrednosti:

- 0 za početak datoteke, čija simbolička vrednost je *SEEK_SET*,
- 1 za tekuću poziciju, čija simbolička vrednost je *SEEK_CUR* i
- 2 za kraj datoteke, čija simbolička vrednost je *SEEK_END*.

Preslikavanje simboličkih u numeričke vrednosti vrši kompjuter. Ako je pozicioniranje uspešno izvršeno, funkcija vraća sadržaj promenljive *tekući_pokazivač*.

Primer 3.4. Nakon upisa podatka u datoteku *student.data* putem naredbe (3.1), sledeća Posix naredba postavlja *tekući_pokazivač* na početak tih podataka, kako bi ih naredba (3.2) mogla pročitati

```
lseek(fd, -broj, SEEK_CUR);
```

gde je *broj* je promenljiva tipa *long*, čija je vrednost broj bajtova koje je funkcija (3.1) upisala u datoteku. □

3.1.4. ZATVARANJE DATOTEKE

Sistemski poziv za zatvaranje datoteke oslobađa logičku oznaku datoteke za korišćenje sa nekom drugom datotekom i, ako je datoteka bila otvorena za pisanje, tada:

- inicira prenos na disk preostalih podataka iz dela operativne memorije, u koji operativni sistem privremeno smešta podatke, pre upisa u datoteku na disku i.
- prenosi tekući sadržaj sistemske tabele datoteke iz operativne memorije na disk.

O prenosu preostalih podataka na disk će biti više reči u glavi 5, u okviru diskusije o baferima.

Operativni sistem, često, automatski zatvara datoteku nakon normalnog završetka programa, koji je datoteku otvorio. Međutim, to nije pravilo koje nema izuzetaka. Osim toga, ako se program nije završio normalno, može doći do gubitka podataka. Zbog toga se zatvaranje datoteke smatra dobrom programerskom praksom.

Primer 3.5. Poziv funkcije za zatvaranje datoteke ima, u Posix jeziku, sledeći opšti oblik

```
int close(int fd);
```

3.1.5. Unix KOMANDE ZA RAD SA DATOTEKAMA

Operativni sistem Unix podržava mnoge komande za rad sa datotekama. Te komande mogu biti inicirane interaktivno sa komandnog prompta, ili iz takozvanih skript (komandnih) datoteka operativnog sistema. U narednom tekstu biće prikazane samo one, koje su od značaja za tekst ovog poglavlja. Većina tih komandi ima mnogo opcija, čija upotreba izlazi izvan okvira ovog teksta. Na slici 3.2 su date odabrane Unix komande sa kratkim opisom. Za detalje treba konsultovati neki Unix priručnik.

less <code>ime_datoteke</code>	prikazuje sadržaj tekstualne datoteke <code>ime_datoteke</code> na ekranu
grep <niz_kar> <code>ime_dat</code>	prikazuje one slogove datoteke, koji sadrže <niz_kar>
cp <code>ime_dat1</code> <code>ime_dat2</code>	kopira sadržaj datoteke <code>ime_dat1</code> u datoteku <code>ime_dat2</code>
mv <code>ime_dat1</code> <code>ime_dat2</code>	premešta datoteku <code>ime_dat1</code> u datoteku <code>ime_dat2</code>
rm <code>ime_datoteke</code>	briše datoteku <code>ime_datoteke</code>
chmod ovl <code>ime_dat</code>	menja definiciju ovlašćenja za datoteku <code>ime_dat</code>

Slika 3.2.

Većina komandi navedenih na slici 3.2 ima očigledno značenje. O komandi `grep` će biti više reči u glavi posvećenoj sekvenčijalnim datotekama. Primena niza drugih komandi je ilustrovana u narednim delovima ove glave. Komanda `less` se intenzivno koristi u glavi 4. Na ovom mestu će biti komentarisana samo naredba `chmod`. Komanda `chmod` služi za promenu ovlašćenja za korišćenje datoteke.

Primer 3.6. Neka je operativni sistem postavio vrednost 744 u promenljivu `ovl` nakon formiranja datoteke `student.data`. Sadržaj promenljive `ovl` daje vlasniku pravo čitanja, pisanja i izvršavanja, a grupi i ostalima pravo čitanja. Naredba

```
chmod 650 student.data
```

oduzima vlasniku pravo izvršavanja, grupi dodaje pravo izvršavanja, a ostalima ukida pravo čitanja datoteke. Nakon toga, promenljiva `ovl` će imati sadržaj prema slici 3.3. □

	r w e	r w e	r w e
ovl	= 1 1 0	1 0 1	0 0 0
	vlasnik	grupa	ostali

Slika 3.3.

3.1.6 O POGLEDU NA DATOTEKE OPERATIVNOG SISTEMA Unix

Operativni sistem Unix podržava apstraktни pogled na datoteke, kao niz bajtova. Tako pristup se odlikuje velikom jednostavnosću i fleksibilnošću, ali prebacuje na korisnički aplikativni program svu brigu oko semantike sadržaja datoteke i zahteva uvođenje posebnih mehanizama za prepoznavanje granica podataka sa samostalnim značenjem.

Kao posledica opisane apstrakcije izgleda datoteke, sistemski pozivi zahtevaju od programa:

- specificiranje broja bajtova koje treba upisati u datoteku, a ne specificiranje neke semantički definisane jedinice podatka,
- specificiranje broja bajtova koje treba pročitati iz datoteke, jer operativni sistem nema mogućnost da sam odredi takve granice podataka u datoteci, koje bi imalo smisla pročitati,
- specificiranje rednog broja bajta od početka datoteke na koji treba postaviti vrednost *tekuceg pokazivača*, jer ne postoji kriterijum, na osnovu kojeg bi sam operativni sistem mogao da zaključi gde u datoteci počinje neka smisaozaokružena jedinica podataka.

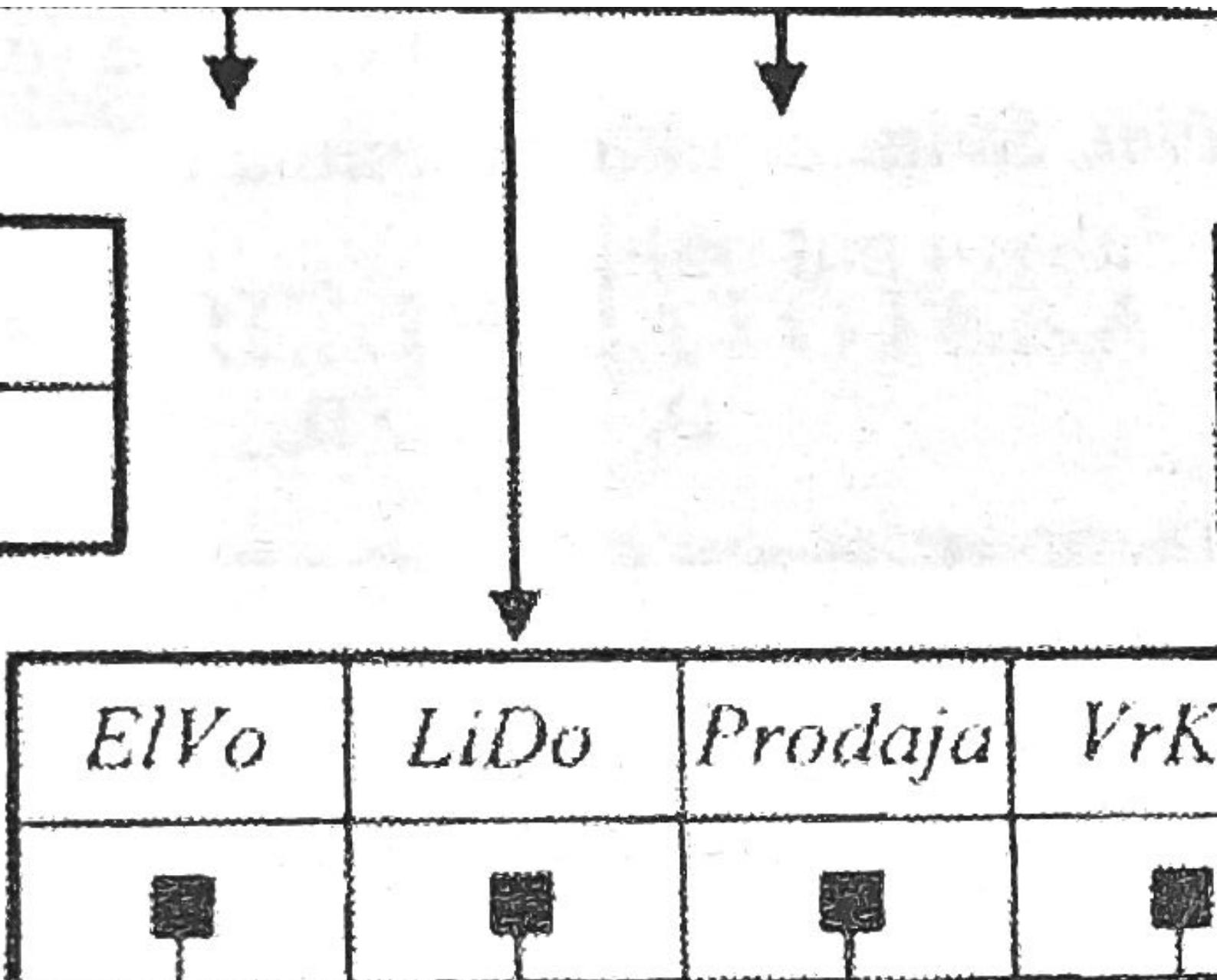
Može se zaključiti da Unix, kao operativni sistem, svojom koncepcijском jednostavnosću i fleksibilnošću pогледа na datoteke, prenosi odgovornost za rešavanje određenog broja kompleksnih zadataka na aplikativni program. Ovom pitanju je posvećena glava 4 ove knjige.

U nekim drugim okruženjima, kapacitet lokacije se, u programu, eksplicitno deklariše, tako da programski jezik i operativni sistem, između ostalog, automatski preračunavaju pomak, izražen brojem lokacija, u pomak izražen brojem bajtova.

3.2 KATALOG

Katalog predstavlja spisak datoteka. Naziva se i direktorijumom. Katalog omogućava korisniku da dodeli naziv datoteci pri njenom formiraju i da je poziva specificirajući samo taj naziv kad je ponovo želi koristiti. Pojam naziva datoteke, na ovom mestu, odnosi se na njeno fizičko ime. To je niz reči odvojenih separatorima. Kao što je to opisano u glavi 3, operativni sistem formira datoteku putem sistemskog poziva open, na osnovu zahteva korisničkog programa. To je i trenutak kada operativni sistem upisuje ime datoteke u katalog.

Potreba da se nazivi datoteka upisuju, pronalaze i menjaju u proizvoljnem redosledu, zahteva da katalog, koji i sam predstavlja datoteku, bude tako organizovan da dozvoli brz pristup slučajno odabranom zapisu - nazivu datoteke. Da bi se to postiglo, katalog se organizuje kao struktura tipa stabla. Koren te strukture naziva se glavnim katalogom ili glavnim direktorijumom, a čvorovi na nižim nivoima hijerarhije nazivaju se potkatalozima ili poddirektorijumima. Glavni katalog, kao čvor stabla, sadrži niz imena svojih potkataloga i datoteka. Svakom imenu je pridružen pokazivač. Ako je podređeni objekat potkatalog, pokazivač predstavlja nje-



STRUKTURA DIREKTORIJUMA OPERATIVNOG SISTEMA Unix

U operativnom sistemu Unix, katalog se naziva *sistemom datoteka*. Sistem datoteka je struktura stabla nad skupom direktorijuma i datoteka. Direktorijum u korenu stabla se označava kosom crtom /. Svi direktorijumi, uključujući koren, mogu sadržati dve vrste podređenih objekata. To su: datoteke i direktorijumi. Pošto Unix posmatra uređaje, kao što su tastatura, konzolni ekran i štampač kao datoteke, direktorijumi mogu sadržati i pokazivače ka tim uređajima. Imena datoteka na disku odgovaraju njihovim fizičkim imenima.

Svakom direktorijumu ili datoteci D_i u sistemu datoteka operativnog sistema Unix odgovara jedinstven apsolutni put od korena stabla, oblika

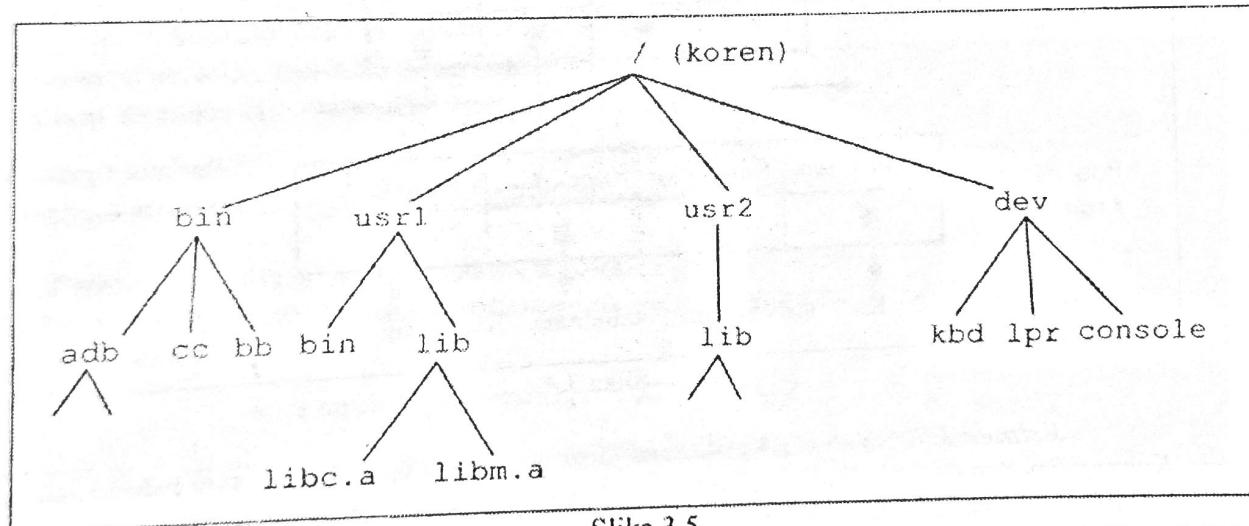
$$/D_1 / \dots / D_i / \dots / D_n,$$

gde je D_i ($1 \leq i \leq n - 1$) naziv poddirektorijuma, a D_n ili naziv poddirektorijuma, ili naziv datoteke. Apsolutni put uvek počinje kosom crtom (/), koja predstavlja naziv korena, a završava se bilo nazivom direktorijuma, ili datoteke. Znak kosa crta predstavlja i naziv korena i separator u putu.

Izdavanje komandi operativnom sistemu Unix se uvek vrši sa nekog direktorijuma. Taj direktorijum se naziva *tekućim*. Svim direktorijumima i datotekama, koji su podređeni tekućem direktorijumu D_t , odgovara relativni put oblika

$$D_{t+1} / \dots / D_m,$$

gde je D_{t+1} naziv direktorijuma, direktno podređenog D_t , a D_m naziv ili direktorijuma ili datoteke.



Slika 3.5.

Primer 3.8. Na slici 3.5 je prikazana struktura jednog direktorijuma operativnog sistema Unix. Apsolutni put datoteke libm.a je

$$/usr2/lib/libm.a$$

Apsolutni put datoteke console (konzolni ekran) je

$$/dev/console$$

Neka je `usr1` tekući direktorijum. Tada je

`lib/libm.a`

relativni put do datoteke `libm.a`. □

Naredni primer ilustruje postupak definisanja direktorijuma i njegovog sadržaja. U primeru se koriste naredbe komandnog jezika operativnog sistema Unix, date u prethodnoj tački ove glave.

Primer 3.9. Neka je `/dev` tekući direktorijum, a potrebno je definisati novi direktorijum sa nazivom `knjiga`, koji će biti direktno podređen direktorijumu `/usr2`. Komandom

`[/dev] % cd /usr2`

sa Unix prompta, vrši se promena direktorijuma sa `/dev` na `/usr2`. Komandom

`[/usr2] % mkdir knjiga`

kreira se novi direktorijum sa imenom `knjiga` neposredno ispod direktorijuma `usr2`. Komandom

`[/usr2] % cd knjiga`

se menja tekući direktorijum sa `/usr2` na `/usr2/knjiga`. Inicijalno, direktorijum `/usr2/knjiga` je prazan.

Neka je na direktorijum `/usr2/knjiga` izvršen upis sledećih datoteka:

- `student.cpp`, koja sadrži izvorni C program za kreiranje datoteke `student.data`,
- `std`, koja sadrži objektni kod programa `student.cpp` i
- `student.data`, koja sadrži slogove studenata, prema strukturi na slici 3.1 u glavi 3.

Datoteka `student.cpp` je rezultat razvoja programa u nekom editoru teksta. Datoteka `std` je dobijena izvršenjem Unix komande

`[/usr2/knjiga] % g++ student.cpp -o std`

koja vrši kompilaciju programa `student.cpp` i smešta rezultat kompilacije pod imenom `std` na tekući direktorijum. Datoteka `student.data` je rezultat izvršenja programa `std` putem naredbe

`[/usr2/knjiga] % ./std`

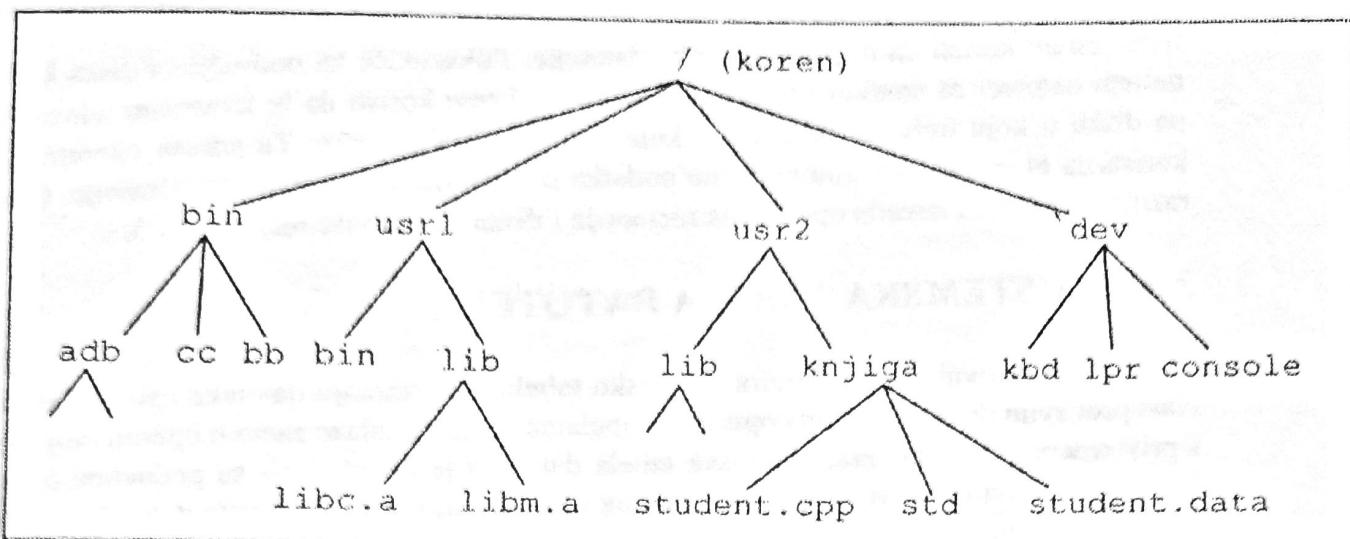
Ako se sada izda komanda

`[/usr2/knjiga] % ls knjiga`

dobija se

`std student.cpp student.data`

kao sadržaj direktorijuma `knjiga`. Na slici 3.6 je prikazana struktura direktorijuma nakon opisanih aktivnosti kreiranja novog direktorijuma i upisa datoteka u direktorijum na slici 3.5. □



Slika 3.6.

3.3 TABELE

Tabele predstavljaju mehanizam putem kojeg operativni sistem povezuje logičko ime datoteke, koje mu aplikativni program dostavlja kao parametar sistemskog poziva, sa datotekom na disku. U izvršavanju jednog sistemskog poziva, operativni sistem koristi:

- tabelu logičkih imena datoteka,
- tabelu otvorenih datoteka,
- tabelu opisa datoteka i
- sistemsku tabelu datoteke.

Operativni sistem koristi ove tabele u navedenom redosledu. Svaka od ovih tabela opisana je u daljem tekstu.

3.3.1 TABELA LOGIČKIH IMENA DATOTEKA

Tabela logičkih imena datoteka*) pripada aplikativnom programu. Sadrži parove (*logičko_ime_datoteke, pokazivač*), gde pokazivač ukazuje na jedan red tabele otvorenih datoteka. Formira je i delimično popunjava kompjajler prilikom prevodenja programa. Kompjajler generiše po jedan par (*logičko_ime_datoteke, null*) za svaku datoteku u programu. Tabela logičkih imena datoteka se nalazi unutar dela operativne memorije dodeljenog korisničkom programu.

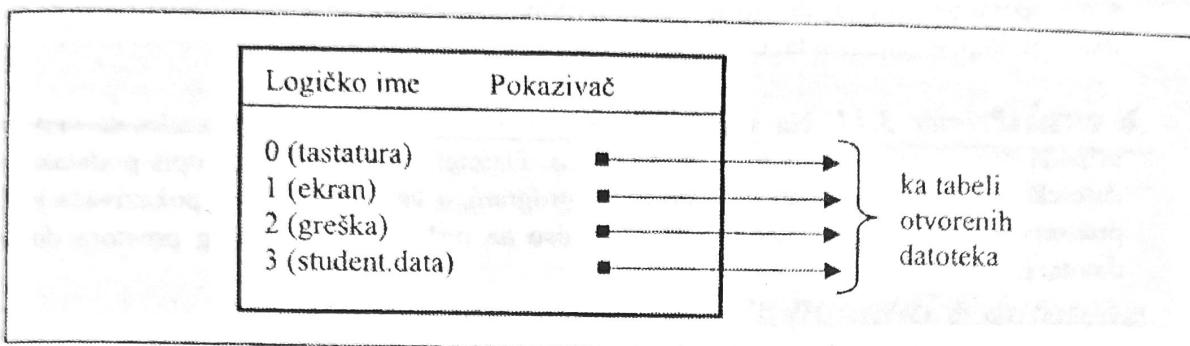
Operativni sistem ažurira sadržaj tabele logičkih imena datoteka pri otvaranju svake datoteke u programu, upisujući adresu odgovarajuće vrste u tabeli otvorenih datoteka.

Primer 3.10. Na slici 3.7 je prikazana tabela logičkih imena datoteka sa četiri para (*logičko_ime_datoteke, pokazivač*). Ovu tabelu bi mogao sadržati program za formiranje i ažuriranje datoteke *student.data* iz primera 3.1. Broj 3 predstavlja logičko ime datoteke

*) Tabela logičkih imena datoteka se u operativnom sistemu Unix naziva *file descriptor table*.

`student.data`. Njen pokazivač ka tabeli otvorenih datoteka je operativni sistem upisao pri izvršenju sistemskog poziva

```
open(student.data, O_APPEND | O_CREAT | O_RDWR);
```



Slika 3.7

Kada aplikativni program izda sistemski poziv tipa `read`, `write`, ili `lseek`, operativni sistem pristupa tabeli logičkih imena datoteka, da bi u njoj našao adresu odgovarajućeg reda u tabeli otvorenih datoteka.

3.3.2 TABELA OTVORENIH DATOTEKA

Tabela otvorenih datoteka pripada operativnom sistemu. Nalazi se u delu operativne memorije, koji koristi operativni sistem. Sadrži niz podataka za svaku otvorenu datoteku, bez obzira na to koji je program datoteku otvorio. Operativni sistem upisuje u ovu tabelu sledeće podatke pri otvaranju datoteke:

- način korišćenja,
- početna vrednost tekućeg pokazivača,
- pokazivači ka funkcijama za fizičku razmenu podataka sa memorijskim uređajem, na kojem se datoteka nalazi i
- pokazivač ka sistemskoj tabeli posmatrane datoteke.

Drugi parametar sistemskog poziva `open(char *ime_dat, int akt [int ovl])` definiše način korišćenja datoteke. Način korišćenja datoteke može biti neka kombinacija pisanja i čitanja, kako je to opisano u tački 3.1.1. U slučaju da je način korišćenja datoteke definisan kao upis na kraj datoteke, početna vrednost tekućeg pokazivača se postavlja na kraj datoteke, inače se postavlja na početak datoteke. Funkcije za fizičku razmenu podataka iniciraju stvarni upis i čitanje podataka sa diska. Za svaki tip disk uređaja, operativni sistem poseduje funkcije za upis i čitanje. U te funkcije su ugradene procedure za upravljanje fizičkom razmenom podataka. Svakom tipu uređaja odgovaraju specifične funkcije, koje vode računa o njegovim karakteristikama^{*)}. Konačno, tabela opisa datoteka sadrži dalje podatke o datoteci, koji su neophodni za izvršenje ulazno-izlazne operacije.

^{*)} Ove funkcije se nazivaju ulazno-izlaznim programima, a u okruženju operativnog sistema Unix, nazivaju se *device driver*.

Kada program izda poziv za upis, čitanje ili pozicioniranje datoteke, nakon pristupa tabeli logičkih imena, operativni sistem pristupa tabeli otvorenih datoteka. Tu proverava da li se zahtevana operacija sme izvršiti na datoteci, pronalazi tekući pokazivač, a ako je reč o pisanju ili čitanju, i adresu funkcije, koju će pozvati da izvrši operaciju. Međutim, za izvršenje same operacije pisanja, ili čitanja, operativnom sistemu su potrebni još neki dodatni podaci o datoteci, koji se nalaze u tabeli opisa datoteke.

Primer 3.11. Na slici 3.8 je prikazana tabela otvorenih datoteka sa vrstom, koja pripada otvorenoj datoteci student.data. Datoteka je otvorena za upis podataka na kraj datoteke i za čitanje, koristi je samo jedan program, a vrednost tekućeg pokazivača je 94, što predstavlja redni broj tekućeg bajta u odnosu na početak memorijskog prostora dodeljenog datoteci. □

Način korišćenja	Broj korisnika	Tekući pokazivač	Funkcija za pisanje	Funkcija za čitanje	Tabela opisa
upis na kraj i čitanje	1	94
...
...

Slika 3.8.

3.3.3 TABELA OPISA DATOTEKA

Tabela opisa datoteka*) pripada operativnom sistemu i nalazi se u operativnoj memoriji. Pri otvaranju datoteke, operativni sistem upisuje u tabelu opisa datoteka podatke, kao što su:

- identifikaciona oznaka disk uređaja, na kojem se datoteka nalazi,
- dozvole za korišćenje datoteke,
- vlasnik datoteke,
- veličina datoteke u bajtima,
- broj blokova datoteke i
- niz pokazivača ka onim područjima na disku, koji su dodeljeni datoteci za smeštaj podataka.

Sve ove podatke operativni sistem pronalazi u sistemskoj tabeli datoteke**).

Operativni sistem koristi podatke o pravima korišćenja da bi proverio da li program ima pravo da izvrši zahtevanu operaciju nad datotekom. Informaciju o veličini datoteke opera-

*) Tabela opisa datoteka se u operativnom sistemu Unix naziva *inode table*.

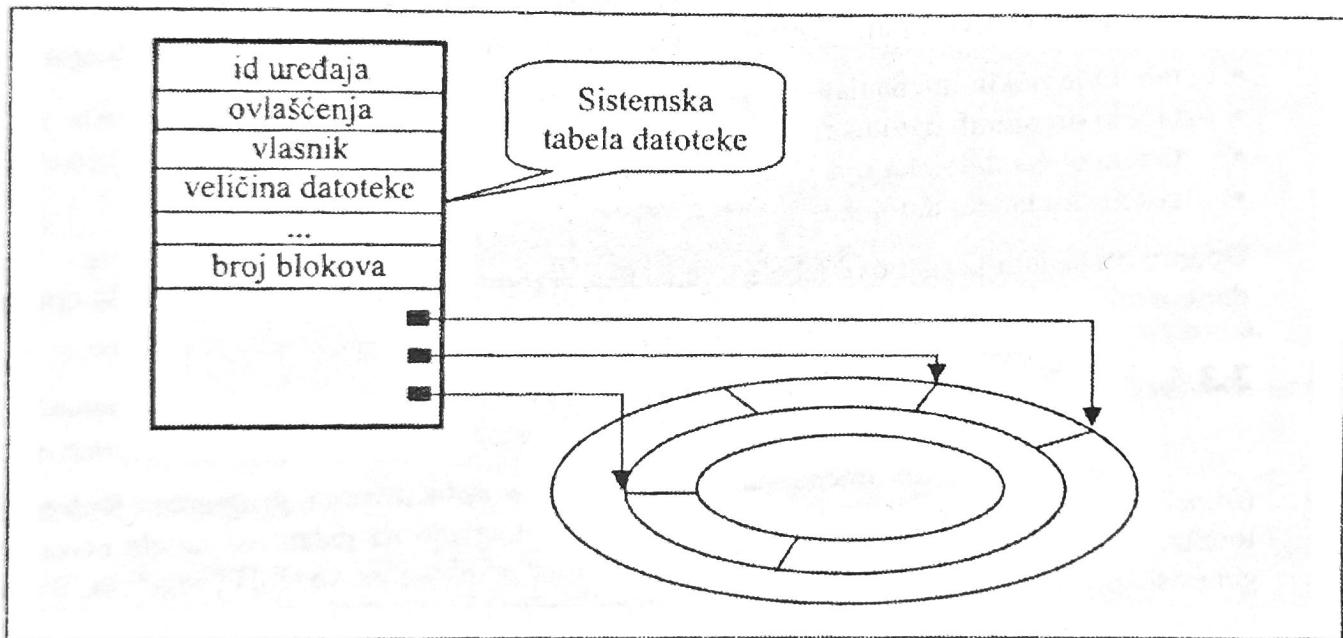
**) U operativnom sistemu Unix se sistemska tabela datoteke naziva *index node*, ili kratko *inode*.

tivni sistem koristi da bi pronašao kraj datoteke. Pokazivače ka područjima diska koji su dodeljeni datoteci za smeštaj podataka, operativni sistem koristi da bi izračunao adresu lokacije na disku u koju treba da piše, ili iz koje treba da čita podatke. Tu adresu operativni sistem koristi da bi inicirao stvarnu razmenu podatka između diska i operativne memorije. O stvarnoj razmeni podataka između operativne memorije i diska će biti više reči u glavi 5.

3.3.4 SISTEMSKA TABELA DATOTEKE

Operativni sistem formira sistemsku tabelu pri kreiranju datoteke i smješta je na disk. Nasuprot svim drugim, do sada opisanim tabelama, koje se nalaze samo u operativnoj memoriji i privremenog su karaktera, sistemskata tabela datoteke je trajan zapis sa podacima o datoteci. Operativni sistem učitava u operativnu memoriju sadržaj sistemske tabele datoteke pri svakom otvaranju datoteke i upisuje njen, eventualno, izmenjeni sadržaj nazad na disk po zatvaranju datoteke.

Na slici 3.9 je prikazana struktura jedne sistemskе tabele datoteke sa tri pokazivača ka tri područja na disku. Stvarni broj područja dodeljenih datoteci zavisi od njene veličine i veličine područja na disku i o tome će biti više reči u glavi 5. Smisao ostalih komponenata sistemskе tabele datoteke je opisan u prethodnoj tački, posvećenoj tabeli opisa datoteka.



Slika 3.9.