Operativni sistemi
- Fajl sistemi -

Veljko Stanković

Uvod

— Fajl predstavlja logicku jedinicu skladistenja.

- OS preslikava fajlove na medijume za skladistenje podataka.
- Predstavljaju najmanju jedinicu podataka koju korisnik moze da snimi ili joj pristupi na sekundarnim sistemima za skladistenje podataka (HDD, ODD...)
- Predstavljaju programe i podatke.
- Mogu biti numericki, alfanumericki i binarni.
- Mogu biti slobodne forme ili strogo uredjene strukture.
- Struktura fajla zavisi od njegovog tipa
  - ☑ Tekst fajl (text file)
  - ☑ Izvorni fajl (source file)
  - ☑ Objektni fajl (object file)
  - ☑ Izvrsni fajl (executable)

Uvod

— Podaci se mogu organizovati u vidu:

- Polje (Field)
  - ☑ Osnovni element predstavljanja podataka. Jedno polje može da čuva samo jednu vrednost (int, foat, char...).
  - ☑ Polje može biti fiksne ili promenljive dužine.
  - ☑ Kod polja promenljive dužine, ono se obično sastoji iz tri dela: podatka, imena polja i dužine polja. Umesto dužine polja moguće je koristiti i posebne delimitere između polja.
- Zapis (Record)
  - ☑ Predstavlja koelkciju polja koja se može posmatrati kao jedinstven entitet.
  - ☑ Nezavisno da li su fiksne dužine ili ne, uvek ima polje koje ukazuje na dužinu zapisa.
- Datoteka (File)
  - ☑ Predstavlja kolekciju zapisa organizovanih u logičku celinu.
- Baza podataka (Database)

Uvod

— Atributi fajlova variraju u zavisnosti od OS-a, ali se tipicno sastoje od:

- **Ime**
  - ☑ Simbolicko ime fajla. Jedina informacija koja je razumljiva za korisnika.
- **Identifikator**
  - ☑ Jedinstveni broj kojim se identifikuje fajl u okviru sistema.
- **Tip**
  - ☑ Ova informacija je potrebna kod sistema koji podrzavaju razlicite vrste fajlova.
- **Lokacija**
  - ☑ Pokazuje na uredjaj gde je smesten fajl i na lokaciju fajla na tom uredjaju.
- **Velicina**
  - ☑ Velicina fajla izrazena u bajtovima, recima ili blokovima i obicno maksimalna dozvoljena velicina fajla.
- **Zastita**
  - ☑ Informacija ko ima pravo pristupa fajlu (read, write, execute)
- Vreme, datum, identifikacija korisnika
  - ☑ Podaci krosini za zastitu i nadzor korisnika.

Uvod

1010011
1110100
1100001
1010110

— Podaci o fajlovima se čuvaju u okviru strukture direktorijuma, koja se nalazi takođe na sekundarnoj memoriji.

— Unos strukture direktorijuma se sastoji od imena fajla i njegovog identifikatora.

— Identifikator fajla određuje lokaciju ostalih podataka o fajlu.

Operacije nad fajlovima

1010011
1110100
1100001
1010110

- Kreiranje fajla
  - Rezervise se mesto za fajl u okviru fajl sistema.
  - Unose se podaci o fajlu u strukturu direktoijuma.
- Upis u fajl (write)
  - Poziva se sistemska rutina kojoj se kao parametri daju ime fajla i podaci koje treba upisati.
  - U okviru fajl mora postojati write pointer koji pokazuje gde treba upisati nove podatke.
    - ☑ Nakon svakog upisa podataka treba azurirati write pointer.
- Čitanje podatka iz fajla (read)
  - Sistemskoj rutini se daju kao ulazni parametri ime fajla i lokacija u GM gde treba smestiti sledeci blok podataka.
  - U okviru fajla se cuva read pointer koji ukazuje na poziciju sledeceg bloka podataka koje treba ucitati.

Operacije nad fajlovima

1010011
1110100
1100001
1010110

- Promena lokacije fajla
  - Trazi se pogodna lokaciju u strukturi direktorijuma i menj se vrednost current-file-position pointera.
  - Ne mora obavezno da zahteva izvrsavanje I?O operacija.
- Brisanje fajla
  - Oslobadaja se prostor na sekundarnoj memoriji.
  - Uklanja se unos o fajlu u strukturi direktorijuma
- Brisanje dela fajla
  - Omogucava korisniku da sacuva sve atribute fajla i obrise sve ili deo podataka sadrzanih u fajlu.
- Dodatne operacije sa fajlovima
  - Nadovezivanje, preimenovanje, kopiranje,

Operacije nad fajlovima

- Operacije nad fajlovima mogu zahtevati cesto pretrazivanje stabla direktorijuma.
  - Pojedini sistemi zahtevaju upotebu sistemskog poziva open() pre upotrebe fajla.
  - OS odrzava malu tabelu (open-file table) gde cuva podatke o otvorenim fajlovima.
    - ☑ Izbegava se pretrazivanje stabla direktorijuma.
  - Nakon zavrsetka rada sa fajlom on se zatvara
  - Operacije za rad sa zatvorenim fajlovima
    - ☑ create, delete

Operacije nad fajlovima

1010011
1110100
1100001
1010110

— Svakom otvorenom fajlu se pridružuju sledeći podaci:

- File pointer
  - ☑ Pkazuje na poslednju poziciju u fajlu kojoj je pristupljeno (read/write)
- File-open count
  - ☑ Prati broj procesa koji su otvorili isti fajl.
  - ☑ Kada vise nijedan proces ne koristi dati fajl, njegov identifikator u tabeli otvorenih fajlova (open-file table) se moze dodeliti nekom drugom fajlu kako tabela ne bi bila previse velika
- Lokacija fajla na disku
  - ☑ Čuva se u GM kako OS ne bi pristupao sekundarnoj memoriji svaki put kada treba da pristupi fajlu.
- Prava pristupa
  - ☑ Prati se prava pristupa svakom procesu koji pokusa da pristupi fajlu.

Tipovi fajlova

1010011
1110100
1100001
1010110

- Pri projektovanju OS-a moguce je odabrati da OS prepoznaje određene tipove fajlova.
  - Uobicajeni nacin da se odredi tip fajla jeste da se doda imenu fajla kao ekstenzija.
- Tip fajla u ekstenziji moze da ukazuje i na unutrasnju strukuru fajla.
  - Brojnost razlicitih tipova fajlova utice na slozenost OS-a.
    - ☑ OS mora da sadrzi kod za rad sa svakim od tipova fajlova.
  - OS-i obicno namecu i podrzavaju minimalni broj tipova fajlova.
  - Previse malo tipova fajlova moze dovesti do potreskoca prilikom progriranja jer programer mora da predvidi i sam ugradi odredjene usluge.

Interna struktura fajla

1010011
1110100
1100001
1010110

- Logička struktura fajla pokazuje kako su zapisi u okviru fajla organizovani.
- Od organiacije fajla zavisiće vreme pristupa podacima, lakoća ažuriranja, ekonomičnost skladištenja, održavanje i pouzdanost.
- Orgnizacija fajla:
  - Pile (gomila)
  - sekvencijalni fajl
  - indeksno-sekvencijalni fajl
  - indeksirani fajl
  - fajl sa direkstnim (hash-ovanim) pristupom

Interna struktura fajla

1010011
1110100
1100001
1010110

- Gomila
  - Najjednostavniji način organizacije podataka.
  - Podaci se zapisuju onako kako pristižu ili se prikupljaju.
  - Zapsi mogu biti različito orgnizovani pa svako polje mora imati sopstveni deskriptor.
  - Pošto ne postoji jasno definisana struktira fajla, zapisima se pristupa pretraživanjem svih zapisa.
- Sekvencijalni fajl
  - Svi zapisi imaju istu dužinu i strukturu polja čime je primena pojedinačnih deskpritora polja nepotrebna. Struktura zapisa, redosled i veličina polja predstavljaju atribute fajla.
  - Svakom polje se pridružuje ključ na osnovu koga se vrši pretraživanje. Pristup zapisima se vrši sekvencijlnim pretraživanjem ključeva.
  - Dodavanje podataka je problematično. Moguće je podatke prvo smestiti u log pile fajl pa ih potom uklopiti u strukturu fajla prateći ključ. Druga mogućnost jeste orgnizovanje u vidu linearne liste čime se pojednostalvjuje ažuriranje.

Interna struktura fajla

1010011
1110100
1100001
1010110

- Indeksno sekvencijlni fajl
  - Ima istu strukturu kao i sekvencijalni fajl.
  - Zapisi su orgnizovani prema ključu koj ise smešta u indeks koji se pridružujue fajlu.
  - Pretražiivanje i pristup zapisima se znatno ubrzava.
- Indeksirani fajl
  - Zapisima se pristupa samo na osnovu indeksa u kojima se pamte ključevi po kojima se može vršiti pretraživanja. Različiti indeksi odgovaraju različitim poljima u okviru zapisa kojise koriste kao parametar za pretraživanje.
  - Zapisi se mogu zapisivati na prozvoljnim pozicijama i ne postoji ograničenje u pogledu dužine zapisa.
- Kod fajlova sa direktnim pristupom, zapisima se pristupa na osnovu hašovane vrednosti ključa čime se znatno ubrzava pretraživanje.

Pristup fajlovima

1010011
1110100
1100001
1010110

- Kako pristupiti podacima u fajlu
  - Sekvencijalni pristup
    - ☑ Najjednostavniji i najcesci nacin pristupa podacima.
    - ☑ Podacima se pristupa po odredjenom redu, zapis po zapis.
  - Direktni (relativni) pristup
    - ☑ Fajl se sastoji iz logickih blokova fiksne duzine i omogucava programiam da pristupaju podacima brzo i po proizvoljnom redosledu.
    - ☑ Posmatra se kao numerisana sekvenca blokova ili zapisa.
    - ☑ Baze podataka se obicno organizuju na ovaj nacin.

Struktura direktorijuma

- Ponekad je pozeljno da se na disk smesti više fajl sistema ili da se razliciti delovi HDDa koriste za različite namene.
  - swap protstor ili neformatirani prostor
- Delovi diska koji se koriste za različite fajl sisteme ili različite namene zovu se particije (partition, slices, minidisks).
  - Svaka particija mora da sadrži informacije o fajlovima koji se nalaze na njoj. Ta infromacija se smešta u direktorijum uređaja (**device directory**).
  - Direktorijum sadrži informacije o zapisima na HDD
    - ☑ Ime, lokacija, velicina, tip svih fajlova na HDD.
    - ☑ Može biti organizovan na više načina.

Struktura direktorijuma

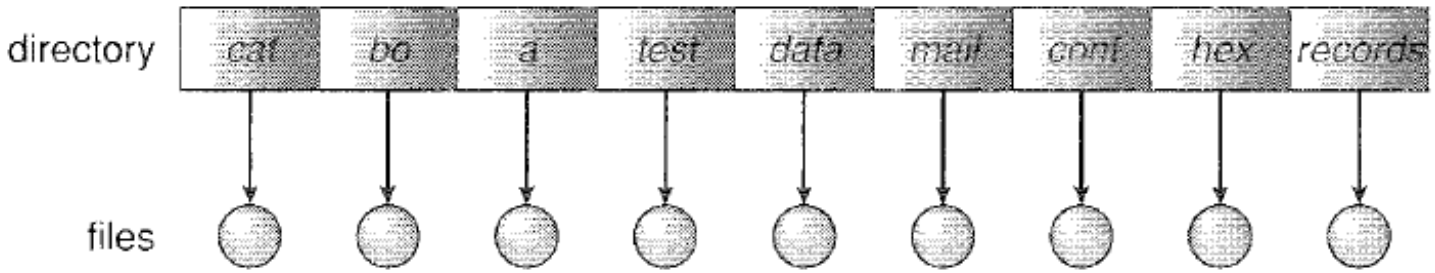
— Operacije nad direktorijumima:

- Potraga za fajlom
- Kreiranje fajla
  - ☑ Kako dodati nove fajlove u direktorijum?
- Brisanje fajlova
  - ☑ Kada vise nisu potrebni, moramo biti u mogucnosti da ih obrisemo.
- Prikazivanje sadržaja direktorijuma
  - ☑ Moramo biti u mogucnosti da vidimo spisak svih fajlova i podataka o njima u direktorijumu.
- Promena imena fajla
  - ☑ Korisnik mora biti u stanju da proemni ime fajla u zavisnosti od njegove funkcije.
  - ☑ Prir promeni imena moze biti potrebno da fajl promeni svoj polozaj u strukturi direktorijuma.
- Pretraživanje i ažuriranje sistema

Struktura direktorijuma
- Logicka organizacija direktorijuma -

— Jednonivooska oragnizacija

- Svi fajlovi se nalaze u jednom direktorijumu
- Najjednostavnija organizacija
- Problemi nastaju sa porastom broja korisnika i fajlova
  - ☑ Problem dodele jedinstvenog imena svakom fajlu
  - ☑ OS obicno dozvoljava imena fjalova odredjene duzine (MSDOS – 11, UNIX – 256)
- Problem pracenja velikog broja fajlova u jednom direktorijumu.



Struktura direktorijuma
- Logicka organizacija direktorijuma -

1010011
1110100
1100001
1010110

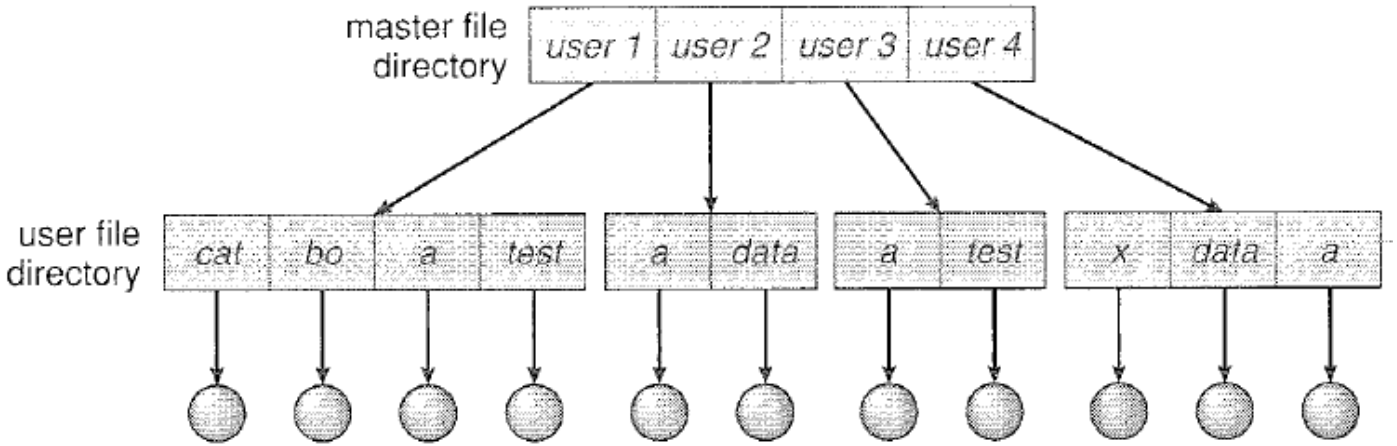
— Dvonivooska organizacija

- Svaki korisnik ima svoj direktorijum (user file directory – UFD)
- Kada korisnik pokrene neku obradu, prvo se pretražuje glavni direktorijum (master file directory), a potom korisnički direktorijum.
- Odvajanje prostora korisnika je korisno sem u slučaju kada oni žele da kooperiraju i pristupaju fajlovima koji pripadaju drugim korisnicima
- Fajlovima se pristupa na osnovu imena korisnika i imena fajla, čime se definiše put (**path**) u stablu direktorijuma.
- Problem sa izvršavanjem sistemskih programa.
  - ☑ Jedna mogućnost je da se uz svaki UFD smeste sistemski fajlovi.
  - ☑ Formira se poseban direktorijum sa sistemskim fajlovima i definise redosled po kome ce direktorijumi biti pretrazivani (**search path**).
    - Svaki korisnik moze imati svoj search path
    - Najcesce koriscen metod kod MSDOS i UNIX.

Struktura direktorijuma
- Logicka organizacija direktorijuma -

1010011
1110100
1100001
1010110

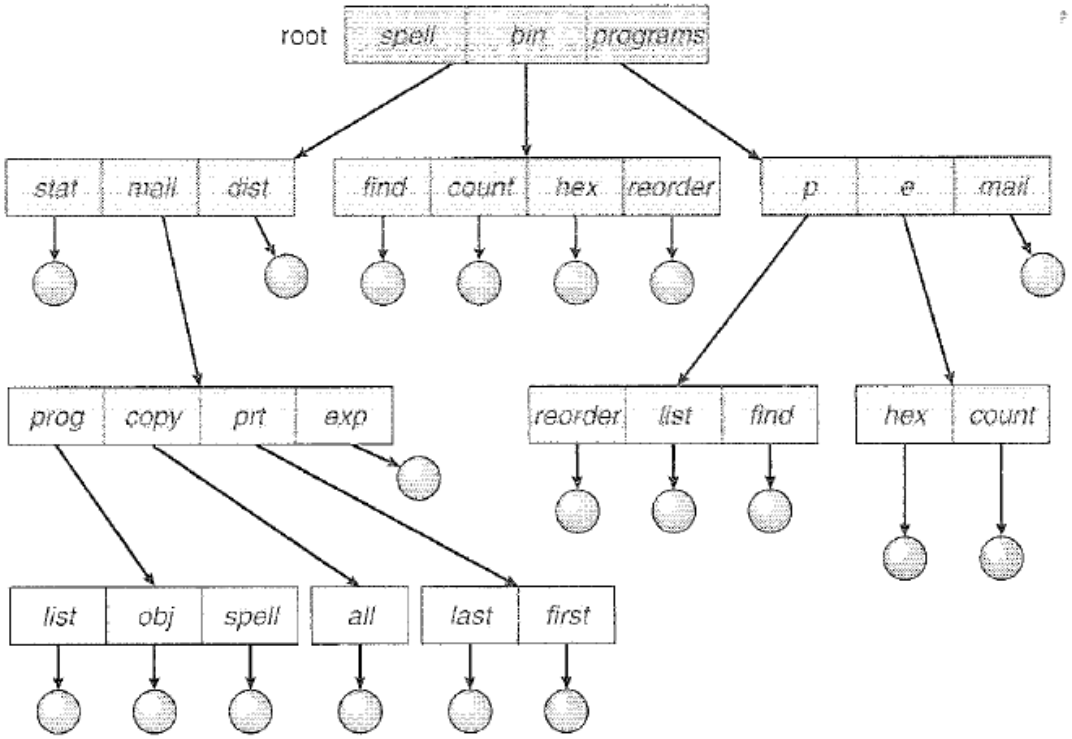
— Dvo nivooska organizacija



Struktura direktorijuma
- Logicka organizacija direktorijuma -

— Hijerarhijska organizacija direktorijuma

- Direktorijumi organizovani u više hijerarhijskih nivoa.
- Svaki direktorijum sadrži poddirektorijume i fajlove.
- Svaki fajl je na jedinstven način određen putanjom do njega (file path name).



Struktura direktorijuma
- Logicka organizacija direktorijuma -

1010011
1110100
1100001
1010110

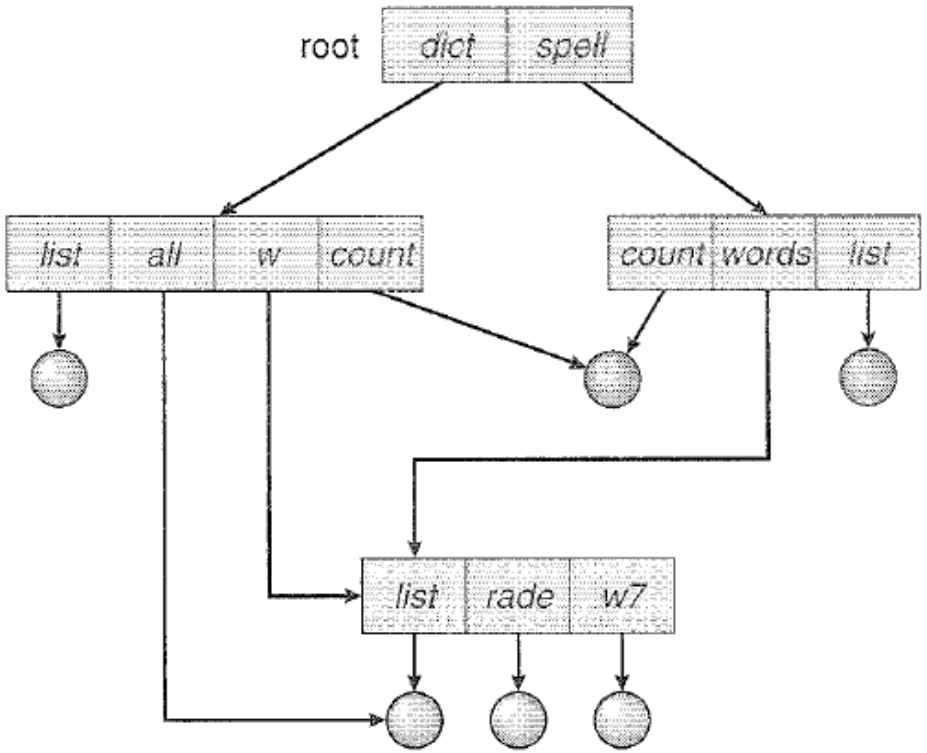
— Hijerarhijska organizacija direktorijuma

- Sistemski poziva za promenu radnog direktorijuma.
- Ako trazeni fajl nije u radnom direktorijumu korisnik moze navesti putanju do njega ili ce OS pretrazivati direktorijume po utvrdjenom redosledu (search path).
- Imena putanja do fajlova (path names) mogu biti:
  - ☑ Apsolutna
    - Daje kompletnu putanju u hijerarhiji direktorijuma do zeljenog fajla pocev od osnovnog (**root**) direktorijuma
  - ☑ Relativna
    - Definise putanju do fajla pocev od radnog direltorijuma.
- Pitanje brisanja direktorijuma
  - ☑ Moze biti obrsian samo ako je prazan (MSDOS)
  - ☑ Automatski se brisu svi poddirektorijumi i fajlovi (UNIX)

Struktura direktorijuma
- Logicka organizacija direktorijuma -

— Aciklicni direktorijumi (Acyclic-graph directories)

- Omogucava direktorijumima da dele poddirektorijume i fajlove, tj. da zapisi u direktorijumima pokazuju na iste fizicke entiete na HDD.



Struktura direktorijuma
- Logicka organizacija direktorijuma -

1010011
1110100
1100001
1010110

— Aciklicni direktorijumi (Acyclic-graph directories)

- Sve promene u fajlu mogu da prate svi kroisnici/procesi koji mu pristupaju.
  - ☑ U klasicnoj hijerarhijskoj strukturi, mogu postojati kopije istog fajla u razlicitim direktorijumima.
- Narocito je vazno deljenje poddirektorijuma, jer ce svi korisnici istovremeno moci da vide ako se kreira neki novi fajl ili direktoijum.
- Zajednicki fajlovi i direktorijumi se mogu implementirati na vise nacina
  - ☑ U direktorijumu se unosi novi zapis koji se zove **link**, koji predstavlja pointer na drugi fajl ili direktorijum.
    - Kada prilikom pretrazivanja direktorijuma naidjemo na link, njegovo ime se ukljucuje u putanju fajla.
    - Problem sa pointerima nakon brisanja fajla/poddirektorijuma. Moraju se svi blagovremeno azurirati.
  - ☑ Formira se potreban broj nezavisnih kopija fajla/poddirektorijuma.
    - Sve promene se moraju balgovremeno azurirati kod svih kopija → problem konzistentnosti.

Struktura direktorijuma
- Logicka organizacija direktorijuma -

1010011
1110100
1100001
1010110

— Aciklicni direktorijumi (Acyclic-graph directories)

- Problem brisanja zajednickih fajlova/poddirektorijuma je lakse resiti ako su oni implementirani uz pomoc linkova.
  - ☑ Brise se samo odredjeni link.
  - ☑ Ako se obrise konkretni fajl, potrebno je obrisati i sve linkove.
    - Ukoliko se informacija o linkovima ne cuva uz fajl, pretraga svih linkova moze biti veoma zahtevna.
    - Linkovi se ne diraju, vec kada korisnici pokusaju da pristupe fajlovima koji us obrisani, tada ih OS obavesta i brise link.
- Drugi pristup jeste da se fajl ne brise sve dok postoje linkovi koji vode ka njemu.
  - ☑ Potrebno da uz svaki fajl imamo spisak linkava ka njemu, ili da imamo neki mehanizam da utvrdimo kada je zadnja referenca izbrisana.

— Veliki problem predstavlja ogranicenje da u grafu ne sme biti petlji.

- Moze se ograniciti broj poddirektorijuma u kojima se trezi zeljeni fajl.

File-system mounting

- Na slican nacin kao sto fajl moramo da otvorimo pre nego sto mu pristupimo, fajl sistem se mora ucitati (mount) pre nego sto moze da mu se pristupi.
  - Struktura direktotijuma se moze sastojati iz vise delova (volume), i savkome se mora pristupiti kako bi bili dostupni fajl-sistem domenu.
- OS-u se daje ime uredjaja (device) i tacka (mount point) unutar fajl sistem gde se nadovezati novi fajl sistem.
  - Obicno je mount pont prazan direktorijum.
- OS proverava da li novi fajl sistem ima odredjenu strukturu.

Implementacija fajl sistema

- Fajl sistem se permanantno nalazi na sekundarnoj memoriji (HDD/ODD) koja je namenjena skladištenju velikih količina podataka.
- Mehanizmi za:
  - primenu fajl sistema,
  - dodelu prostora na HDD
  - oslobadjanju prostora na HDD
  - nalazenje podataka na HDD
  - Povezivanje delova OS sa sekundarnom memorijom.

Implementacija fajl sistema

— Upravljanje fajl sistemom trebe da omogući:

- Skladištenje podataka i operacije nad njima.
- Da garantuje integritet podataka.
- Da optimizuje performase sistema u pokledu količine podataka i vremena odziva na zahtev korisnika.
- Da omogući primenu različitih tipova uređaja za skladištenje podataka.
- Da eliminiše ili smanji mogućnost izgubljenih ili uništenih podataka.
- Da standardizuje skup rutina za pristup podacima.
- Da kod všekorisnički sistema omogući pristup podacima od strane više korisnika.

Implementacija fajl sistema

- Minimalan set zahteva za upravljanje i pristup podacima:
  - Svaki korisnik treba da bude ustanju da kreira, obriše, očita, upiše i modifikuje sadržaj fajlova.
  - Svaki korisnik treba da ima mogućnost kontrolisanog pristupa fajlovima drugog korisnika.
  - Korinisk trea ad bude ustanju da kontroliše pravo pristupa drugih korisnika svojim fajlovima.
  - Korinisk treba da bude u stanju da promeni strukturu fajlova prema potrebama.
  - Korisnik treba da bude u stanju da premešta podatke između fajlova.
  - U slučaju oštećenja falova korisnik treba da bude ustanju da napravi rezervne kopije i oporavise od gubitka podataka.
  - Omogući pristup fajlovim na osnovu imena a ne na osnovu numeričkog identifikatora.

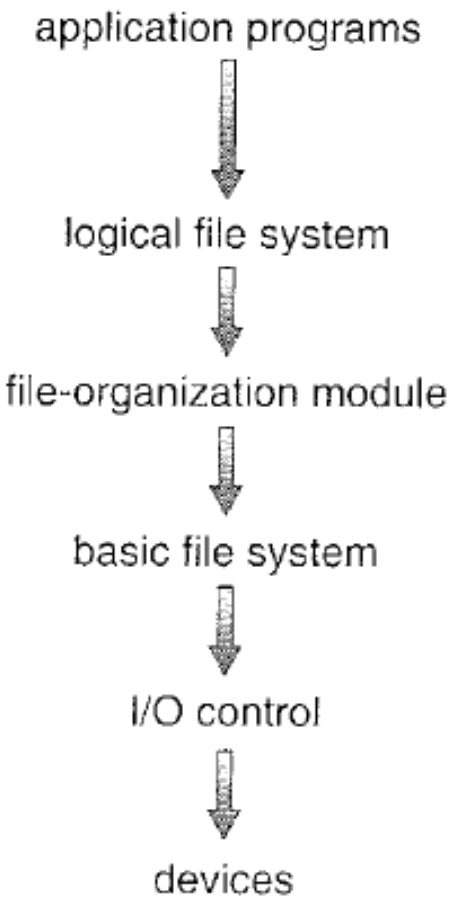
Implementacija fajl sistema

1010011
1110100
1100001
1010110

- Prednosti primene HDD
  - Podaci na nekoj lokaciji se mogu citati, modifikovati ili upisati novi u isti blok.
  - Omogucava direktan pristup podacima.
- Da bi se poboljsala I/O efikasnost, prenos podataka izmedju GM i HDD se vrsi u blokovima.
- Svaki blok ima jedan ili vise sektora.
  - Velicina sektora varira od 32B do 4096B, tipicno 512B.
- Fajl sistem je obicno organizovan iz vise nivoa.

Implementacija fajl sistema

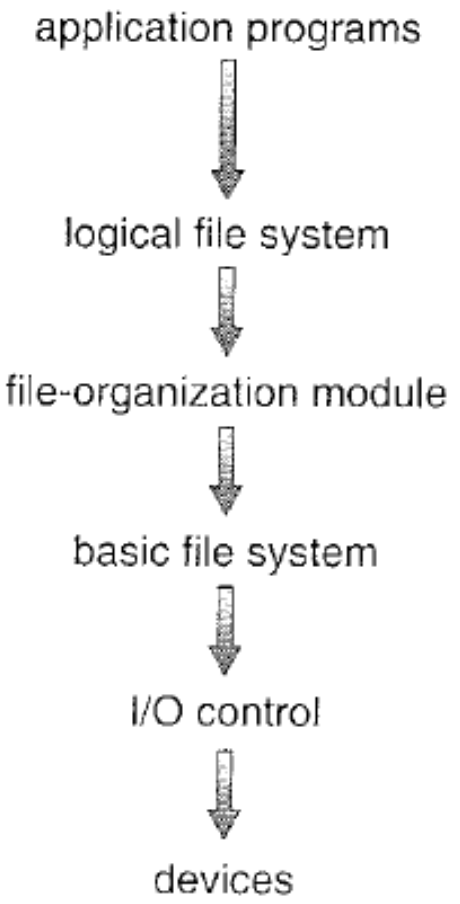
- I/O kontrola
  - sastoji se iz drajvera uređaja i rutina za opsluživanje interapta (interupt handlers) čija je svrha transfer podataka izmedju GM i HDD.
- Osnovni fajl sistem (basic file system)
  - Izdaje osnovne komande drajveru HDDa da upiše/očita određene fizičke blokove u/sa HDD iz/u GM..
- Organizacioni modul (file-organization module)
  - Definiše vezu između logičkih i fizičkih blokova fajlova.
  - Na ovom nivou se upravlja svim strukturama potrebnim za I/O, raspoređivanjem zadataka za I/O i status fajlova.
  - Na osnovu informacija o načinu dodeljivanja prostora na HDD, lokaciji fajla, organizacioni blok prevodi logičku u fizičku adresu bloka koji osnovni fajl sistem treba da prenese, tj da mu pristupi.



Implementacija fajl sistema

1010011
1110100
1100001
1010110

- Logicki fajl sistem
  - Upravlja metapodacima (metadata information).
  - Azurira i upravlja strukturom direktorijuma.
    - ☑ Na osnovu simbolickog imena fajla daje organizacionom modulu potrebne podatke.
  - Svakom fajlu se dodeljuje **file-control block** (FCB) u kome čuva informacije o fajlu kao što su dozvola pristupa i lokacija sadržaja fajla.
  - Zadužen za zaštitu informacija.
- Primeri
  - ODD: ISO 9660
  - UNIX: UNIX file sistem (UFS), (Berkeley) Fast File System (FFS)
  - MS Windows: FAT, FAT32, NTFS (Windows NT FS)
  - Linux: najcesce extended file system (ext4)



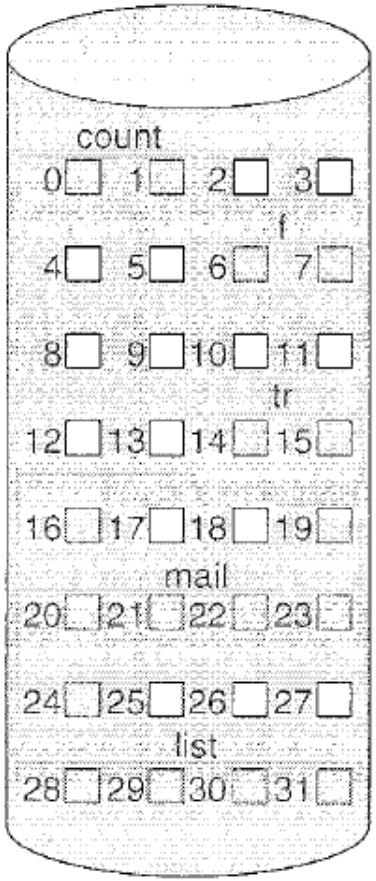
Dodela memorije fajlovima

1010011
1110100
1100001
1010110

- Kako dodeliti prostor na HDD-u tako da se on iskoristi sto je moguce bolje i da se fajlovima moze pristupiti sto brze?
- Kontinualna alokacija
  - Svaki fajl zauzima skup sekvencijalnih blokova na HDD.
  - Broj pretražiavanja HDDa je minimalan jer su blokovi jedan do drugog.
  - Lokacija fajla je određena adresom prvog bloka na disku i brojem blokova.
  - Podržava sekvencijalni i direktni pristu podacima.
  - Slična problematika kao kod dodele GM
    - ☑ Eksterna fragmentacija
    - ☑ Interna fragmentacija
  - Problem pri kreiranju fajla kako dorediti potreban prostor (veličina fajla može da varira).

Dodela memorije fajlovima

— Kontinualna alokacija



directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Dodela memorije fajlovima

1010011
1110100
1100001
1010110

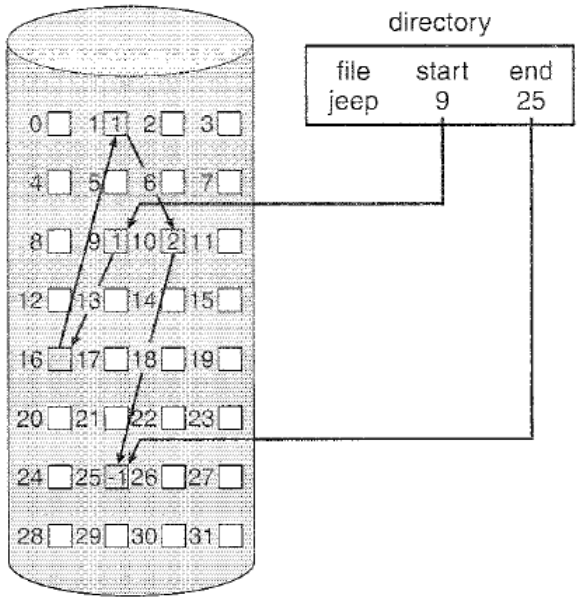
— Povezana alokacija

- Fajlovi se smeštaju na disk u vidu povezane liste.
  - ☑ Svaki blok podataka sadrži na kraju pointer na sledeći blok podataka.
  - ☑ Zadnji blok sadrži Null pointer koji ukazuje na kraj fajla.
- Direktorijum sadrži pointere na prvi i na poslednji blok fajla.
- Nema eksterne fragmentacije.
- Nema potrebe da se veličina fajla unapred definiše.
- Nedostaci:
  - ☑ Omogućava samo sekvencijalni pristup podacima unutar fajla.
  - ☑ Unutar svakog bloka se troši odredjeni prostor za smeštanje pointera.
    - Blokovi se dodeljuju u grupama – klasterima.
  - ☑ Osetljiva na grešku pri očitavanju pointera.

Dodela memorije fajlovima

— Povezana alokacija

- FAT sistem (File allocation table)
  - ☑ Na početku svake particije se smešta tabela u kojoj se nalaze pointeri a indeksirana je rednim brojem bloka na HDD.
  - ☑ Svaki fajl je odredjen adresom prvog bloka.
  - ☑ Na istoj adresi u FAT tabeli se ocitava vrednost pointera na sledeci blok.
  - ☑ Adresa zadnjeg bloka fajla u FAT tabeli sadrzi Null pointer.
  - ☑ Zahteva veliki broj pristupa HDDu.
  - ☑ Moguca direktan pristup podacima.



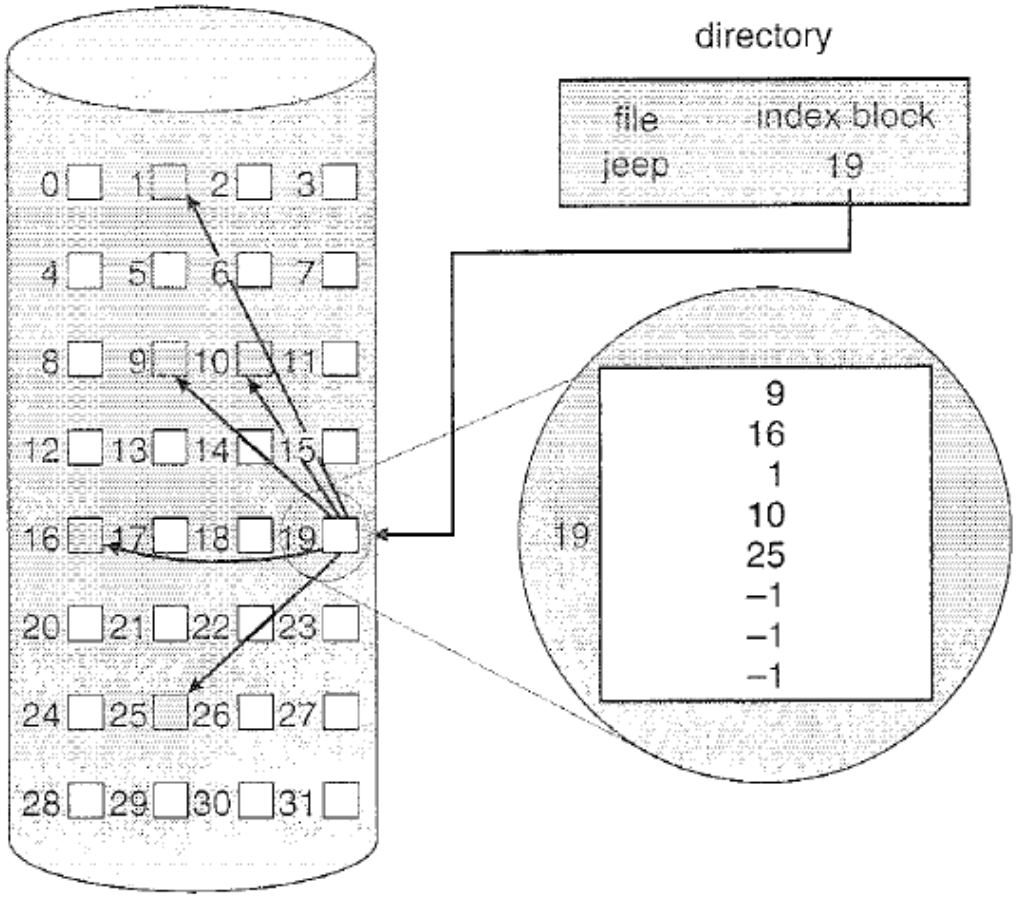
Dodela memorije fajlovima

— Indeksirana alokacija

- Povezana alokacija razresava problem eksterne fragmentacija ali nema mogucnost direktnog pristupa podacima.
- Da bi omogucili direktan pristup podacima, pointeri svih blokova koji su dodeljeni fajlu se smestaju na jedno mesto: blok indeksa.
- Nema eksterne fragmentacije.
- Problem: Koliko veliki treba da bude blok indeksa?

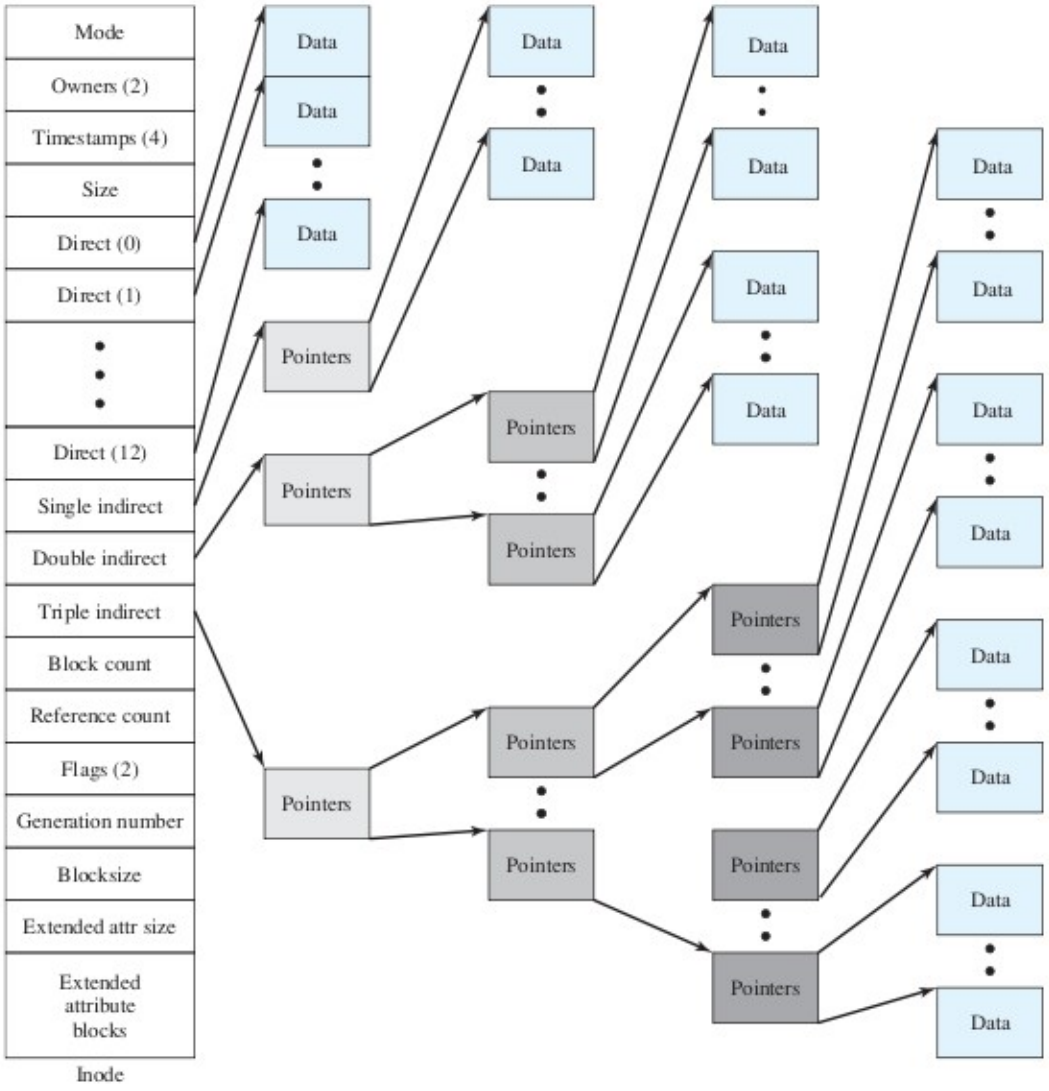
Dodela memorije fajlovima

— Indeksirana alokacija



UNIX inode

1010011
1110100
1100001
1010110



1010011
1110100
1100001
1010110

1010011
1110100
1100001
1010110

Implementacija fajl sistema

- Za implementaciju fajl sistema se koriste nekoliko struktura podataka koje se nalaze na HDD i u GM.
- Delovi fajl sistema na HDD
  - **Boot control block** – sadrži podatke neophodne da bi sistem mogao da učita OS sa te particije. Obično se radi o prvom bloku particije.
    - ☑ NTFS: partition boot sector
  - **Volume (partition) control block** – sadrzi detalje o particiji
    - ☑ Broj blokova u particiji
    - ☑ Velicina blokova
    - ☑ Broj i pointeri na slobodne blokove
    - ☑ Slobodni FCB i pointeri na slobodne FCB.
    - ☑ Primer:
      - UNIX: **superblock**
      - Win: **master file table**

Implementacija fajl sistema

— Delovi fajl sistema na HDD

- Struktura direktorijuma koja sluzi da bi organizovali fajlove.
  - ☑ Primer:
    - UNIX: imena fijalova i **inode** brojeve.
    - Win: nalazi se u master file table.
- FCB svakog fajla
  - ☑ Vlasnik fajla
  - ☑ Kontrola pristupa
  - ☑ Velicina
  - ☑ Lokacija blokova podataka
  - ☑ Primer:
    - UFS: inode
    - NTFS: relaciona baza podataka u okviru master file table

Implementacija fajl sistema

- Podaci fajl sistema u GM sluze za upravljanje fajl sistemom i poboljsavanje performansi uz pomoc kes memorije.
  - Mount table sadri informacije o svim particijama kojima smo pristupili.
  - Informacija o direktorijumima kojima smo skoro pristupali.
  - Kopiju FCB svakog otvorenog bloka i dodatne infromacije se cuvaju u **system-wide open-file table**.
  - Pointer ka svakom otvorenom fajlu prema pripadnostima odredjenim procesima se cuvaju u **per-process open-file-table**.

Implementacija fajl sistema

1010011
1110100
1100001
1010110

- Particije i pristupanje particijama (mount)
  - Disk moze biti podeljen na vise particija ili se jedna particija moze nalaziti na vise diskova.
  - Svaka particija moze biti “sirova” (raw), tj bez deifnisanog fajl sistema ili sa definisanim fajl sistemom (cooked).
  - Sirova particija se koristi onda kada to odgovara potrebama korisnika ili OS-a
    - ☑ swap particija UNIXa
- Boot informacija se smesta u posebnoj particiji sa posebnim fajl sistemom jer pri iniciranju sistem nema predefinisani fajl sistem.
  - Obicno predstavlja sekvencu blokova koji se kao slika ucitavaju u memoriju.
  - Root particiji je particija koja sadrzi OS se pristupa (mount) u toku boot-ovanja.
  - Nakon bootovanja, OS proverava da li HDD sadrzi validan fajl sistem na taj nacin sto ucitava strukturu direktorijuma i proverava da li zadovljava odredjenu formu.

Implementacija direktorijuma

1010011
1110100
1100001
1010110

- Izbor algoritama za azuriranje i upravljanje direktorijuma bitno utiče na efikasnost, performanse i pouzdanost fajl sistema.
- Linearna lista
  - Lista imena fajlova sa pointerima ka odredjenim blokovima podataka.
  - Pronalazenje fajla zahteva linearno pretrazivanje.
  - Problemi pri kreiranju, brisanju ili modifikovanju fajla.
- Hash tabela
  - Direktorijum pamti podatke o fajlovima u obliku liste.
  - Na osnovu imena svakog fajla proracunava se hash vrednost koja se koristi pri indeksiranju.
    - ☑ Znatno smanjuje vreme pretrage.
    - ☑ Problem se javlja kada dodje do kolizije, tj. kada imena dva i vise fajlova kao rezutlat hash operacije daju isti broj.