Fajlovi i direktorijumi

stat, fstat i lstat funkcije

*int stat(const char *path, struct stat *buf);*

*int fstat(int fd, struct stat *buf);*

*int lstat(const char *path, struct stat *buf);*

Ove 3 funkcije smeštaju informacije o navedenom fajlu u strukturu na koju pokazuje *buf*. Funkcija fstat kao prvi argument prima otvoreni fajl deskriptor dok ostale 2 funkcije primaju putanju do fajla. Kad god se pokuša neka operacija nad simboličkim linkom, izvršava se operacija nad fajlom na koji link pokazuje. Ukoliko hoćemo da radimo sa samim linkom, onda obično postoji varijacija sistemskog poziva koja počinje slovom 'l'. Funkcija lstat se razlikuje od stat jer kada je u pitanju simbolički link, lstat *vraća informaciju o samom linku a ne o fajlu na koji pokazuje link*. Komanda **ls -l** koristi stat funkciju za davanje informacija o fajlu.

Tipovi fajlova

Postoji 7 tipova fajlova u UNIXu.

1. Obični (regularni) fajlovi, najčešći tip fajlova, sadrže podatke u nekoj formi. Kernel ne razlikuje da li su ovi podaci tekstualni ili binarni (izuzev kada se radi o izvršnim fajlovima).
2. Direktorijumi – mogu se smatrati nekom vrstom tabela. Direktorijum sadrži imena drugih fajlova, kao i pokazivače na informacije o ovim fajlovima (pokazivač na i-nod).

U Unix-u postoje dve vrste uređaja, jedni sa kojima se mora raditi sekvencijalno, i drugi sa kojima se može kretati i unapred i unazad. Svi uređaji se predstavljaju jednim od dva naredna tipa fajlova:

3. Blok specijalni fajl – tip fajla koji obezbeđuje baferisani U/I u blokovima fiksne dužine prema uređajima kao što su hard disk, cd-rom, memorija.
4. Karakterski specijalni fajl – obezbeđuje nebaferisani U/I prema uređajima. - radi se sekvencijalno sa njima. Prenos jedan po jedan karakter (miš, tastatura, terminal, modemi).

Poslednja 3 tipa fajlova su:

5. FIFO. Koristi se za komunikaciju među procesima. Nekada se naziva pipe (cev).
6. Soketi. Koristi se za mrežnu komunikaciju među procesima, a može se koristiti i za komunikaciju na istom hostu.
7. Simbolički linkovi. Tip fajla koji pokazuje na druge fajlove.

(PRIMER – types)

./types ..

./types /dev/tty – tekući terminal

./types /dev/sda – prvi hard disk

./types /dev/initctl – kontrolni kanal za komunikaciju sa init procesom

Funkcije koje koristimo su u stvari makroi. Makro S_ISDIR je definisan sa

#define S_ISDIR(mode) (((mode) & S_IFMT) == S_IFDIR)

Set-user-ID i set-gropu-ID

Svaki fajl ima svog vlasnika, korisnika (user) koji je kreirao taj fajl. Svaki korisnik može pripadati jednoj ili više grupa na sistemu. Kada se kreira fajl, nekoj podrazumevanoj grupi korisnika koji ga kreira se

dodeljuju vlasništvo nad fajlom (korisnik može promeniti grupu koja je vlasnik pomoću naredbe chgrp). Dakle, svaki fajl ima dva vlasnika: korisnika i grupu (mogu se videti pomoću komande ls -l). Vlasništvo grupe omogućava da svi članovi te grupe imaju veća prava nad tim fajlom od ostalih korisnika na sistemu (npr. pored prava da čitaju imaju pravo i da pišu po tom fajlu).

Primer promene grupe: chgrp adm types.c (promena grupe koja je vlasnik fajla types.c – korisnik koji je pokrenuo komandu mora biti vlasnik fajla types.c i pripadati grupi adm).

Svaki proces ima 6 ili više ID-ieva povezanih sa njim.

Real user ID i **real group ID** identifikuju ko smo (u kompaniji bi odgovarali karticama za identifikaciju zaposlenih). Ova dva polja se preuzimaju iz fajla /etc/passwd kada se logujemo na sistem. **Effective user ID**, **effective group ID**, i **supplementary group IDs** određuju naša prava pristupa (u kompaniji bi odgovarali ključevima koje zaposleni imaju). **Saved set-user-ID** i **saved set-group-ID** sadrže kopije effective user IDa i effective group IDa kada se program izvršava.

Kao što je rečeno, svaki fajl ima vlasnika i grupu vlasnika. Oni su određeni poljima **st_uid** I **st_gid** u strukturi **stat.** Kada izvršavamo program, uobičajeno je da je effective user ID jednak real user IDu, a effective group ID je jednak real group IDu. Ali postoji mogućnost da se postavi specijalni fleg u **st_mode** koji označava da se pri obrađivanju fajla postavi effective user ID procesa da bude vlasnik fajla (st_uid). Slično, drugi bit se može promeniti tako da effective group ID bude vlasnik fajla. Ova dva bita se zovu **set-user-ID** bit i **set-group-ID** bit.

Npr. ako je vlasnik fajla superuser (root) a set-user-ID bit je postavljen, onda tokom izvršavanja programa proces ima superuser privilegije. Ovo ne zavisi od real-user-IDa procesa koji obrađuje fajl. Primer: funkcija **passwd.**

ls -l /usr/bin/passwd	- passwd je komanda za promenu lozinke na sistemu, pri izvršavanju se dobijaju superuser privilegije (slovo s u pravima je oznaka za ovo)
ls -l /etc/passwd	- fajl kome pristupa prethodna komanda

Prava pristupa

ls -l

Prva kolona je tip fajla, 'd' za direktorijum. Polje **st_mode** sadrži i bitove koji određuju prava pristupa za fajl. Postoji 9 bitova za prava pristupa, podeljenih u 3 kategorije.

Iz komandne linije možemo promeniti prava pristupa. Za to se koristi komanda **chmod.** Na primer

chmod g+w-r fajl

postavlja pravo pisanja a briše pravo čitanja za grupu, dok komanda

chmod og+w fajl

postavlja pravo pisanja za ostale i grupu.

Pravila:

1) Kada hoćemo da otvorimo bilo koji fajl po imenu, moramo imati execute privilegije u svim direktorijumima koji se pominju u putanji. Npr

poincare.matf.bg.ac.rs/~djenic/2016/sistemi/1.html

/usr/include/stdio.h

Ukoliko se već nalazimo u direktorijumu /usr/include, onda su nam potrebne samo execute privilegije u tom direktorijumu. Read i execute privilegije u direktorijumu znače različite stvari: read označava da možemo da vidimo imena fajlova koji se nalaze u direktorijumu, a execute da imamo dozvolu da prođemo kroz direktorijum da bismo došli do fajla koji nam treba. Pravo pisanja se odnosi na kreiranje i brisanje fajlova u okviru direktorijuma.

2) read privilegije u fajlu odlučuju da li možemo da pristupimo postojećem fajlu za čitanje: kada pomoću open funkcije pokušamo da otvorimo fajl pomoću O_RDONLY ili O_RDWR flega, ukoliko nemamo pravo čitanja biće prijavljena greška

3) write privilegije u fajlu odlučuju da li možemo da pristupimo postojećem fajlu za pisanje: O_WRONLY i O_RDWR flegovi u open funkciji.

4) ne možemo da kreiramo novi fajl u direktorijumu ukoliko nemamo write i execute privilegije u tom direktorijumu

5) da bi obrisali fajl potrebne su nam write i execute privilegije u direktorijumu u kome se fajl nalazi. Nisu nam potrebne privilegije za sam fajl.

Kernel sekvencijalno radi 4 provere kada proces pokušava da radi sa fajlom:

1. Ukoliko je effective user ID procesa 0, pristup je dozvoljen.
2. Ukoliko je effective user ID procesa jednak owner IDu fajla, pristup je dozvoljen ukoliko je odgovarajući bit pristupa postavljen (user-read, user-write ili user-execute).
3. Ukoliko je effective group ID procesa ili jedan od supplementary group IDs procesa jednak group IDu fajla, pristup je dozvoljen ukoliko je odgovarajuci bit pristupa postavljen.
4. Ukoliko je odgovarajuci bit za *other* postavljen pristup je dozvoljen. Inače, pristup nije dozvoljen

Napomena: Drugi operativni sistemi funkcionišu drugačije. Npr, postoji access control list, svakom fajlu pridružena jedna lista koja govori koji korisnik, odnosno grupa korisnika može da radi sa tim fajlom. Lošije jer zauzima više prostora i teže je menjati prava. Bolje jer mogu detaljnije da se zadaju prava.

Postoje 3 specijalna prava pristupa: S_ISUID (set-user-ID), S_ISGID (set-group-ID) i sticky pravo pristupa S_ISVTX (ukupno 9+3=12). Ova prava su postavljena na vrlo malom broju fajlova. Ukoliko se izvrši komande

chmod +s types

ls -l

onda slovo s označava da je postavljen jedan od ovih bitova kao i bit za execute. Slovo t označava da je postavljen sticky bit. Ako umesto s (t) stoji S (T) onda execute bit nije postavljen.

Funkcija umask

Funkcija **umask** postavlja masku za prava pristupa pri kreiranju fajla, i vraća prethodnu masku. Ovo je jedna od retkih funkcija koje nemaju kao mogućnost vraćanje greške. Maska se koristi kad god proces kreira novi fajl ili direktorijum. Svi bitovi koji su postavljeni u maski, **isključeni su pri kreiranju fajla.**

(PRIMER – umask)

Kada želimo da obezbedimo da su pojedini bitovi za pristup postavljeni moramo modifikovati umask vrednost dok se proces izvršava. Menjanje maske u procesu ne utiče na masku u shell-u. Vrednost umask se reprezentuje kao oktalni broj.

Za razliku od ispisa prava pristupa kao kod ls -l komande, kod **oktalnog zapisa** prava pristupa su predstavljena oktalnom trojkom. Slovo r nosi vrednost 4, w nosi vrednost 2, x nosi vrednost 1. Inicijalna prava pristupa određuje open, kada je navedena opcija O_CREAT.

Neke od komandi koje se mogu uneti u shell-u su:

umask - prikazuje koja su prava onemogućena pri kreiranju fajlova

umask -S - prikazuje koja su prava omogućena

umask 027 - postavlja masku na 027

chmod i fchmod funkcije

*int chmod(const char *path, mode_t mode);*

int fchmod(int fd, mode_t mode);

Omogućavaju nam da promenimo bitove pristupa postojećeg fajla. Da bi se promenili bitovi pristupa, potrebno je da effective-user-ID procesa bude jednak IDu vlasnika fajla, ili da proces ima superuser privilegije.

Prednost UNIXa je da možemo u jednoj for petlji da prođemo kroz ceo sistem i promenimo prava pristupa.

Napomena: **umask vrednost ne utiče na promenu prava pristupa fajlu, već utiče na prava samo pri kreiranju fajla!**

Sticky bit

Ranije imao drukčiju namenu, danas se koristi za direktorijume. Direktorijum

/tmp

ima postavljen sticky bit, sto označava da korisnici mogu da menjaju svoje fajlove, ali ne mogu da brišu ili menjaju fajlove koji pripadaju drugim korisnicima u okviru ovog direktorijuma.

chown, fchown i lchown funkcije

Ove funkcije omogućavaju da se promeni user ID i group ID fajla. Najčešće je **samo root** korisniku dozvoljeno da promeni vlasnika. U suprotnom, zlonameran korisnik može da da fajl drugom korisniku, postavi ga negde gde ovaj neće moci da ga nađe, i radi sa njim. Kod drugog korisnika može da dođe po punjenja prostora. Obični korisnici mogu da postave za vlasnika njihovu grupu.

Veličina fajla

Polje **st_size** u strukturi stat sadrži veličinu fajla u bajtovima. Ovo polje ima smisla samo kod regularnih fajlova, direktorijuma i simboličkih linkova. Kod običnih fajlova dozvoljena je veličina 0; kod simboličkih linkova veličina fajla je broj bajtova u imenu. U strukturi stat postoje i polja **st_blksize** i **st_blocks**. Prvo predstavlja preferiranu veličinu bloka za I/O operacije na fajlu, dok drugo predstavlja broj 512-bajt blokova koji su alocirani.

I-nod sadrži podatke o fajlu. Sadržaj fajla je **rasut po blokovima**. Blokovi se alociraju gde ima prostora, a i-nod sadrži listu blokova. Kada koristimo read, kernel vidi koliki je trenutno ofset, podeli to sa veličinom blokova, i vidi u kom smo bloku, iz i-noda čita gde je ta pozicija na disku pa odatle čita podatke.

Zadaci

1. Napisati program *file_info.c* koji za sve fajlove čija se imena navode kao argumenti komandne linije ispisuje tip fajla, veličinu u bajtovima, ID vlasnika fajla i ime fajla. Ispis treba da bude što sličniji ispisu komande ls. Za simboličke linkove ispisati podatke o fajlu na koji pokazuje simbolički link.
2. Modifikovati prethodno napisani program (*file_info.c*) tako da se ispisuju i prava pristupa. Ispitivanje svakog od 9 prava uraditi pomoću komandi oblika *if (stat.st_mode & S_IRUSR).* Veličinu fajla X ispisati u najpogodnijoj meri, tj. Ako je veličina:
    1. X < 1024 u bajtovima
    2. 1024 <= X < 1024*1024 u kilobajtima
    3. X > 1024*1024 u megabajtima
3. Napisati program *postavi_prava.c* koji kao argumente komandne linije prima ime fajla i nova prava pristupa za taj fajl. Prava pristupa su predstavljena celim brojem koji označava oktalnu konstantu bez vodeće nule (npr. 644, 711, itd). Program postavlja navedena prava pristupa fajlu (funkcijom chmod), a ako fajl ne postoji kreira ga sa navedenim pravima pristupa (open funkcijom). Mogući način testiranja da li fajl postoji: izvršiti stat funkciju na njemu i u slučaju da je došlo do greške proveriti da li je razlog greške nepostojanje fajla (test je oblika *errno==konstanta,* gde konstanta počinje sa 'E' i potrebno je pronaći njeno ime u man stranici stat funkcije). Pri kreiranju fajla navesti i opciju O_EXCL. Ovom opcijom se omogućava da se otkrije situacija kada je između stat i open neki drugi korisnik kreirao fajl sa istim imenom i u tom slučaju prijavi greška.