4.14 Fajl sistemi

I-nod sadrži sve informacije o fajlu. Za svaki i-nod se čuva broj linkova, koji predstavlja broj direktorijumskih odrednica (directory entries) koji pokazuju na taj i-nod. Samo kada se broj linkova smanji na 0, fajl može biti obrisan. Dakle, operacija "unlinking" ne znači uvek i brisanje blokova koji čine fajl. Funkcija koja briše direktorijumsku odrednicu se zove *unlink*. U strukturi stat broj linkova se čuva u polju **st_nlink**. Ovaj tip linkova zove se **hard linkovi**. Drugi tip linkova su simbolički linkovi.

U i-nodu se čuvaju tip fajla, prava pristupa, veličina, pokazivači na blokove fajla, itd. Većina informacija u stat strukturi se dobija iz i-noda. U direktorijumskom zapisu se čuvaju samo **ime fajla** i **i-nod broj.** Ograničenje je da možemo da imamo samo direktorijumske odrednice koje pokazuju na i-nodove na istom fajl sistemu. Kada menjamo ime fajla bez promene fajl sistema, sadržaj fajla se ne pomera, već se jedino dodaje nova direktorijumska odrednica koja pokazuje na postojeći i-nod, i unlink-uje stara direktorijumska odrednica. Ovako funkcioniše komanda mv.

Svaki direktorijum ima broj linkova barem dva jer na njega pokazuje direktorijumska odrednica koja čuva ime direktorijuma (to je roditeljski direktorijum) i u samom tom direktorijumu odrednica '.' . Svaki direktorijum koji ne sadrži poddirektorijume ima tačno 2 linka, dok direktorijum koji sadrži poddirektorijum ima broj linkova barem 3.

4.15 link, unlink, remove i rename funkcije

Link na postojeći fajl se kreira pomoću **link** funkcije. Kreiranje nove direktorijumske odrednice i povećanje broja linkova je **atomična operacija**. Kreiranje hard linkova za direktorijume je dozvoljeno samo superuseru. Razlog je što bi zlonameran korisnik mogao da napravi kružne petlje u sistemu, koje većina alata koji rade sa fajl sistemom ne mogu da otkriju.

Komanda **unlink** briše direktorijumsku odrednicu i smanjuje broj linkova fajla koji se prima kao argument. Ukoliko postoji još linkova na fajl, fajl je i dalje dostupan. Samo smanjivanjem broja linkova na 0, sadržaj fajla se briše.

Ukoliko proces ima fajl otvoren, onda se taj fajl ne može obrisati tokom izvršavanja tog procesa. Tek kada se fajl zatvori, kernel proverava broj procesa koji imaju ovaj fajl otvoren, ukoliko je on 0, sadržaj fajla se briše.

(PRIMER – unlink)

df .

Funkcija *unlink* se često koristi da se privremeni fajl koji kreira program obriše ako dođe do pucanja programa. Proces kreira fajl pomoću *open* ili *creat* i odmah poziva *unlink*.

Za unlink-ovanje fajla može se koristiti i funkcija ***remove***. Za fajl, remove je identična unlink, a za direktorijum je identična *rmdir.*

Ime fajla ili direktorijuma menjamo pomoću **rename** funkcije.

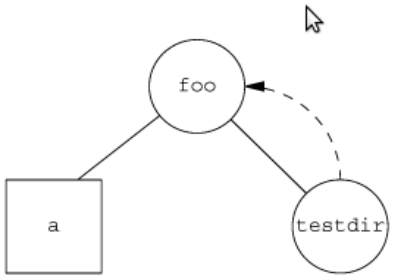
4.16 Simbolički linkovi

To su indirektni pokazivači na fajlove. Uvedeni su da prevaziđu nedostaci hard linkova:

1. Hard linkovi zahtevaju da se link i fajl nalaze na istom fajl sistemu
2. Samo superuser moze da kreira hard link za direktorijum

Postoje funkcije koje prate simboličke linkove i one koje to ne rade. Možemo da napravimo petlje u fajl sistemu koristeći simboličke linkove. Većina funkcija postavlja vrednost *errno* na ELOOP kada dođe do ovoga.

mkdir foo
touch foo/a
ln -s ../foo foo/testdir
ls -l foo



Ovakva petlja se lako odstranjuje, dovoljno je da pozovemo unlink, jer ova funkcija ne prati simboličke linkove. Međutim, petlja nastala preko hard linkova mnogo se teže briše.

4.18 Vremena fajlova

Polja za vreme se čuvaju za svaki fajl.

Field	Description	Example	ls(1) option
st_atime	last-access time of file data	read	-u
st_mtime	last-modification time of file data	write	default
st_ctime	last-change time of i-node status	chmod, chown	-c

Vreme modifikacije (**st_mtime**) označava kada je poslednji put menjan sadržaj fajla. Vreme promene statusa (**st_ctime**) označava kada je poslednji put menjan i-nod (promena prava pristupa, vlasnika, broja linkova).

Vreme posledjeg pristupa (**st_atime**) može iskoristiti administrator sistema da obriše fajlove kojima se nije skorije pristupalo (npr. a.out fajlovi kojima se nije pristupalo u prethodnoj nedelji).

ls -l – vreme modifikacije
ls -lu – vreme pristupa
ls -lc – vreme promene statusa

4.19 Funkcija utime

Vremena pristupa i modifikacije mogu se promeniti pomoću **utime** funkcije. Vremena u strukturi **struct utimbuf** u koju se smeštaju ova dva vremena, su predstavljena sekundama od Epohe, tj. od

1.1.1970. godine.

1. Ukoliko je **times** argument null pokazivač, oba vremena se postavljaju na trenutno vreme. Effective user ID mora biti jednak vlasniku fajla ili moramo imati prava pisanja za fajl.
2. Inače su vremena postavljena na vrednosti u strukturi na koju pokazuje times. Effective user ID mora biti jednak vlasniku fajla ili proces mora imati superuser privilegije.

Vreme promene statusa i-noda se automatski menja kada se pozove funkcija utime.

Za dobijanje trenutnog vremena može se koristiti funkcija **time**. Ona vraća broj sekundi od Epohe u trenutku u kome se poziva.

```
time_t time(time_t *t);
```

4.21 Čitanje direktorijuma

Direktorijume može da čita svako ko ima prava čitanja za taj direktorijum. Na različitim sistemima različito su predstavljene direktorijumske odrednice. Zbog toga u mnogim Unix-ima *nije dozvoljeno da programi koriste read funkciju pri pristupanju direktorijumima.* Za prolazak kroz direktorijume koriste se funkcije *opendir* i *readdir.*

Dirent struktura mora sadrzati bar dva polja:

```
struct dirent {
  ino_t d_ino;                    /* i-node number */
  char  d_name[NAME_MAX + 1];     /* null-terminated filename */
}
```

Pomoću opendir funkcije pristupa se direktorijumskim odrednicama, a redosled elemenata zavisi od implementacije.

(PRIMER types_counter)

Uputstvo: program **ne** testirati na **/** ili nekom direktorijumu koji sadrži mnogo poddirektorijuma.

./fts /dev/disk
./fts /dev/input
./fts ~/public_html

VAŽNO: Primetiti da su promenljive *directory* i *entry* u gornjem primeru samo pokazivači, tj. da nije alocirana memorija za njih! To je zbog toga što funkcije *opendir* i *readdir* vraćaju pokazivače na statički alociranu memoriju, pa korisnik ne mora voditi računa o alokaciji/dealokaciji memorije.

NAPOMENA: Nećemo proveravati greške za *readdir* funkciju, to bi se radilo tako što se zapamti vrednost errno pre korišćenja ove funkcije, a po završetku korišćenja proverilo da li je ta vrednost izmenjena. Ako jeste, došlo je do greške.

Probati da se u gornjem primeru koristi funkcija *stat* umesto *lstat*. Zašto može doći do problema kada postoje simbolički linkovi poput onih kreiranih kao u poglavlju 4.16?

Obezbeđena je funkcija **ftw(3)** (file tree walk) koja prolazi kroz hijerarhiju i zove funkciju koju je definisao korisnik za svaki fajl. Ova funkcija prati simboličke linkove (pa će se neki fajlovi brojati dva puta), pa postoji funkcija **nftw(3)** koja poseduje opciju za nepraćenje simboličkih linkova.
BITNO: Ove dve funkcije ne menjaju trenutni radni direktorijum – može se iskoristiti u zadacima.

4.22 Radni direktorijum

Svaki proces ima radni direktorijum. Ovo je direktorijum od koga kreće pretraga za svim relativnim putanjama fajlova. Kada se korisnik uloguje na sistem r.d. se postavlja na 6. polje u fajlu /etc/passwd, tj. na home direktorijum. Pomoću funkcija chdir i fchdir mozemo promeniti r.d. procesa.

Funkcija **getcwd** počinje sa izvršavanjem u r.d. i kreće se uz hijerarhiju koristeći tačka-tačka da se popne za jedan nivo. Ova funkcija potom čita zapise u direktorijumu dok ne nađe ime koje odgovara i-nodu iz koga je upravo došla. Ponavljanjem ove procedure dok se ne stigne do / dobija se apsolutna putanja fajla.

(PRIMER chdir)
Primer ilustruje da proces ne utiče na r.d. roditeljskog procesa.

```
pwd
./chdir
pwd
```

Da bi se promenio r.d. u shellu koristi se komanda **cd**. Ne može se napisati poseban program jer neće biti promenjena stvarna vrednost u shellu. Shell implementira neke programe u sebi. Prvo traži da li komandu može sam da odradi, pa ako ne, onda traži komandu u PATH-u.

(PRIMER types_counter_chdir) Malo izmenjen primer types_counter tako da koristi chdir funkciju.

Zadaci:
1. Napisati program *utime.c* koji briše sadržaj fajlova čija su imena navedena u komandnoj liniji, a zatim vraća vremena zadnjeg pristupa i modifikacije na one vrednosti koje su bile postavljene pre brisanja. Uputstvo: pokupiti vremena pomoću stat funkcije, obrisati **sadržaj** fajla *open* funkcijom, postaviti stara vremena pomoću utime funkcije. BITNO: koristiti funkciju ***utime*** a ne ***utimes***.

2. Napisati program koji pronalazi sve fajlove sa imenom *Makefile* u direktorijumu datim prvim argumentom komandne linije. Pretraga treba da se vrši i u svim njegovim poddirektorijumima. Štampati putanju svakog takvog fajla i njegovu veličinu u bajtovima. Nije dozvoljeno korišćenje funkcija *ftw* i *nftw*. Ne postoji pretpostavka o dužini putanje fajla!

3. Napisati modifikaciju prethodnog programa koji umesto nadovezivanja putanje koristi chdir funciju. Pri ispisu putanje može se koristiti funkcija *getcwd* za dobijanje lokacije gde se trenutni fajl nalazi.

4. Prethodni zadatak uraditi koristeći funkciju *nftw*.

5. Napisati funkciju koja štampa tekući direktorijum procesa koristeći *opendir* i *readdir*.