Borislav Đorđević
Dragan Pleskonjić
Nemanja Maček

Operativni sistemi: UNIX i Linux

Viša elektrotehnička škola
Beograd, 2004.

Autori:	dr Borislav Đorđević mr Dragan Pleskonjić Nemanja Maček
Recenzenti:	prof. dr Borivoj Lazić mr Slobodan Obradović
Izdavač:	Viša elektrotehnička škola u Beogradu
Za izdavača:	mr Dragoljub Martinović
Lektor:	Anđelka Kovačević
Tehnička obrada:	Nemanja Maček, Borislav Đorđević, Dragan Pleskonjić
Dizajn:	Nemanja Maček
Štampa:	*ime štamparije* štampano u 400 primeraka

Predgovor

UNIX je stabilan operativni sistem visokih performansi, pogodan za izvršavanje velikog broja različitih aplikacija. Kao takav, dostupan je već trideset godina za brojne računarske arhitekture. Većina velikih svetskih proizvođača računara razvija sopstvenu komercijalnu varijantu UNIX operativnog sistema, poput SCO, HP-UX, IBM AIX i Sun Solaris. Alternativa kvalitetnim, ali relativno skupim UNIX operativnim sistemima je Linux, koji zadržava većinu dobrih osobina UNIX sistema, a dodatno se odlikuje raspoloživim izvornim kodom i praktično besplatnim korišćenjem. Zahvaljujući intuitivnom, user-friendly grafičkom okruženju, Linux se može koristiti kao operativni sistem na radnim stanicama. Takođe, može se koristiti i kao operativni sistem na serverima u LAN i WAN mrežama, gde mu može biti dodeljena uloga servera za datoteke, web i mail servera, mrežne skretnice (rutera) ili mrežne barijere (firewall uređaja)

Knjiga se prvenstveno bavi korišćenjem i administracijom Linux operativnog sistema i namenjena je osnovnom kursu iz predmeta Operativni sistemi na Višoj elektrotehničkoj školi u Beogradu. Globalno, knjiga se može podeliti na četiri tematske celine koje se obrađuju u okviru predmeta: rad sa sistemima datoteka, kontrola pristupa na nivou sistema datoteka, rad sa datotekama i administracija UNIX sistema. Takođe, knjiga može poslužiti kao koristan izvor informacija svakom čitaocu koji se ozbiljnije bavi administracijom UNIX ili Linux operativnog sistema. Knjiga je podeljena u četrnaest glava, koje su ukratko opisane u nastavku teksta.

U prvoj glavi ”UVOD U UNIX OPERATIVNI SISTEM” opisan je kraći istorijat UNIX i Linux operativnih sistema, razjašnjeni su pojmovi GNU GPL licence i Open Source softvera, dat je opšti pregled Linux sistema, značajnih delova jezgra i osnovnih servisa Linux sistema.

U drugoj glavi ”BLOK UREĐAJI I ADMINISTRACIJA SISTEMA DATOTEKA” prikazani su osnovni postupci administracije blok uređaja (diskova, disketa, CD i DVD uređaja i magnetnih traka) i različitih sistema datoteka. Objašnjeni su postupci podele diskova na particije, kreiranja i aktiviranja sistema datoteka i formiranja aktivnog UNIX stabla, administracije swap prostora i oporavka oštećenih sistema datoteka. Glava takođe obuhvata kratak pregled savremenih journaling sistema datoteka kao što su ext3, ReiserFS, JFS i XFS i objašnjenja značajnih sistemskih direktorijuma u aktivnom UNIX stablu.

U trećoj glavi ”KORISNICI I GRUPE” prikazani su osnovni postupci administracije korisničkih naloga i grupa i objašnjeni formati fundamentalnih datoteka /etc/passwd, /etc/shadow i /etc/group. Nakon toga su objašnjeni pojmovi stvarnog i efektivnog identifikatora korisnika i procedura privremenog prijavljivanja na sistem pod drugim imenom.

U četvrtoj glavi ”KONTROLA PRISTUPA NA NIVOU SISTEMA DATOTEKA” najpre se uvode pojmovi vlasničkih odnosa i pristupnih prava za datoteke i direktorijume. U ovoj glavi su prikazani postupci promene pristupnih prava i vlasničkih odnosa, postavljanja specijalnih atributa SUID, SGID i sticky bit, kao i specijalnih atributa datoteka na ext2/ext3 sistemima datoteka. Takođe, ilustrovan je i jednostavan postupak postavljanja kvota na sistemima datoteka.

U petoj glavi ”RAD SA DATOTEKAMA IZ KOMANDNE LINIJE” prikazane su razne komande za rad sa datotekama i direktorijumima. Glava obuhvata više celina. Na početku glave opisane su osnovne funkncije komandnog interpretera i opšte komande za rad sa datotekama. Nakon toga, detaljno su opisane komande za kopiranje, pomeranje i brisanje datoteka, odnosi između originala i kopije, kao i uslovi neophodni za izvršenje odgovarajuće komande. U okviru ove celine detaljno su objašnjeni pojmovi hard i simboličkih linkova. Sledeću celinu obuhvataju komande za rad sa direktorijimima, sa posebnim osvrtom na komandu za pretraživanje direktorijuma, find. Na kraju glave dat je opis značajnih komandi za rad sa tekstualnim datotekama, uključujući i najpoznatije tekst editore.

U šestoj glavi ”SHELL PROGRAMIRANJE” najpre su dati primeri jednostavnih shell programa i objašnjenje postupka pokretanja shell programa. Nakon toga su date definicije sistemskih i korisničkih promenljivih, opis komandi specifičnih za shell programiranje i raznih shell proširenja. Zatim su obrađene konstrukcije u shell programiranju, kao što su uslovne konstrukcije, petlje i funkcije. Na kraju glave su dati primeri složenijih shell programa.

U sedmoj glavi ”MREŽNO OKRUŽENJE” dat je najpre kraći uvod u TCP/IP skup protokola i lokalne računarske mreže. Nakon toga objašnjeno je konfigurisanje Linux mrežnog okruženja, koje obuhvata konfiguracione datoteke i programe za administraciju TCP/IP skupa protokola i mrežnih servisa. Na kraju glave opisani su mrežni sistem datoteka (NFS), mehanizam centralizovane autentifikacije (NIS) i Apache web server čime je Linux prikazan kao mrežni server u lokalnoj računarskoj mreži i na Internetu.

U osmoj glavi ”ŠTAMPAČI” prikazan je najpre proces štampanja pod UNIX sistemom, dat je opis štampača, redova čekanja za štampu i print servera. U okviru ove glave detaljno je obrađen CUPS (Common UNIX Printing System), objašnjene su komande koje se koriste za štampanje, administraciju reda za štampu i podešavanje karakteristika štampača i dokumenata iz komandne linije. Na kraju glave opisani su osnovni postupci administracije CUPS sistema, koji obuhvataju instalaciju paketa, upravljanje štampačima, klasama štampača i redovima za štampu.

U devetoj glavi ”ARHIVIRANJE I BACKUP” uvode se pojmovi arhive i rezervne kopije podataka, nakon čega se navode razlike između arhiva i sistema datoteka. Nakon toga su opisani Linux programi za backup i arhiviranje, tar i cpio, sa pratećim primerima. Na kraju glave ukratko su opisani programi dump i restore za integralno arhiviranje čitavih sistema datoteka.

U desetoj glavi ”ADMINISTRACIJA PROCESA” objašnjen je pojam i date su osnovne vrste procesa, nakon čega su opisane komande za prikazivanje procesa i slanje signala procesima. Zatim su objašnjeni prioriteti procesa, izvršavanje procesa u pozadini, grupe procesa i kontrola posla. Na kraju glave opisani su postupci zakazivanja i periodičnog izvršavanja komandi programima at i cron.

U jedanestoj glavi ”PODIZANJE I ZAUSTAVLJANJE SISTEMA” prikazane su procedure podizanja (boot) i zaustavljanja sistema (shutdown). U ovoj glavi su opisani proces init, različiti nivoi izvršenja UNIX sistema, inicijalizacione rc datoteke, procesi getty, login i shell i komanda za zaustavljanje sistema shutdown.

U dvanestoj glavi ”INSTALACIJA SOFTVERSKIH PAKETA” najpre je opisan postupak instalacije softvera, a zatim osnovne vrste softverskih paketa, kao što su tarball, RPM i deb paketi. Nakon toga je dato kraće uputstvo za korišćenje poznatih paket menadžera: Red Hat Package Manager i Debian package management system.

U trinaestoj glavi ”KONFIGURISANJE LINUX KERNELA” najpre je dat opis programskih modula jezgra, a zatim su objašnjeni postupci dodavanja i uklanjanja modula iz aktivnog jezgra. Nakon toga se diskutuje o problemima koji se ne mogu rešiti modulima. Na kraju glave detaljno je opisana procedura prevođenja kernela.

U četrnaestoj glavi ”SIGURNOST I ZAŠTITA UNIX I LINUX SISTEMA” dat je kraći pregled zaštite UNIX i Linux sistema. U okviru ove glave obrađeni su neki standardni mehanizmi zaštite UNIX sistema i opšte sigurnosti Linux sistema, paket za šifrovanje i potpisivanje GNU Privacy Guard i iptables mrežna barijera.

U dodatku A dat je spisak značajnijih komandi Linux sistema.

U dodatku B naveden je prevod GNU opšte javne licence na srpski jezik.

U dodatku C naveden je plan i program laboratorijskih vežbi iz predmeta Operativni sistemi, koji se sluša na Višoj elektrotehničkoj školi. U okviru ovog dodatka data su pitanja koja čitaoci mogu iskoristiti za proveru stečenog znanja.

Zahvalnost

Zahvaljujemo se svima koji su učestvovali ili na bilo koji način pomogli u pripremi i realizaciji ove knjige. Posebno se zahvaljujemo:

- Maček Draganu, koji je detaljno pregledao rukopis i ukazao na greške,
- Nikolić Ivanu (Joe), na savetima na osnovu kojih su neka poglavlja dobila svoj konačan oblik,
- saradnicima Gavrilović Predragu, Krneta Borislavu i Krstanović Mladenu, na svim korisnim savetima i sugestijama,
- Kurtić Nikoli, Dostanić Miroslavu, Kukobat Gojku, Pavlović Draganu, Samardžić Saši i Bogojević Čedomiru, studentima Više elektrotehničke škole, koji su aktivnim učestvovanjem pomogli u pripremi materijala za ovu knjigu.

Autori

Sadržaj

1

UVOD U UNIX I LINUX OPERATIVNE SISTEME

UNIX je stabilan, moćan i fleksibilan operativni sistem visokih performansi pogodan za izvršavanje kritičnih aplikacija od visoke važnosti. UNIX je čvrsto povezan sa mrežnim servisima TCP/IP protokola, čime je u potpunosti promenjena slika UNIX servera i radnog okruženja iz prošlosti. Umesto servera sa klasičnim serijskim terminalima UNIX server se nalazi u mreži, pri čemu sa radnim stanicama ostvaruje vezu preko LAN/WAN mreže i TCP/IP skupa protokola. Većina velikih svetskih proizvođača računara razvija specifičnu varijantu UNIX operativnog sistema, što ukazuje na njegov kvalitet, popularnost i rasprostranjenost. Većina UNIX sistema, poput IBM AIX i Sun Solaris, je komercijalna - korisnik mora da plati licencu za korišćenje, a izvorni kod nije raspoloživ. To su razlozi narastajuće popularnosti Linux sistema koji zadržava većinu dobrih osobina UNIX sistema, a dodatno se odlikuje raspoloživim izvornim kodom i praktično besplatnim korišćenjem. Zbog toga danas većina proizvođača računara osim sopstvene komercijalne verzije UNIX sistema nudi i podršku za Linux. Linux se najčešće koristi kao operativni sistem na radnim stanicama ili serverima u manjoj ili srednjoj klasi servera, a jedna od oblasti dominantne primene, u kojoj veliki broj korisnika podržava i promoviše Linux kao bazični server, su Internet servisi.

Istorijat UNIX operativnog sistema

Istorijat i razvoj UNIX operativnog sistema. Vrste UNIX sistema. Linux distribucije, GNU GPL licenca i Open Source softver.

Razvoj UNIX operativnog sistema počeo je sredinom 1960-tih godina u AT&T Bell laboratorijama u saradnji sa kompanijom General Electric i tehnološkim institutom Massachusetts. Projekat, odnosno operativni sistem MULTICS (Multiplexed Information and Computing Service) predstavljao je interaktivni operativni sistem namenjen da opslužuje veliki broj korisnika čiji su terminali direktnim serijskim ili modemskim komunikacionim kanalima povezani na centralizovani server. Takav koncept operativnog

sistema bio je preambiciozan za tadašnji stepen razvijenosti hardvera, pri čemu se prvenstveno misli na procesorsku snagu i količinu sistemske memorije. MULTICS nije doživeo svoju praktičnu primenu jer se posle nekoliko godina razvoja pokazao kao preskup i preambiciozan projekat od koga su AT&T Bell laboratorije odustale. Bez obzira na to, teorijska i praktična rešenja projekta našla su primenu u mnogim operativnim sistemima. Konkretno, MULTICS je preteča UNIX sistema koji se smatra jednim od najkvalitetnijih i najrasprostranjenijih operativnih sistema, na čijem se razvoju radi preko 30 godina, sa tendencijom dalje egzistencije i usavršavanja.

Dva fundamentalna imena vezana za razvoj UNIX operativnog sistema su svakako Ken Thompson, MULTICS sistemski programer u Bell laboratorijama, i Dennis Ritchie, poznatiji kao tvorac programskog jezika C. Godine 1969, Ken Thompson je započeo razvoj novog operativnog sistema za DEC PDP-7 računar, napravivši redukovani MULTICS, odnosno UNICS (Uniplexed Information and Computing Service). Radi lakšeg izgovora i pisanja ime UNICS je kasnije evoluiralo u UNIX. UNIX je prvobitno napisan u asemblerskom jeziku, a samim tim bio je potpuno zavisan od klase procesora za koji se realizuje. Godine 1971, Ritchie je napisao programski jezik C kao viši programski jezik koji omogućava sistemsko programiranje, a zatim sa Thompsonom preveo kôd UNIX sistema na C, što se može smatrati prekretnicom i jednim od najznačajnijih poteza u razvoju operativnih sistema. Zahvaljujući C jeziku UNIX je mogao biti prenešen na razne računarske arhitekture sa vrlo malo programskih modifikacija, što je svakako bio ključ uspeha i popularnosti UNIX operativnog sistema.

Nakon prevoda na C jezik, autori su u cilju daljeg unapređenja prosledili izvorni kôd UNIX sistema univerzitetima širom Amerike, pri čemu su programeri sa Berkeley univerziteta u Kaliforniji dominantno uticali na dalji razvoj. Nakon brojnih modifikacija koda nastao je BSD UNIX (Berkeley Software Distribution), koji je takođe u izvornoj formi distribuiran američkim univerzitetima na dalje usavršavanje. Najznačajniji doprinos grupe koja je realizovala BSD UNIX odnosi se na softver za umrežavanje, koji omogućava funkcionalnost operativnog sistema u LAN i WAN mrežama. BSD UNIX mrežna rešenja prihvatili su brojni proizvođači računara koji su razvijali sopstvene varijante UNIX operativnog sistema:

- SunOS, kompanije Sun Microsystems, baziran na BSD UNIX v4.2
- System V UNIX, kompanije AT&T
- XENIX, kompanije Microsoft, razvijen za PC računare sa Intel procesorima.

Godine 1988, kombinovanjem dobrih osobina Sun OS/BSD, AT&T System V Release 3 i XENIX operativnih sistema nastala je nova varijanta UNIX sistema - System V Release 4 (SVR4), koja je ubrzo postala de facto standard i osnova za dalji razvoj UNIX sistema.

Sledeći korak u razvoju predstavlja pokušaj standardizacije i realizacije međusobne kompatibilnosti raznih vrsta UNIX sistema.

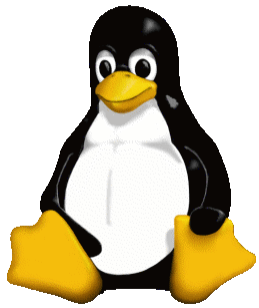
Vrste UNIX sistema

Danas postoji veliki broj različitih vrsta UNIX sistema koje su zasnovane ili na industrijskom standardu SVR4 ili na BSD distribuciji. One vrste UNIX sistema koje

potiču od istog standarda međusobno su veoma slične, tako da se na osnovu nekih indikatora (najčešće na osnovu sistemskih komandi) može odrediti koji je standard korišćen kao osnova za izgradnju i razvoj specifičnog UNIX sistema. Komanda za štampanje na sistemima čija je osnova System V je lp, dok je komanda za štampanje na BSD baziranim sistemima lpr. Komanda koja opisuje tekuće procese na System V sistemima je ps -ef , dok BSD koristi ps -aux. Razlike postoje i u drugim komandama.

UNIX je sada zaštićeno ime u vlasništvu organizacije The Open Group (www.opengroup.org), koja je definisala POSIX standard (Portable Operating System Interface) u cilju povećanja kompatibilnosti raznih vrsta UNIX sistema. Svaki proizvođač svoju UNIX distribuciju naziva određenim unikatnim imenom: Solaris (Sun Microsystems), AT&T UNIX, AIX (IBM), HP-UX (Hewlett-Packard), Tru64UNIX (Compaq). Iako je većina UNIX sistema kreirana da radi uglavnom na određenoj arhitekturi sa određenom klasom procesora (najčešće su to procesori matične kompanije) postoje varijante koje rade na više klasa procesora. Na primer, Solaris koji je prvenstveno namenjen Sun računarima postoji i u verziji za PC arhitekturu, odnosno za Intel procesore, dok Linux, pored PC arhitekture podržava i druge arhitekture kao što su Sun, Apple Mackintosh i IBM. Kada se savlada rad na jednoj vrsti UNIX sistema prelazak na neku drugu verziju je relativno lak.

Linux operativni sistem



Jedna od poslednjih varijanti UNIX operativnih sistema, čiji je razvoj započeo Linus Torvalds 1991. godine na Univerzitetu u Helsinkiju, je Linux. Torvalds je svoj operativni sistem koji objedinjuje oba standarda, SRV4 i BSD, objavio na Internetu i podsticao druge programere širom sveta da se priključe njegovom daljem razvoju. Ubrzo, Linux je postao veoma popularan među računarskim entuzijastima koji su tražili alternativno rešenje za postojeće operativne sisteme za PC računare (DOS, Windows). Linux je svojom koncepcijom stabilnog a jeftinog operativnog sistema doživeo veliku ekspanziju i popularnost. Simbol Linux sistema je mali pingvin (Tux).

Linux je prvobitno namenjen 32-bitnim Intel x86 mikroprocesorima (počevši od 80386), na kojima može funkcionisati kao radna stanica (workstation) ili kao server. Jezgro Linux sistema je kasnije modifikovano i prilagođeno procesorima koji ne pripadaju Intel x86 klasi, među kojima treba istaći Intel IA-64, DEC Alpha, SUN SPARC/UltraSPARC, Motorola 68000, MIPS, PowerPC i IBM mainframe S/390. Može se konstatovati da današnji Linux u odnosu na bilo koji operativni sistem podržava najširi spektar procesora i računarskih arhitektura.

GNU/Linux i Open Source Software

Veliki deo komponenti Linux operativnom sistemu dodali su nezavisni programeri i programeri GNU projekta (www.gnu.org),

koji pripada slobodnoj softverskoj fondaciji (FSF - Free Software Foundation). Svi GNU/Linux operativni sistemi koriste Linux kernel kao fundamentalni deo koji kontroliše interakciju između hardvera i aplikacija, i GNU aplikacije kao dodatne komponente operativnog sistema.

Linux je raspoloživ kao besplatan operativni sistem pod GNU GPL licencom (GNU General Public Licence), što važi i za neke druge vrste UNIX sistema, kao što su FreeBSD i NetBSD. Linux je softver sa otvorenim izvornim kodom (Open Source), što znači da je izvorni kôd javno raspoloživ i može biti modifikovan tako da odgovara specifičnim potrebama. Linux se može slobodno distribuirati među korisnicima. Ovakav koncept je potpuno suprotan konceptu komercijalnog softvera, gde izvorni kôd nije dostupan i svaki korisnik mora da plati licencu za korišćenje. Komercijalni softver je baziran na autorskim pravima (copyright laws) koja preciziraju limite koje korisnici softvera imaju u odnosu na izvorni kôd, korišćenje i dalje distribuiranje softvera. Linux se besplatno može preuzeti sa različitih web-sajtova.

Linux distribucije

Brojne profitne i neprofitne organizacije čine Linux raspoloživim u formi distribucija, odnosno različitih kombinacija kernela, sistemskog softvera i korisničkih aplikacija. Većina distribucija sadrži kolekciju CD/DVD medijuma na kojima se nalaze operativni sistem, izvorni kôd, detaljna dokumentacija, kao i štampana uputstva za instalaciju i upotrebu sistema. Cene ovakvih distribucija su u većini slučajeva simbolične, osim ako se u distribuciji nalazi komercijalan softver ili je distribucija specifične namene.

Osnovna komponenta svake Linux distribucije je kernel operativnog sistema. Osim kernela i sistemskog softvera u distribuciji se nalaze i instalacioni alati, softver za podizanje operatvnog sistema (boot loader), razne korisničke aplikacije (kancelarijski paketi - office suite, softver za manipulaciju bit-mapiranih slika) i serverski paketi. Većina distribucija je, poput Windows sistema, grafički orijentisana prema korisniku, dok su neke distribucije namenjene za sistemske administratore i programere familijarne sa tradicionalnim UNIX okruženjem.

U poznatije Linux distribucije spadaju: Debian GNU/Linux (http://www.debian.org), Linux Mandrake (http://linux-mandrake.com/en), Red Hat Linux (http://www.redhat.com), Slackware Linux (http://www.slackware.com) i SuSE Linux (http://www.suse.com).

Opšti pregled Linux sistema

Delovi operativnog sistema. Značajni delovi kernela i osnovni servisi Linux sistema.

Linux je višekorisnički, višeprocesni operativni sistem sa potpunim skupom UNIX kompatibilnih alata, projektovan tako da poštuje relevantne POSIX standarde. Linux sistemi podržavaju tradicionalnu UNIX semantiku i potpuno implementiraju standardni UNIX mrežni model.

Linux operativni sistem sastoji se od kernela, sistemskog softvera, korisničkih aplikacija, programskih prevodilaca i njihovih odgovarajućih biblioteka (GCC - GNU C Compiler i C biblioteka za Linux) i dokumentacije. Sadržaj konkretne Linux distribucije definisan je sadržajem instalacionih medijuma, koji u slučaju nekih Linux sistema uključuju razne FTP sajtove širom sveta.

Kernel je jezgro operativnog sistema - on omogućava konkurentno izvršavanje procesa, dodeljuje im memoriju i druge resurse i obezbeđuje mehanizam za ostvarivanje usluga operativnog sistema. Kernel štiti korisničke procese od direktnog pristupa hardveru - procesi pristupaju hardveru korišćenjem sistemskih poziva kernela, čime se obezbeđuje jedna vrsta zaštite između samih korisnika. Sistemski programi koriste kernel u cilju implementacije različitih servisa operativnog sistema.

Svi programi, uključujući i sistemske, funkcionišu na nivou iznad kernela, što se naziva korisnički režim rada, dok se sistemske aktivnosti poput pristupa hardveru obavljaju na nivou kernela, odnosno u sistemskom režimu rada (supervisory mode). Razlika između sistemskih i aplikativnih programa je u njihovoj nameni: aplikacije su namenjene za razne korisne aktivnosti (kao što su obrada teksta i slike), dok su sistemski programi namenjeni za rad sa sistemom i administraciju. Na primer, tekst procesor je korisnička aplikacija, dok je komanda mount sistemski program. Razlike između korisničkih i sistemskih programa su ponekad veoma male i značajne samo za stroge kategorizacije softvera.

Linux kernel

Tri osnovne verzije Linux kernela su početna verzija, verzija 1.x i verzija 2.x. Početna verzija 0.01, koju je 1991. godine kreirao Linus Torvalds, podržavala je samo Intel 80386 kompatibilne procesore, mali broj hardverskih uređaja i Minix sistem datoteka. Mrežni servisi nisu imali kernelsku podršku. Verzija 1.0, nastala u martu 1994. godine, uključivala je podršku za standardne TCP/IP mrežne protokole, BSD-kompatibilni socket interfejs za mrežno programiranje i drajversku podršku za mrežne kartice. Ova verzija je dodatno podržavala ext i ext2 sisteme datoteka, široku klasu SCSI disk kontrolera, kao i brojne hardverske uređaje. Verzija 1.2 (mart 1995) je poslednja verzija Linux kernela namenjena isključivo PC arhitekturi. U verziji 2.0 (jun 1996) uvedena je podrška za više arhitektura (Motorola i Intel procesori, Sun Sparc i PowerMac sistemi), kao i podrška za višeprocesorsku arhitekturu (SMP). Dodatno, poboljšano je upravljanje memorijom i uvećane se performanse TCP/IP protokol steka, a ugrađena je i podrška za unutrašnje kernelske niti (internal kernel threads). Kernel je modularizovan, odnosno uvedena je mikro-kernel struktura sa izmenljivim drajverskim modulima (loadable kernel modules), a standardizovan je i konfiguracioni interfejs.

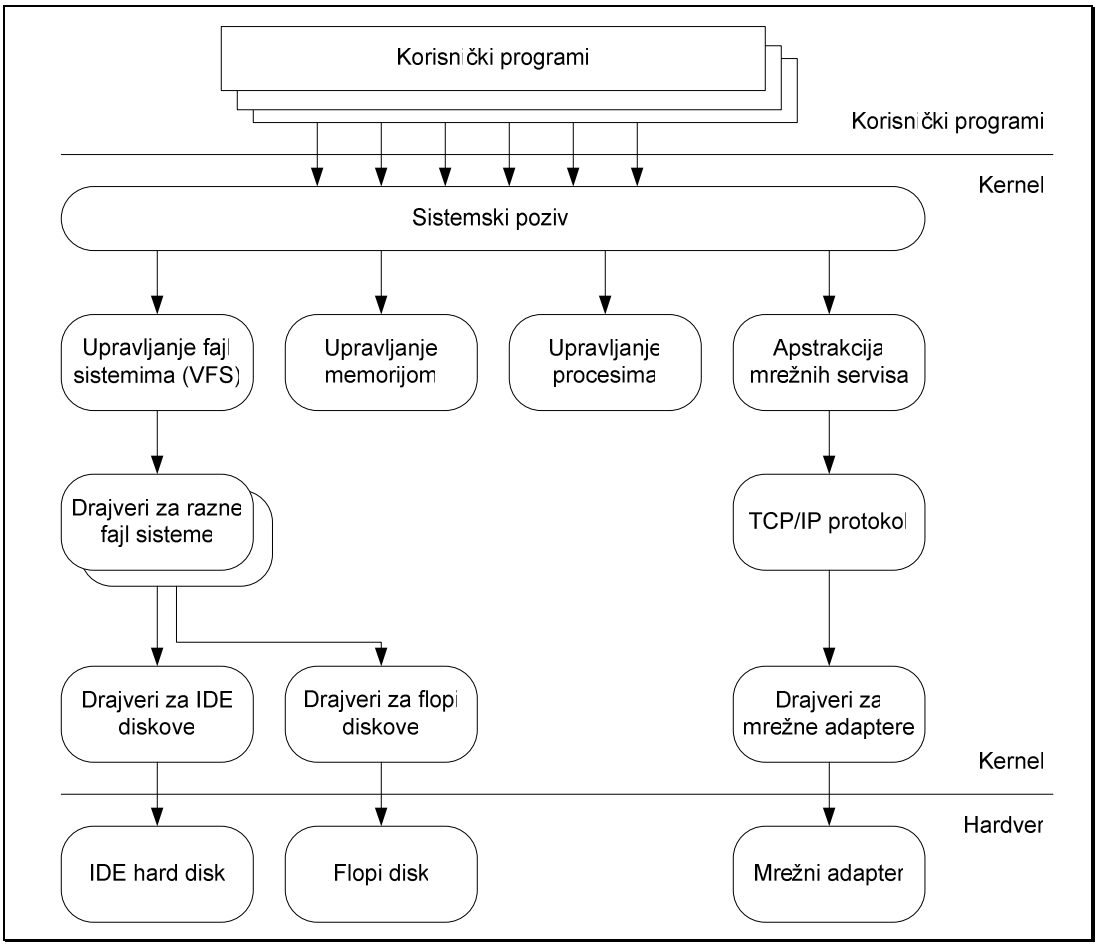
Struktura Linux sistema

Osnovu Linux sistema čine kernel, sistemske biblioteke i sistemski programi. Kernel je odgovoran za najznačajnije funkcije operativnog sistema. Dve osnovne karakteristike kernela su:

- kernel kôd se izvršava u kernelskom modu u kome je jedino moguće pristupati svim komponentama hardvera,

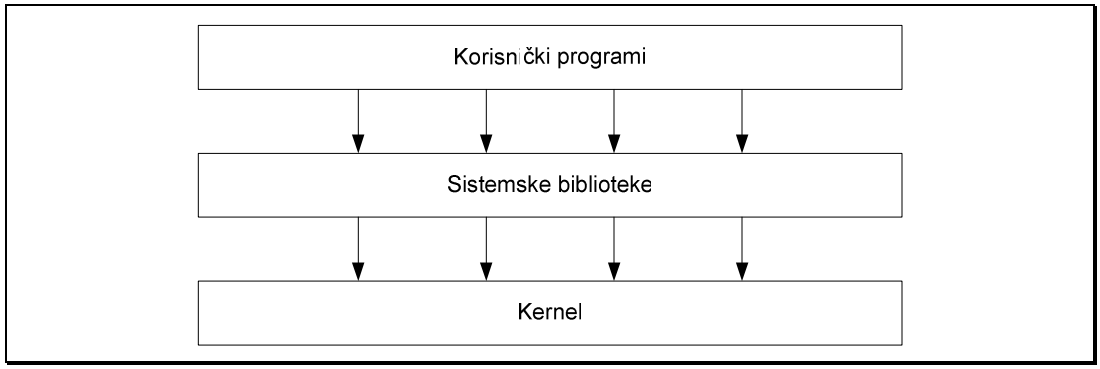
- kompletan kernel kôd i sve kernel strukture podataka čuvaju se u istom adresnom prostoru (monolithic).

Kod većine UNIX sistema aplikacije se preko sistemskog poziva direktno obraćaju kernelu, kao što je prikazano na slici 1.1.



Slika 1.1 *UNIX kernel*

Kod Linux sistema sistemski pozivi se upućuju kernelu preko sistemskih biblioteka koje definišu standardni set funkcija preko kojih aplikacije komuniciraju sa kernelom. Ovaj metod komunikacije sa kernelom prikazan je na slici 1.2.



Slika 1.2 *Struktura Linux sistema*

Sistemski programi izvršavaju specifične upravljačke poslove, kao što je konfigurisanje mrežnih uređaja i protokola, punjenje kernelskih modula itd.

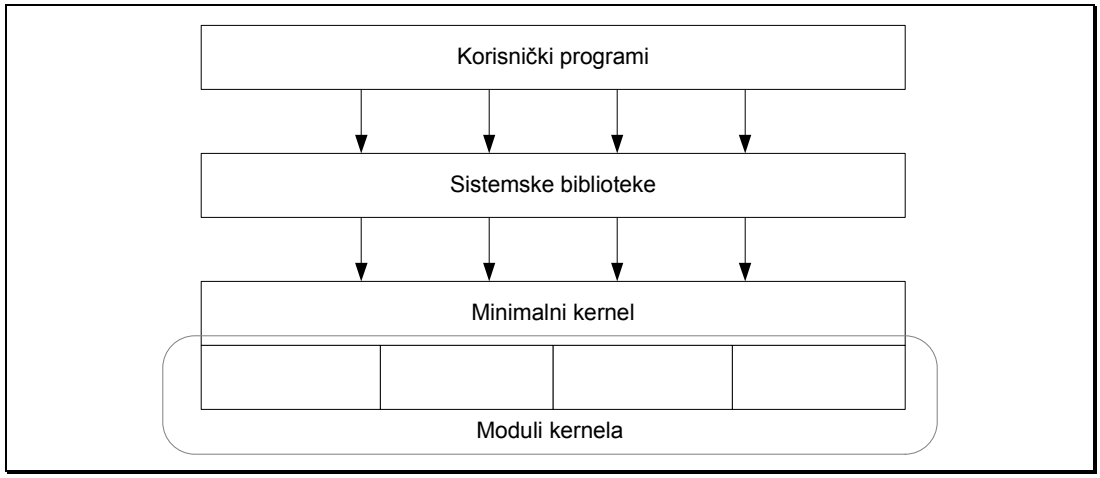
Modularni kernel

Moduli kernela su delovi kernelskog koda koji može da se prevede, napuni u memoriju ili izbaci iz memorije nezavisno od ostatka kernela. Kernelski moduli implementiraju drajvere za hardverske uređaje, novi sistem datoteka, mrežne protokole, itd. Moduli omogućavaju raznim programerima da napišu i distribuiraju drajvere koji ne moraju da prođu GPL licencu.

Moduli kernela omogućavaju micro-kernel arhitekturu, odnosno realizaciju minimalne stabilne konfiguracije kernela bez dodatnih drajvera. Potrebni drajveri pune se u memoriju kao moduli kernela. Module Linux kernela čine tri komponente:

- upravljanje modulom, koja omogućava punjenje modula u kernelsku memoriju i komunikaciju modula sa ostatkom kernela, proveru da li je modul u memoriji i da li se koristi i izbacivanje modula iz memorije (pod uslovom da se modul ne koristi),
- registracija drajvera, koja omogućava modulu da objavi ostatku kernela da je novi drajver u memoriji i da je raspoloživ za korišćenje. Kernel održava dinamičku tabelu drajvera koji se pomoću posebnog seta programa mogu napuniti ili izbaciti iz memorije u svakom trenutku,
- rezolucija konflikata, odnosno mehanizam koji služi da spreči hardverske konflikte tako što omogućava drajveru da rezerviše hardverske resurse (IRQ, DMA, ports) i time spreči druge drajvere ili autoprobe funkciju da ih koriste.

Na slici 1.3 prikazana je struktura modularnog Linux kernela.



Slika 1.3 *Modularni Linux kernel*

Značajni delovi kernela

Linux kernel čini nekoliko značajnih komponenti:

- upravljanje procesima
- upravljanje memorijom
- upravljanje sistemima datoteka (VFS)
- apstrakcija mrežnih servisa
- podrška za hardverske uređaje
- podrška za različite sisteme datoteka
- podrška za TCP/IP.

Kritične komponente Linux kernela su upravljanje procesima i upravljanje memorijom. Komponenta za upravljanje memorijom kontroliše dodeljivanje memorije i swap prostora procesima, kernelskim komponentama kao i bafersko keširanje. Komponenta za upravljanje procesima kreira procese i omogućava višeprocesni rad (multitasking) dodeljujući procesor procesima po odgovarajućem algoritmu.

Na najnižem nivou kernel sadrži podršku u vidu drajvera za razne hardverske uređaje. S obzirom da postoji veliki broj različitih vrsta hardverskih uređaja broj drajvera je takođe veoma veliki. Za hardverske uređaje iste vrste (npr. diskove), koji vrše sličnu funkciju ali se razlikuju u načinu softverske kontrole, formirane su opšte klase drajvera na sledeći način: svaki član klase pruža isti interfejs prema ostatku kernela čime obezbeđuje podršku za operacije koje su karakteristične za tu vrstu uređaja, a hardver opslužuje na odgovarajući način. Na primer, svi disk drajveri pružaju isti interfejs prema ostatku kernela i svi imaju operacije tipa inicijalizacije drajva, čitanja podataka iz određenog sektora i upisa podataka u određeni sektor.

Takođe, neki softverski servisi koje kernel podržava imaju slične osobine čime se omogućava njihova apstrakcija u klase. Na primer, različiti mrežni protokoli se apstrakuju

u jedan programski interfejs koji se naziva "BSD socket library". Drugi primer je virtuelni sistem datoteka (VFS - virtual filesystem), koji apstrakuje operacije u sistemu datoteka, pri čemu svaki tip sistema datoteka obezbeđuje specifične implementacije raznih operacija. Zahtev za korišćenjem sistema datoteka koji šalje korisnik prolazi kroz VFS sloj, koji isti prosleđuje na odgovarajući drajver za konkretni sistem datoteka.

Upravljanje procesima

Linux koristi standardni UNIX proces mehanizam (fork) koji razdvaja kreiranje procesa i njegovo izvršenje u dve različite operacije:

- sistemski poziv fork, koji kreira novi proces,
- sistemski poziv exec, koji izvršava program u resursima novostvorenog procesa.

Pod UNIX sistemom sve informacije koje operativni sistem mora čuvati da bi kontrolisao jedan proces predstavljaju kontekst tog procesa. Pod Linux operativnim sistemom, svaki proces je u potpunosti opisan identitetom, okolinom, i kontekstom.

Identitet procesa obuhvata sledeće informacije:

- Identifikator procesa (Process ID - PID), pomoću kog Linix kontroliše proces;
- Akreditivi (Credentials). Svaki proces pripada jednom korisniku koji ima svoj user ID i jedan ili više grupnih IDs koji određuju prava pristupa procesu u radu sa datotekama;
- Ličnost (Personality). Ova informacija se ne koristi kod drugih UNIX sistema, a Linux svakom procesu dodeljuje lični identifikator koji može imati uticaja za neke sistemske pozive.

Okolina procesa se nasleđuje od procesa roditelja (odnosno od procesa koji je izazvao kreiranje datog procesa). U okolinu procesa spadaju vektor argumenata koje proces roditelj prosleđuje programu i vektor okoline, odnosno lista promenljivih koje definišu okolinu procesa (environment).

Kontekst procesa je stanje procesa u datom trenutku vremena. Kontekst procesa čine sledeće komponente:

- kontekst za raspoređivanje (scheduling context), koji služi za efikasnu suspenziju ili ponovni start procesa. Obuhvata sve CPU registre, prioritet procesa i kernelski stek procesa;
- statistički kontekst, koji sadrži informacije o resursima koje proces koristi u jednom trenutku, kao i kompletnu upotrebu resursa za vreme trajanja jednog procesa (accounting information);
- tabela datoteka (file table), tj. polje ukazivača na kernelske strukture datoteka;
- kontekst sistema datoteka (file-system context);
- tabela za upravljanje signalima (signal-handler table), koja definiše ukazivače na programe koji se pozivaju nakon određenog signala;

- kontekst virtulene memorije (virtual-memory context), koji potpuno opisuje korišćenje memorije od strane procesa.

Procesi i niti

Linux koristi istu internu reprezentaciju za procese i niti - nit (thread) je jednostavno novi proces koji deli adresni prostor roditelja. Za razliku od novog procesa koji pomoću sistemskog poziva fork formira novi kontekst sa jedinstvenim adresnim prostorom, nit nastaje pomoću sistemskog poziva clone koji kreira novi kontekst, ali dozvoljava novom procesu da deli adresni prostor roditelja.

Dodeljivanje procesora procesima

Linux koristi 2 algoritma za dodelu procesora procesima (process-scheduling algorithms):

- time-sharing algoritam za korektno raspoređivanje između procesa (fair preemptive scheduling). Dodela se vrši na osnovu prioriteta procesa koji definiše korisnik i kredita (efektivni prioritet) koji raste s porastom vremena čekanja na procesor po sledećoj rekurzivnoj formuli: $kredit = \frac{kredit}{2} + prioritet$
- real-time algoritam za procese gde su apsolutni prioriteti mnogo značajniji od ravnomerne raspodele. Linux je ipak soft real-time operativni sistem.

Koji će se algoritam primeniti zavisi od klase u kojoj se proces nalazi (FIFO ili round-robin). Trenutna pozicija procesa u svakoj od klasa određuje se na osnovu prioriteta, što znači da će se izvršavati onaj proces koji ima najviši prioritet, a u slučaju da su prioriteti isti, izvršava se proces koji je najduže čekao. U FIFO klasi procesi nastavljaju da rade sve dok ne završe rad ili ne uđu u blokirano stanje, dok u round-robin klasi svaki proces radi dok mu ne istekne vremenski kvantum (time-slice), posle čega prekida rad i odlazi na kraj liste za čekanje.

Počevši od kernela 2.0 Linux podržava SMP, što znači da se različiti procesi ili niti mogu izvršavati paralelno na posebnim procesorima. Da bi se obezbedila procesorska sinhronizacija kernela u SMP okruženju samo jedan CPU može izvršavati kôd u kernel modu.

Komunikacija između procesa

Komunikacija između procesa obuhvata obaveštavanje procesa o događaju i prenos podataka s jednog procesa na drugi. Kao i UNIX sistem, Linux informiše procese u korisničkom režimu o događaju putem signala. Procesi u kernel modu umesto signala koriste specijalnu vrstu deljive memorije (wait.queue struktura) za interprocesnu komunikaciju.

Za prosleđivanje podataka između procesa koristi se pipe mehanizam, koji omogućava jednosmernu razmenu podataka putem komunikacionog kanala koji proces nasleđuje od roditelja, i deljiva memorija, koja je brza i fleksibilna, ali zahteva sinhronizaciju.

Upravljanje memorijom

Upravljanje memorijom obuhvata upravljanje operativnom (RAM) memorijom i upravljanje virtuelnom memorijom.

Upravljanje operativnom, odnosno fizičkom memorijom obuhvata dodeljivanje (alokaciju) i oslobađanje stranica (pages, normal extent), grupe stranica (large extent) i malih memorijskih blokova (small extent). Fizičkom memorijom se upravlja po sistemu drugova (Buddy heap). Cela fizička memorija se deli na udružene blokove čije su veličine stepeni broja 2. Blokovi se prema potrebi alokacije dalje razbijaju na manje blokove ili se parovi udružuju u veće celine.

Sistem virtuelne memorije povećava ukupan adresni prostor koji je dostupan procesima - sistem kreira stranicu virtuelne memorije na zahtev, upravlja punjenjem te stanice u fizičku memoriju sa diska i povratkom stranice na disk u swap prostor. Kada stranica mora da napusti memoriju i ode na disk izvršava se takozvani page-out alogoritam, koji je na Linux sistemu realizovan LFU konceptom (Least Frequently Used - najređe korišćen). Novi virtuelni adresni prostor formira se nakon kreiranja novog procesa sistemskim pozivom fork, i nakon izvršavanja novog programa sistemskim pozivom exec.

Regioni virtuelne memorije obuhvataju fizičke stanice, odnosno okvire u koje su fizičke stranice smeštene (frames) i swap prostor na disku.

Izvršavanje korisničkih programa

Linux podržava brojne formate za punjenje i izvršavanje programa. Među njima svakako treba istaći stari UNIX format a.out i novi elf format koji je maksimalno prilagođen konceptu virtuelne memorije. Zaglavlje ELF formata opisuje sekcije programa. Sekcije programa su po veličini prilagođene veličini stanice virtuelne memorije.

Program kod kog su funkcije iz sistemske biblioteke direktno ugrađene u kôd programa je program sa statičkim povezivanjem. Glavni nedostatak ovakvog načina povezivanja je povećanje veličine koda, jer svaki poziv funkcije iz biblioteke kopira celu funkciju u kôd. Takođe, sa veličinom koda raste i količina memorije koja je potrebna za njegovo izvršavanje. Na drugoj strani, dinamičko povezivanje je efikasnije u smislu iskorišćenja memorije - sama funkcija se ne kopira u kôd, tako da je za izvršenje potrebna manja količina memorije, ali se programi po nepisanom pravilu izvšavaju sporije.

Ulazno - izlazni sistem

Linux deli uređaje u tri klase: blok uređaje (poput diskova i CD-ROM uređaja), karakter uređaje (poput štampača) i mrežne uređaje. Svaki uređaj je predstavljen specijalnom datotekom (device node, device file) koja se nalazi u direktorijumu /dev root sistema datoteka. Kada korisnik upisuje podatke u datoteku koja predstavlja neki uređaj ili čita iz te datoteke, vrši se neka ulazno-izlazna operacija, odnosno sistem šalje ili prima podatke sa uređaja koji je predstavljen tom datotekom. Time se ukida potreba za postojanjem posebnih programa (a samim tim i posebnom metodologijom programiranja ulazno-izlaznih operacija) neophodnih za rad sa uređajima. Na primer, korisnik može da odštampa

tekst na štampaču jednostavnom redirekcijom standardnog izlaza na datoteku /dev/lp1 koji predstavlja štampač:

```
# cat izvestaj.txt > /dev/lp1
```

Ova komanda će korektno odštampati datoteku izvestaj.txt ukoliko je ona u obliku koji štampač razume (npr. tekstualna datoteka). Međutim, nije preporučljivo da više korisnika istovremeno šalju datoteke na štampač pomoću redirekcije izlaza, jer se ovim zaobilazi red za čekanje za štampač (print spooler). Poseban program, lpr (line printer), obezbeđuje da datoteke poslate na štampu čekaju u redu, i prosleđuje ih štampaču tek kad je štampanje prethodne datoteke završeno. Slični programi postoje za većinu uređaja, tako da korisnici uglavnom ne koriste specijalne datoteke.

Direktorijum /dev nastaje prilikom instalacije Linux sistema i u njemu se nalaze sve specijalne datoteke, bez obzira na to da li je uređaj instaliran na sistem ili ne - postojanje datoteke /dev/sda ne znači da je na sistem instaliran SCSI disk. Postojanje svih datoteka olakšava proces instalacije novog hardvera, tj. oslobađa administratora sistema potrebe za kreiranjem specijalnih datoteka sa korektnim parametrima.

Sistemi datoteka i aktivno UNIX stablo

Linux sistemi datoteka koriste hijerarhijsku strukturu stabla i semantku UNIX sistema datoteka. Interno, kernel sakriva detalje i upravlja različitim sistemima datoteka preko jednog nivoa apstrakcije koji se naziva virtuelni sistem datoteka VFS.

Aktivno Linux stablo datoteka čini jedan ili više sistema datoteka koji su montirani na odgovarajuće direktorijume preko kojih im se pristupa. Osnovu aktivnog stabla datoteka čini korenski sistem datoteka (root filesystem), čiji koreni (root) direktorijum ujedno predstavlja i koreni direktorijum aktivnog stabla datoteka. Zavisno od hardverske konfiguracije i odluke administratora sistema, struktura aktivnog Linux stabla može biti jednostavna (aktivno stablo realizovano jednim sistemom datoteka), ili složena (aktivno stablo realizovano većim brojem sistema datoteka - root, /boot, /var, /usr, /home ...).

Mrežne strukture

Umrežavanje je ključno područje funkcionalnosti Linux sistema. Linux koristi standardni TCP/IP protokol stek kao osnovni komunikacioni protokol, a dodatno podržava i brojne druge protokole koji nisu uobičajeni za komunikaciju dva UNIX sistema (AppleTalk, IPX, Samba).

Interno, umrežavanje pod Linux sistemom obuhvata tri softverska nivoa: socket interfejs, protokol drajvere i drajvere za mrežne kartice.

Osnovni servisi Linux sistema

init

Proces init se pokreće kao prvi proces na svakom Linux sistemu i to je poslednja akcija koju kernel obavlja prilikom podizanja sistema. Kada se pokrene, init nastavlja proces

podizanja operativnog sistema, obavljajući razne inicijalne procedure kao što su provera i akitviranje sistema datoteka i pokretanje servisa (daemons).

Potpun skup aktivnosti koje će proces init obaviti zavisi od načina podizanja samog Linux sistema. Proces init obezbeđuje koncept jednokorisničkog režima rada (single user mode) u kome niko ne može da se prijavi na sistem osim administratora (root), koji u ovom režimu može da koristi komandni interpreter isključivo na konzoli. Osim ovog režima rada, koji se koristi u svrhe administarcije ili oporavka sistema, init obezbeđuje i standardni višekorisnički režim rada (multiuser mode). Neki režimi rada Linux sistema se generalizuju i nazivaju nivoima izvršenja (runlevel) - jednokorisnički i višekorisnički režim predstavljaju dva nivoa izvršenja, a osim njih postoje i dodatni nivoi izvršenja, poput nivoa sa grafičim okruženjem (X) i nivoa sa mrežnim servisima. Linux dozvoljava do 10 nivoa izvršavanja, 0-9, od kojih su samo neki u upotrebi. Nivo izvršenja 0 definiše se kao gašenje sistema (system halt), nivo izvršenja 1 kao jednokorisnički režim, a nivo izvršenja 6 kao ponovno podizanje sistema (system reboot). Ostali nivoi definisani su u datoteci /etc/inittab za konkretnu Linux distribuciju, pri čemu definicije mogu biti različite za različite distribucije.

U stanju normalnog izvršenja proces init obezbeđuje funkcionisanje procesa getty, koji korisnicima omogućava proceduru prijavljivanja na sistem (login) i usvaja procese siročiće (orphan), odnosno procese čiji su roditeljski procesi nestali, čime se održava integritet rodbinskog stabla procesa.

Prilikom zaustavljanja Linux sistema proces init redom obustavlja sve druge procese, deaktivira sve aktivne sisteme datoteka i zaustavlja procesor, a dodatno može obaviti i druge aktivnosti za koje je konfigurisan.

Prijavljivanje sa terminala

Funkciju prijavljivanja preko serijskih linija i sistemske konzole (bez grafičkog okruženja) obezbeđuje proces getty. Proces init kao proces roditelj pokreće posebnu instancu programa getty za svaki terminal na kome je dozvoljeno prijavljivanje na sistem. Dalje, getty čita korisničko ime i o tome izveštava proces login, koji proverava unešeno korisničko ime i lozinku. Ako su korisničko ime i lozinka korektni login pokreće proces shell koji prihvata i izvršava korisničke komande. Proces init će otpočeti novu instancu procesa getty kada proces komandnog interpretera završi aktivnost, odnosno kada se korisnik odjavi sa sistema, ili ako se login proces završi nekorektnim prijavljivanjem. Evidenciju o login procedurama ne vodi kernel već to obavljaju sistemski programi.

syslog

Kernel i mnogi sistemski programi generišu razna upozorenja i poruke o greškama koje se upisuju u datoteke, tako da se mogu pregledati nakon izvesnog vremena. Program koji obavlja funkciju upisivanja poruka u datoteke je syslog. Program se može konfigurisati da poruke aranžira u različite datoteke na osnovu stepena značajnosti i procesa koji je poruke generisao. Na primer, poruke koje generiše kernel se, kao vrlo značajne poruke koje mogu ukazivati na ozbiljne probleme u sistemu, upisuju u posebnu datoteku.

Periodično izvršavanje komandi

Većina korisnika, uključujući sistem administratore često ima potrebu za periodičnim izvršavanjem neke komande. Na primer, sistem administrator može da zakaže periodično izvršenje komande koja čisti direktorijume sa privremenim datotekama (/tmp i /var/tmp), čime sprečava prepunjenje diska. Funkciju periodičnog izvršenja komandi obezbeđuje servis cron. Svaki korisnik može u svojoj crontab datoteci definisati komande koje želi periodično da izvršava i vreme kada te komande treba izvršiti. Cron demon vodi računa o izvršenju komandi u specificiranom vremenu.

Servis koji je sličan cron servisu je at, ali za razliku od cron servisa komandu pokreće samo jednom, u specificiranom trenutku.

Grafički korisnički interfejs

Linux ne ugrađuje korisnički interfejs u kernel - korisnički interfejs se implementira na korisničkom nivou. Koriste se dve vrste interfejsa, alfanumerički i grafički, čime se postiže veća fleksibilnost sistema. Nedostatak ovakvog uređenja je relativno složena implementacija različitih korisničkih interfejsa za svaki program, što operativni sistem čini težim za učenje i korišćenje.

Primarno grafičko okruženje koje se koristi u Linux sistemima je X Window System (skraćeno - X). X ne implementira korisnički interfejs već obezbeđuje alate pomoću kojih grafički korisnički interfejs može da se implementira. Najčešće korišćeni window menadžeri na Linux sistemima su: KDE i Gnome.

Mrežni rad

Umrežavanje je čin povezivanja dva ili više računara relativno visokog stepena autonomije u cilju omogućavanja njihove međusobne komunikacije. Iako proces povezivanja računara u mrežu i konfigurisanja mrežnog okruženja može biti komplikovan, krajnji rezultat je veoma koristan - korisnici mogu koristiti deljene mrežne resurse poput štampača, direktorijuma i up-linkova, a takođe se mogu omogućiti i mehanizmi centralizovane autentifikacije i administracije. Većina osnovnih servisa Linux sistema, poput sistema datoteka, štampanja i arhiviranja podataka može se realizovati pomoću mrežnih funkcija operativnog sistema.

Detaljne informacije o mrežnom radu pod Linux operativnim sistemom korisnici mogu dobiti iz knjiga "Securing and Optimizing Linux: The Ultimate Solution", Gerhard Mourani, koja se besplatno može preuzeti sa sajta www.openna.com i "Linux Network Administrators Guide", koja se takođe besplatno može preuzeti sa sajta www.tldp.org (The Linux Documentation Project).

Prijavljivanje sa mreže

Procedura prijavljivanja sa mreže razlikuje se od klasične procedure prijavljivanja na sistem, koja zahteva postojanje posebne fizičke serijske linije za svaki terminal na kom je potrebno omogućiti prijavljivanje i rad. Za svakog korisnika koji se prijavljuje na sistem preko mreže postoji posebna virtuelna mrežna konekcija. Broj tih konekcija teorijski može

biti neograničen, tako da nije moguće izvršavati poseban getty proces za svaku virtuelnu konekciju.

Za potrebe prijavljivanja sa mreže, umesto gomile getty procesa, postoji glavni demon proces (master daemon, wrapper) koji osluškuje sve nadolazeće zahteve. Ukoliko se pojavi zahtev za prijavljivanje na sistem, wrapper kao proces roditelj pokreće novi proces (telnet ili rlogin demon), sličan getty procesu, koji dalje upravlja pokušajem prijavljivanja, dok glavni demon nastavlja da osluškuje nove zahteve sa mreže.

Deljenje datoteka i mrežni sistemi datoteka

U značajnije servise UNIX i Linux sistema spada i deljenje datoteka na mreži. Mrežni sistemi datoteka funkcionišu na sledeći način: zahtevi za operacijama nad datotekama i direktorijumima šalju se po mreži na računar na čijim se diskovima sistem datoteka nalazi, a korisnik ima utisak da se sve datoteke nalaze na lokalnom sistemu datoteka. Na ovaj način omogućava se deljenje datoteka na krajnje jednostavan način, pošto se ne zahteva nikakva modifikacija korisničkih programa.

Najčešće korišćeni tip mrežnih sistema datoteka je NFS (Network File System), koji je razvila kompanija Sun Microsystems. Drugi servis, koji omogućava pristup Linux mrežnim sistemima datoteka sa MS Windows radnih stanica, je Samba (http://www.samba.org). Po istim protokolima je moguće deliti i štampače.

Elektronska pošta

Elektronska pošta je najpopularniji metod komunikacije putem računara. Svaki korisnik ima prijemno poštansko sanduče (incoming mailbox) u vidu datoteke u specijalnom formatu koja se po pravilu nalazi na direktorijumu /var/spool/mail. Kada neko šalje poštu specijalan program, koji omogućava prijem i slanje pisama, locira prijemno poštansko sanduče i dodaje pismo u mailbox datoteku. Ukoliko se prijemno poštansko sanduče nalazi na drugom računaru, pismo se najpre šalje preko mreže ka destinacionom računaru, čiji će mail program dalje pismo distribuirati u odgovarajuće sanduče.

Poštanski sistem čine dve osnovne vrste programa: MTA - Mail Transfer Agent (na primer sendmail) koji vrše isporuku pošte u lokalno poštansko sanduče i prosleđuju poštu udaljenim računarima, i MUA - Mail User Agent (pine, mutt ili elm) koje služe korisnicima za čitanje i slanje pošte.

Štampanje

Jedan štampač u jednom vremenskom trenuntku može da koristi samo jedan korisnik, ali je krajnje neekonomično ne dozvoliti upotrebu štampača u različitim vremenskim trenucima većem broju korisnika. Zato štampačima upravlja program koji implementira red čekanja za dati štampač (printer queue): svi zahtevi za štampu postave se u red čekanja i kada štampač obavi jedan posao upravljački program mu automatski šalje sledeći. Na ovaj način se omogućava korišćenje štampača većem broju korisnika, pri čemu su korisnici oslobođeni organizacije reda čekanja za štampač.

Program koji implementira red čekanja čuva kopiju koju treba štampati na disku print servera, čime se omogućava aplikativnom programu da preda zahtev za štampu u red i nakon toga nastavi svoje aktivnosti. Disk je relativno brz uređaj u odnosu na štampač, tako da predaja zahteva u red traje kratko. Nakon predaje zahteva aplikativni program ne mora da čeka da zahtev bude odštampan, a takođe ne mora ni da kontroliše proces štampe, tako da nastavlja s daljim radom. Ovakav način organizacije štampanja je jako povoljan, zato što korisnik može da izda zahtev za štampanjem jednog dokumenta, zatim otvori drugi dokument i pošalje ga na štampu bez ikakvog čekanja.

2

BLOK UREĐAJI I ADMINISTRACIJA SISTEMA DATOTEKA

Prilikom instalacije ili nadogradnje operativnog sistema potrebno je utrošiti izvesnu količinu vremena na administraciju diskova i sistema datoteka. Sistemi datoteka su neophodni za instalaciju operativnog sistema i čuvanje datoteka na sekundarnim memorijama. Postupci koji su ovde objašnjeni najčešće se izvode pre instalacije operativnog sistema, ali se mogu ponoviti ukoliko se ukaže potreba za podešavanjem parametara sistema datoteka ili ukoliko se na sistem dodaje novi disk.

Četiri osnovna zadatka iz oblasti administracije diskova su:

- formatiranje diska na niskom nivou, čime se disk priprema za korišćenje. Većina diskova koji se danas proizvode fabrički su preformatirani, tako da je administrator sistema oslobođen tog dela posla;
- podela diska na particije. Disk se deli na particije ukoliko je na računar potrebno instalirati više operativnih sistema ili u cilju razdvajanja sistemskih i korisničkih datoteka, čime se pojednostavljuju postupci backupa i arhiviranja;
- kreiranje sistema datoteka na particijama diska. Korisnici svoje podatke čuvaju kao datoteke koje sa hijerarhijskom strukturom direktorijuma u kojima se nalaze čine sistem datoteka;
- aktiviranje sistema datoteka, montiranjem (mounting) na odgovarajuće direktorijume, čime se formira struktura aktivnog direktorijumskog stabla. Ovaj postupak se obavlja ili automatski, prilikom podizanja sistema, ili ručno.

Blok uređaji

Hard i flopi diskovi. Formatiranje magnetnih medijuma. CD-ROM uređaji i magnetne trake.

U blok uređaje spadaju hard diskovi i uređaji sa izmenljivim medijumima - disketni uređaji (flopi diskovi), CD-ROM uređaji (uključujući DVD i uređaje za pisanje po medijumu) i magnetne trake. Ovi uređaji se mogu koristiti za čuvanje podataka u formi arhiva (koje se po pravilu čuvaju na izmenljivim medijumima) i smeštanje sistema datoteka. U daljem tekstu opisani su blok uređaji i osnovni postupci njihove administracije.

Hard diskovi

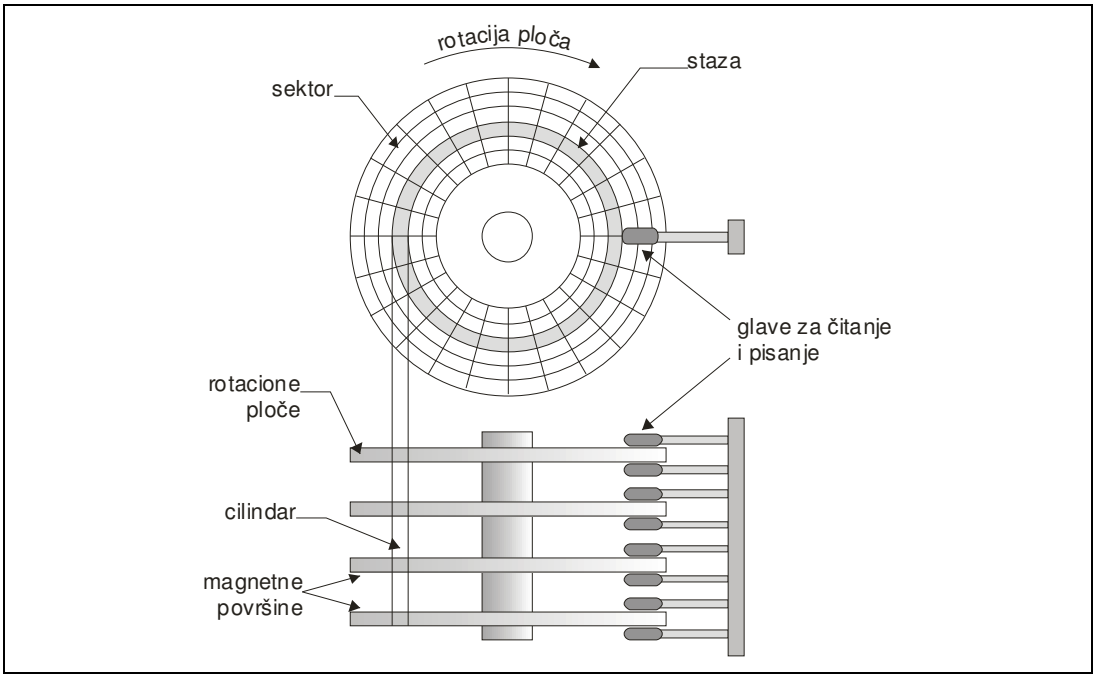
Fizičke osobine i geometrija diskova

Hard disk se sastoji od jedne ili više kružnih ploča premazanih tankim slojem magnetne supstance. Za svaku površinu postoji posebna glava za čitanje i pisanje (read-write head) koja čita ili upisuje podatke sa magnetnih ploča. Ploče rotiraju oko zajedničke ose konstantnom ugaonom brzinom koja najčešće iznosi 5400-7200 obrtaja u minuti za standardne IDE diskove, odnosno 10000 i više obrtaja za profesionalne SCSI diskove. Glave za čitanje i pisanje kreću se pravolinijski po prečniku ploča čime im je, uz rotaciju ploča, omogućen pristup svim delovima magnetne površine. Na slici 3.1. prikazana je struktura diskova.

Procesor računara i disk komuniciraju preko disk kontrolera (disk controller). Disk kontroleri različitih diskova pružaju isti interfejs ka ostatku računara i time pojednostavljuju pristup podacima na disku, tj. oslobađaju računar potrebe za poznavanjem načina rada i kontrolom elektromehanike diska. Dodatne funkcije kontrolera su baferovanje podataka koje treba upisati na disk, keširanje diskova (disk caching) i automatsko obeležavanje neispravnih sektora diska.

Površina diska je podeljena u koncentrične prstenove - staze (tracks), a svaka staza je dalje podeljena na sektore (sectors). Ovom podelom moguće je specificirati lokacije diska na kojima se nalaze podaci, ili alocirati prostor za nove podatke - npr. čitanje podatka sa druge površine, iz staze broj 3, sa sektora 5. Broj sektora po stazi je kod starijih diskova najčešće isti za sve staze, dok noviji diskovi dele spoljašnje staze na veći broj sektora čime obezbeđuju jednaku magnetnu površinu za sve sektore. Tipična količina podataka koja se može upisati u jedan sektor je 512b, i to je najmanja količina podataka koja se može upisati na disk ili pročitati sa diska.

Sve površine magnetnih ploča jednako su podeljene na staze i sektore. To znači da se glave za čitanje i pisanje na svim pločama diska u jednom vremenskom trenutku nalaze na istim stazama. Ekvidistantne staze svih ploča čine jedan cilindar (cylinder). Datoteke koje nisu smeštene u okviru jednog cilindra su fragmentisane - pomeranje glava sa jedne staze na drugu prilikom čitanja ovakvih datoteka unosi kašnjenje. Performanse diska mogu se uvećati smeštanjem datoteke u okviru jednog cilindra kad god je to moguće.



Slika 3.1 *Struktura diskova*

Geometrija diska određena je brojem magnetnih površina (odnosno glava za čitanje i pisanje), cilindara i sektora, i čuva se u posebnoj memorijskoj lokaciji sa baterijskim napajanjem - CMOS RAM. Operativni sistem čita vrednosti koje opisuju geometriju diska prilikom podizanja sistema ili inicijalizacije drajvera.

BIOS nameće limite geometriji diskova, tako da je nemoguće specificirati više od 1024 staza u CMOS RAM-u, što je nedovoljno za diskove velikih kapaciteta. Ovaj problem se prevazilazi na sledeći način: disk kontroler prijavljuje računaru lažnu geometriju diska koja odgovara nametnutim limitima i prevodi adrese lažne geometrije koje dobija od računara prilikom čitanja ili upisa podataka u adrese realne geometrije. Na primer, ako disk ima 8 glava, 2048 staza i 35 sektora po stazi, kontroler će računaru prijaviti disk sa 16 glava, 1024 staze i 35 sektora po stazi, čime broj sektora ostaje u dozvoljenom opsegu. Prilikom prevođenja adresa kontroler će prepoloviti broj glava za čitanje i pisanje i udvostručiti broj staza. Primer ilustruje krajnje jednostavnu situaciju, dok je u realnim slučajevima prevođenje malo komplikovanije (npr. prevođenje geometrije diska sa 8 glava, 1536 staza i 23 sektora je složenije od prethodnog primera). Ovaj postupak menja pogled operativnog sistema na geometriju diska, tako da je nepraktično koristiti metod smeštaja jedne datoteke na jednom cilindru u cilju povećanja performansi. Prevođenje geometrije je problem IDE diskova - SCSI diskovi koriste metod logičkih sektora za pristup podacima (kontroler prevodi broj logičkog sektora u fizičku adresu - glava, cilindar, sektor) i drugi način komunikacije između procesora i kontrolera. Međutim, i realna geometrija SCSI diskova može ostati nepoznata računaru.

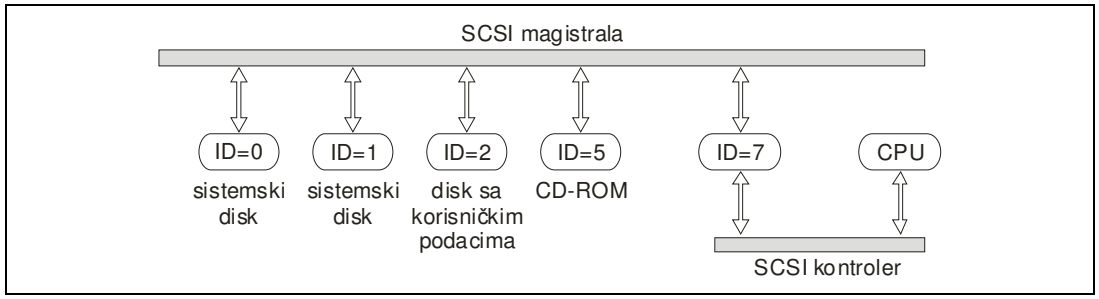
Kako je realna geometrija diskova često nepoznata UNIX sistemi datoteka ne smeštaju datoteke u okvirima jednog cilindra. Situacija je komplikovanija ukoliko na kontroleru postoji keš ili pre-fetch mehanizam, odnosno ako kontroler na osnovu neke logike čita podatke unapred i privremeno ih skladišti u keš memoriji. U cilju povećanja performansi,

UNIX sistemi datoteka koriste metodu smeštanja datoteka u sekvencijalne logičke sektore - ova metoda daje dovoljno dobre performanse, a ne zahteva poznavanje realne geometrije diska.

IDE i SCSI diskovi. Specijalne datoteke koje predstavljaju uređaje.

Dve vrste diskova predstavljaju de facto standarde na PC računarima: IDE i SCSI diskovi. IDE uređaji su dobili naziv po elektronici integrisanoj na samom uređaju (Integrated Drive Electronics). Ovoj klasi uređaja pripadaju relativno jeftini diskovi solidnih performansi: kapacitieta do 160GB, sa brzinama okretanja ploča od 5400-7200 obrtaja u minuti. Kontroleri za IDE uređaje su ugrađeni na matičnim pločama računara (on-board controller) i pružaju interfejs ka računaru pri brzinama od 33-133 Mbit/s. Realna brzina čitanja i pisanja na same magnetne površine znatno je manja, tako da na kontroleru postoji bafer u koji se podaci smeštaju pre upisa na sam disk - na ovaj način se sprečava da performanse sistema značajno padnu prilikom rada sa diskovima. Svaki IDE kontroler ima dva kanala - primarni (primary) i sekundarni (secondary), a na svaki kanal se mogu vezati najviše dva uređaja u odnosu master-slave. Uređaji vezani na različite kanale mogu da primaju ili šalju podatke računaru istovremeno. Na jednom kanalu može biti samo jedan uređaj aktivan u jednom trenutku vremena. Svaki IDE uređaj ima preklopnike (jumpers) koje treba podesiti u željeni režim rada - master ili slave pre vezivanja na kontroler. Administrator sistema određuje način na koji će uređaji biti vezani na IDE kontroler - vezivanje dva brza diska sa kojih se često čita i na koje se često piše na jedan kanal, a dva spora CD-ROM uređaja, koji se povremeno koriste, na drugi kanal nema smisla. Uređaje treba vezati tako da se performanse sistema održe na najvišem mogućem nivou.

SCSI uređaji predstavljaju profesionalnu klasu širokog spektra uređaja - diskova, CD-ROM uređaja, traka, skenera itd. Kontroler za SCSI diskove nije integrisan na matičnim pločama i kupuje se odvojeno. Elektronika na SCSI kontroleru je komplikovanija, interfejs ka računaru je brži (do 320Mbit/s), a na kontroler je moguće vezati od 7 do 15 uređaja što zavisi od modela kontrolera. SCSI uređaji se ne nalaze u master-slave odnosu, već se na kontroler vezuju prema prioritetima. Prioritet svakog uređaja određen je njegovim identifikacionim brojem koji se postavlja preko preklopnika na uređaju. Princip je sledeći: prioritet uređaja sa ID=0 je najviši i treba ga dodeliti sistemskom disku, a prioritet uređaja sa ID=15 je najniži. Identifikacioni broj ID=7 rezervisan je za SCSI kontroler. Na slici 3.2. šematski je prikazan SCSI kontroler:



Slika 3.2 *SCSI kontroler*

Svaki disk u sistemu predstavljen je odvojenom specijalnom datotekom - čvorom (node) sa direktorijuma /dev. Ove datoteke omogućavaju pristup celom disku i koriste se u svrhe particionisanja diska ili pristupanja specijalnim delovima diska kao što je Master Boot Record:

```
# fdisk /dev/hda
# dd if=/dev/hda of=/mnt/floppy/linuxboot count=1
```

Dat je spisak specijalnih datoteka i uređaja na koje se one odnose:

- /dev/hda IDE Primary Master
- /dev/hdb IDE Primary Slave
- /dev/hdc IDE Secondary Master
- /dev/hdd IDE Secondary Slave
- /dev/sda prvi SCSI disk
- /dev/sdb drugi SCSI disk
- /dev/sdc treći SCSI disk
- /dev/sdd četvrti SCSI disk, itd...

RAID (redundantni niz jeftinih diskova)

RAID koncept razvijen je na University of California, Berkeley, sa ciljem da se što bolje iskoriste diskovi malog kapaciteta. RAID tehnika (Redundant Area of Inexpresive Disks) predstavlja različite načine upotrebe diskova radi postizanja veće pouzdanosti i boljih performansi. RAID se deli na nivoe 0-6 koji se mogu realizovati hardverski i softverski. Spominjemo najbitnije:

- RAID level 0 (disk striping) je postupak kojim se podaci ravnomerno raspoređuju na sve diskove u nizu, u cilju poboljšanja performansi sistema. Ovaj postupak se realizuje deljenjem diskova na trake (stripes) čije veličine zavise od tipa operativnog sistema i namene niza diskova. RAID 0 ne unosi redundansu (nema povećanja cene), ali se u slučaju otkaza jednog diska svi podaci nepovratno gube. Tabela 3.1. šematski prikazuje RAID 0 postupak.

Disk 1	Disk 2	Disk 3	Disk 4
A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P
↓	↓	↓	↓

Tabela 3.1 *RAID 0 (Disk Striping)*

- RAID level 1 (disk mirroring) je postupak kojim se za svaki disk u nizu uvodi po jedan rezervni koji u svakom trenutku sadrži sliku originala. RAID 1 uvodi 100% redundanse, zaštita podataka je potpuna, ali je cena sistema duplo veća. Tabela 3.2. šematski prikazuje RAID 1 postupak.

Disk 1	Disk 2	Disk 3 (r1)	Disk 4 (r2)
A	B	A	B
C	D	C	D
E	F	E	F
G	H	G	H
↓	↓	↓	↓

Tabela 3.2 *RAID 1 (Disk Mirroring)*

- RAID level 3 je postupak kojim se podaci ravnomerno raspoređuju na više diskova u nizu u cilju poboljšanja performansi, a u cilju povećanja pouzdanosti uvodi se disk za kontrolu parnosti. Ukoliko dođe do otkaza jednog diska svi podaci su i dalje dostupni. Tabela 3.3. šematski prikazuje RAID 3 postupak.

Disk 1	Disk 2	Disk 3	Disk 4 (par)
A	B	C	par (A, B, C)
D	E	F	par (D, E, F)
G	H	I	par (G, H, I)
J	K	L	par (J, K, L)
↓	↓	↓	↓

Tabela 3.3 *RAID 3 (postupak sa proverom parnosti)*

- RAID level 5 je postupak sličan RAID 3 s tim što se provera parnosti raspodeljuje na sve diskove u sistemu. Podaci su dostupni nakon otkaza jednog diska. Ovaj postupak je poznat pod imenom rotacioni niz parnosti (Rotating Parity Array). Tabela 3.4. šematski prikazuje RAID 5 postupak.

Disk 1	Disk 2	Disk 3	Disk 4
A	B	C	par (A, B, C)
D	E	par (D, E, F)	F
G	par (G, H, I)	H	I
par (J, K, L)	J	K	L
↓	↓	↓	↓

Tabela 3.4 *RAID 5 (rotacioni niz parnosti)*

Flopi diskovi

Disketni uređaj (Floppy Disk Drive) je uređaj koji koristi izmenljive medijume (diskete) za skladištenje podataka. Disketa se sastoji od fleksibilne membrane koja je obostrano

premazana tankim slojem magnetne supstance i odgovara jednoj rotacionoj ploči hard diska. Glave za čitanje i pisanje nalaze se u disketnom uređaju. Ovim je omogućeno sledeće: jedan uređaj može da pristupi različitim medijumima (disketama), a disketi snimljenoj na jednom uređaju može se pristupiti pomoću drugog uređaja.

Diskete se dele na staze i sektore poput hard diskova, pri čemu dve ekvidistantne staze na različitim stranama diskete čine jedan cilindar. Broj staza i sektora je znatno manji nego na hard diskovima. Kontroler za disketne uređaje je integrisan na matičnoj ploči - pomoću njega se na sistem mogu vezati najviše dva uređaja (kojima se u DOS-u i Windows-u pristupa preko logičkih diskova A: i B:). Dve vrste disketnih uređaja predstavljaju standard za PC računare:

- 5.25inch, kapaciteta 360KB sa dvostrukom gustinom zapisa (double density) i 1.2MB sa visokom gustinom zapisa (high density). Ovi uređaji se više ne upotrebljavaju;
- 3.5inch, kapaciteta 720KB (double density), 1.44MB (high density) i 2.88MB (extended density).

Pri tome se više različitih formata disketa može pročitati na istom uređaju - 3.5inch drajv kapaciteta 1.44MB omogućava pristup disketama 720KB i 1.44MB.

S obzirom na razlike koje se javljaju pri radu jednog uređaja sa različitim formatima medijuma sistemu treba saopštiti koji se uređaj koristi i kog su kapaciteta medijumi. Na primer, /dev/fd0H1440 predstavlja prvi disketni uređaj (fd0), standarda 3.5incha, maksimalnog kapaciteta 1.44MB (H), u kome se nalazi disketa visoke gustine zapisa, kapaciteta 1.44MB. Kao posledica, na Linux sistemu postoji veći broj specijalnih datoteka koje predstavljaju disketne uređaje. Dat je spisak specijalnih datoteka, uređaja i standardnih kapaciteta medijuma na koje se one odnose:

- /dev/fd0d360 prvi flopi disk, uređaj 5.25incha, kapacitet 360KB
- /dev/fd1d360 drugi flopi disk, uređaj 5.25incha, kapacitet 360KB
- /dev/fd0h1200 prvi flopi disk, uređaj 5.25incha, kapacitet 1.2MB
- /dev/fd1h1200 drugi flopi disk, uređaj 5.25incha, kapacitet 1.2MB
- /dev/fd0D720 prvi flopi disk, uređaj 3.5incha, kapacitet 720KB
- /dev/fd1D720 drugi flopi disk, uređaj 3.5incha, kapacitet 720KB
- /dev/fd0H1440 prvi flopi disk, uređaj 3.5incha, kapacitet 1.44MB
- /dev/fd1H1440 drugi flopi disk, uređaj 3.5incha, kapacitet 1.44MB
- /dev/fd0E2880 prvi flopi disk, uređaj 3.5incha, kapacitet 2.88MB
- /dev/fd1E2880 drugi flopi disk, uređaj 3.5incha, kapacitet 2.88MB

Imena nodova za disketne uređaje su komplikovana - za Linux su uvedene posebne datoteke, tzv. univerzalni (autodetect) flopi nodovi (/dev/fd0 za prvi, i /dev/fd1 za drugi flopi disk) pomoću kojih se može pristupiti disketi bez eksplicitnog navođenja tipa uređaja i kapaciteta. Pri tom, sistem pokušava da pročita prvi sektor diskete posmatrajući je kao disketu različitog tipa pri svakom pokušaju. Tip diskete je određen kad sistem uspešno pročita prvi sektor i disketa se dalje normalno koristi.

Parametri flopi diskova, koji se koriste prilikom pristupanja disketi pomoću univerzalnih nodova, mogu se podesiti programom setfdprm. Ovaj program omogućava pristup disketama sa nestandardnim kapacitetima (diskete sa nestandardnim brojem sektora - npr.

XDF format), a koristi se i ukoliko autodetekcija medijuma iz nekog razloga otkaže, ili ukoliko odgovarajući nod za disketni uređaj ne postoji. Parametri flopi diskova podešeni ovim programom čuvaju se u datoteci /etc/fdprm.

Formatiranje magnetnih medijuma

Formatiranje je proces upisivanja oznaka koje predstavljaju granice staza i sektora na magnetni medijum, čime se uvodi red u magnetni haos neformatirane površine. Disk (disketa) koji nije formatiran ne može da se koristi.

Terminologija je zbunjujuća za korisnike DOS/Windows operativnih sistema - termin formatiranje diska odnosi se na proces kreiranja sistema datoteka, dok se formatiranje disketa odnosi na kombinovan proces označavanja staza i sektora i kreiranja sistema datoteka.

U cilju razdvajanja ovih pojmova uvode se dva termina:

- formatiranje niskog nivoa (low level formatting), za Linux poznato kao formatiranje,
- formatiranje visokog nivoa (high level formatting), za Linux poznato kao kreiranje sistema datoteka.

Većina IDE i SCSI diskova su fabrički formatirani na niskom nivou, tako da ovaj proces nije potrebno ponavljati. U nekim slučajevima se to i ne preporučuje - disk je ponekad potrebno formatirati na specijalan način da bi se uključile specijalne funkcije, kao što je automatska zamena loših sektora. Diskovi koje treba formatirati obično se isporučuju sa specijalnim softverom koji je za to namenjen, dok se za formatiranje standardnih IDE diskova mogu iskoristiti programi koji se nalaze u BIOS-u računara ili softverski paketi koji se bave tom problematikom.

Prilikom formatiranja moguća je pojava neispravnih površina na disku - takozvanih neispravnih blokova (bad blocks) odnosno neispravnih sektora (bad sectors). Ove površine treba izbegavati prilikom upisa podataka, jer su podaci upisani u njih najčešće nepovratno izgubljeni. Kvalitetniji diskovi imaju ugrađen mehanizam za automatsko obeležavanje neispravnih sektora, ali je glavna logika ugrađena u sisteme datoteka. Alternativna metoda je kreiranje particije koja će sadržati sve neispravne sektore - ovaj pristup je poželjno iskoristiti ukoliko je oštećena veća površina diska.

Komanda badblocks može se koristiti za proveru ispravnosti površine diska ili diskete nakon formatiranja. Kao rezultat izvršenja komande na ekranu se prikazuju brojevi svih neispravnih blokova medijuma koji se analizira - ova lista se kasnije može iskoristiti prilikom kreiranja sistema datoteka. Sledeći primer ilustruje ispitivanje površine diskete koja ima tri neispravna bloka.

```
# badblocks /dev/fd0H1440 1440
62
514
513
```

Inicijalna pretraga za lošim sektorima može se izvesti i pomoću komande mkfs, koja kreira sistem datoteka. Za razliku od komande mkfs komanda badblocks nije destruktivna - postojeći podaci neće biti izbrisani, tako da se može upotrebiti i za ispitivanje površine diska na kome postoji sistem datoteka. Dopuna liste neispravnih blokova u postojećem sistemu datoteka vrši se kombinacijom komandi badblocks i fsck.

Formatiranje disketa

Na Linux sistemu postoji poseban program, fdformat, čija je namena formatiranje disketa. Za razliku od DOS programa format koji pored formatiranja kreira i sistem datoteka, fdformat se ograničava na formatiranje disketa na niskom nivou. Dalje se disketa može upotrebiti za kreiranje sistema datoteka ili kao medijum za arhiviranje.

Programu fdformat potrebno je navesti nod za flopi disk kao parametar:

```
# fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
```

Ukoliko postoji potreba da se kao parametar programu fdformat navede univerzalni nod za flopi disk, potrebno je prvo komandom setfdprm podesiti parametre autodetect uređaja, kao što je prikazano u sledećem primeru:

```
# setfdprm /dev/fd0 1440/1440
# fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
```

Ukoliko se disketa formatira na veći kapacitet od predviđenog fdformat će izvršiti proveru povšine diska i ispitati postojanje neispravnih blokova. Svaki blok na disketi biće proveren nekoliko puta, i ukoliko neispravni blokovi postoje proces formatiranja biće prekinut. Formatiranje diskete na veći kapacitet nije dobra ideja - pojava neispravnih blokova na takvim disketama je česta pojava i ne mora se javiti odmah, čime je rizik od gubitka podataka veći. Neuspešno formatiranje može se upisati u log datoteku /usr/log/messages pomoću syslog programa, ali to nije praktično raditi zbog niske cene disketa i poruka koje fdformat prikazuje korisniku na ekranu.

CD-ROM uređaji

CD-ROM uređaj koristi izmenljive medijume u vidu tankih diskova s visokim stepenom refleksije. Podaci se upisuju na površinu diska u vidu malih rezova, poređanih u spiralu s početkom u centru (što objašnjava termin "narezivanje" diskova). Prilikom čitanja podataka CD-ROM uređaj upućuje laserski zrak duž spirale, koji se može reflektovati na dva načina zavisno od površine s koje se refleksija vrši (glatka površina ili rez). Ovim je omogućeno jednostavno kodiranje bitova, a samim tim i podataka. Ovo je osnovni princip - postupak se svodi na preciznu optiku i elektromehaniku i daleko je komplikovaniji.

U poređenju sa hard diskovima CD-ROM uređaji pripadaju klasi sporih uređaja. Srednje vreme pristupa podacima (average seek time) standardnog IDE diska ne prelazi 15 milisekundi, što je nekoliko desetina puta manje u odnosu na CD-ROM uređaje. CD-ROM uređaj nije prijatno koristiti kao "live" sistem datoteka ukoliko se na njemu nalazi veća količina malih datoteka raspoređenih po direktorijumima. Na primer, neke distribucije Linux sistema obezbeđuju "live" sisteme datoteka na instalacionim medijumima, čime se korisnik oslobađa potrebe da iskopira sve datoteke na hard disk. Takav sistem je relativno spor, s obzirom da distribuciju Linux sistema čini veliki broj malih datoteka. Nasuprot tome, brzina prenosa podataka je relativno visoka u slučaju da se podaci čitaju sekvencijalno (čitanje velikih datoteka ili kopiranje medijuma). CD-ROM je jako koristan uređaj za korišćenje prilikom instalacije softvera - kapacitet medijuma je relativno visok (650-700MB), a brzina nije presudna za instalaciju.

Postoji nekoliko sistema datoteka koji se mogu koristiti za čuvanje podataka na CD-ROM medijumima. Najčešće je korišćen minimalni sistem datoteka specificiran internacionalnim standardom ISO 9660. Većina operativnih sistema prepoznaje ovaj sistem datoteka, a podaci upisani na CD-ROM medijum softverom za narezivanje u jednom operativnom sistemu prenosivi su na druge operativne sisteme i druge računare. Na primer, slike u JPEG formatu ili Video CD narezani pod Linuxom su vidljivi pod Windows operativnim sistemom, ali se Windows izvršni programi ne mogu pokrenuti pod Linux-om bez emulatora (kao što je wine). ISO 9660 sistem datoteka zbog svoje minimalnosti nije u potpunosti upotrebljiv kao "live" UNIX sistem datoteka, tako da je standard nadograđen Rock Ridge ekstenzijom. Rock Ridge proširenje dozvoljava duga imena datoteka, simboličke linkove i ostalo što karakteriše standardne UNIX sisteme datoteka, a pri tome zadržava prenosivost standarda ISO 9660. Linux podržava oba standarda, a detekcija Rock Ridge proširenja je automatska.

Koje će specijalne datoteke biti korišćene za pristup CD-ROM uređajima zavisi od tipa uređaja i interfejsa preko kog je vezan na računar. Ukoliko je CD-ROM uređaj vezan kao Primary Slave na IDE kontroler, korisnik će mu pristupati pomoću datoteke /dev/hdb. Ukoliko je CD-ROM vezan na SCSI kontroler kao SCSI uređaj sa rednim brojem ID=5, korisnik će mu pristupati pomoću datoteke /dev/scd5. Ukoliko se na sistemu nalazi jedan CD-ROM uređaj, njemu se može pristupiti pomoću univerzalnog noda /dev/cdrom.

Magnetne trake

Uređaj za snimanje na magnetne trake (tape drive, streamer) koristi trake, slične audio kasetama. Traka je po prirodi sekvencijalni medijum - stroga sekvencijalnost čini traku sporom i krajnje neprijatnom za rad sa sistemima datoteka. S druge strane, trake su jeftine za proizvodnju, odnosno kupcu se mogu isporučiti trake velike dužine, a samim tim i relativno velikog kapaciteta (2GB-20GB i veće) po pristupačnim cenama. Veliki kapacitet ih čini pogodnim za backup i arhiviranje, a to su postupci koji ne zahtevaju brze uređaje i direktan pristup medijumu.

Podela diskova na particije

Boot sektori, particione tabele i tipovi particija. Podela diska na particije. Predstavljanje particija specijalnim datotekama.

Hard disk se po potrebi može podeliti na nekoliko delova - particija (partitions), koje se ponašaju kao odvojeni diskovi. Disk se deli na particije ukoliko je na računar sa jednim diskom potrebno instalirati više operativnih sistema. Pri tom svaki operativni sistem koristi svoju particiju, a po potrebi može da čita i piše datoteke na druge ukoliko kernel nudi podršku za sisteme datoteka koji se na tim particijama nalaze. Bez particija instalacija većeg broja operativnih sistema bila bi moguća samo sistemima sa dva ili više diskova. Deljenje diska na particije (particionisanje) često se vrši i na računarima na kojima je instaliran samo jedan operativni sistem - na taj način se razdavajaju sistemske i korisničke datoteke, čime se pojednostavljuju postupci backupa i arhiviranja.

Diskete i CD-ROM medijumi ne dele se na particije. Razlozi su sledeći: diskete su medijumi malog kapaciteta, a CD-ROM medijumi se najčešće koriste za arhiviranje podataka pa particionisanje ne bi imalo nikakvog smisla.

Boot sektori, particione tabele i tipovi particija

Master Boot Record, boot sektori i particione tabele

Informacije o svim particijama diska čuvaju se u prvom logičkom sektoru, tj. u prvom sektoru prve staze sa prve površine diska. Ovaj sektor je poznat pod imenom Master Boot Record (MBR) i njemu BIOS pristupa prilikom boot procedure, tj. svaki put kad se računar uključuje. MBR sadrži mali program koji očitava particionu tabelu, proverava koja je particija aktivna, i očitava prvi sektor aktivne particije (boot sektor). U boot sektoru se nalazi mali program čijim pokretanjem započinje boot-strap, odnosno punjenje RAM memorije operativnim sistemom.

Particionisanje diska je konvencija koje se pridržava većina operativnih sistema uključujući UNIX. Postoje operativni sistemi koji se ne pridržavaju strogo ove konvencije, ali mogu koegzistirati na istom disku sa drugim operativnim sistemima. BSD UNIX zahteva jednu particiju (BSD koristi termin slice) kao granicu u okviru koje se vrši particionisanje internim metodama particionisanja. Operativni sistemi koji ne podržavaju particionisanje ne mogu koegzistirati na istom disku sa drugim operativnim sistemima.

Informacije o particionoj tabeli mogu se dobiti pomoću komande fdisk -l:

```
# fdisk -l /dev/sda

Disk /dev/sda: 255 heads, 63 sectors, 262 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/sda1   *         1         6     48163+  83  Linux
/dev/sda2             7        30    192780   82  Linux swap
```

```
/dev/sda3            31       262   1863540   83  Linux
```

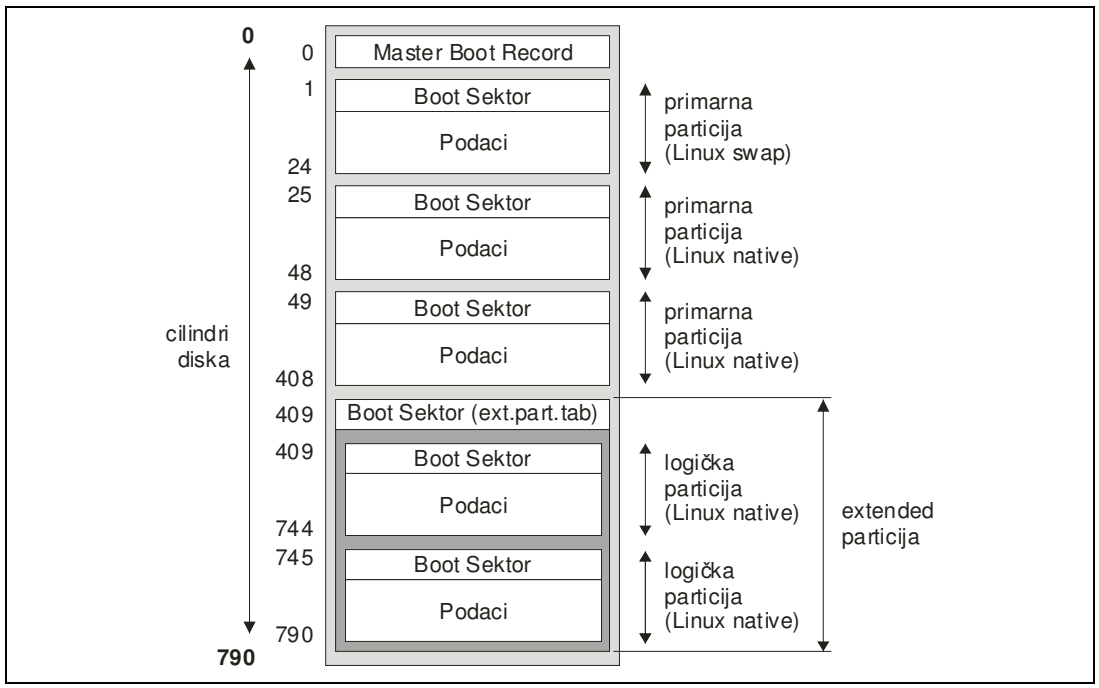
Podatke o particijama treba sačuvati (zapisati na papiru ili snimiti na disketu) da bi se u slučaju oštećenja particiona tabela mogla rekonstruisati bez gubitka datoteka. Oštećena particiona tabela može se oporaviti programom fdisk.

Extended i logičke particije

Originalan koncept particionisanja diskova na PC računarima dozvoljavao je najviše 4 particije po jednom disku, što se ubrzo pokazalo kao nedovoljno. Razlozi su sledeći: nemoguće je na samo 4 particije instalirati više operativnih sistema (naročito ako neki od njih zahtevaju dodatne particije, kao što su swap i boot), boot manager, i odvojiti particiju za korisničke podatke.

Problem je rešen uvođenjem extended particije koja služi kao okvir u kome se mogu kreirati nekoliko logičkih particija. Logičke particije se ponašaju kao primarne, ali se razlikuju po načinu kreiranja. Na ovaj način se maksimalni broj particija po disku uvećava. Informacije o logičkim particijama čuvaju se u boot sektoru extended particije, koji se još naziva i extended partition table. Na disku može postojati najviše jedna extended particija.

Na slici 3.3 šematski je prikazan primer particionisanja jednog diska:



Slika 3.3 *Primer particionisanog diska*

Disk je podeljen na četiri particije - tri primarne i jednu extended, u okviru koje su formirane dve logičke particije. Disk se ponaša kao da na njemu postoji pet primarnih particija, pri čemu osnovni koncept particionisanja nije narušen - u particionoj tabeli se vodi evidencija o samo četiri particije.

Tipovi particija

Particione tabele (u Master Boot Recordu i boot sektoru extended particije) sadrže jedan bajt po particiji koji identifikuje tu particiju. Na taj način se identifikuje koji operativni sistem koristi particiju i u koje svrhe (npr. kao sistem datoteka ili swap prostor). Vrednosti bajta za identifikaciju particije nisu standardizovane, a one koje se najčešće koriste date su u sledećem spisku:

- 0 prazna particija, tj. neiskorišćen prostor
- 5 Extended
- 80 Old MINIX
- 81 Linux / Minix
- 82 Linux swap
- 83 Linux native
- 85 Linux extended
- fd Linux raid auto
- a5 FreeBSD
- a6 OpenBSD
- a9 NetBSD
- 1 DOS 12bit FAT
- 4 DOS 16bit FAT (za sisteme datoteka manje od 32MB)
- 6 DOS 16bit FAT (za sisteme datoteka veće od 32MB)
- 7 HPFS / NTFS
- 64 Novell
- a OS/2 boot manager.

Detaljnije informacije mogu se dobiti pomoću programa fdisk (opcija l - list known partition types).

Podela diska na particije

Većina operativnih sistema koristi posebne programe za rad sa particijama diska, od kojih neki mogu kreirati razne vrste particija. Particije koje će jedan operativni sistem koristiti treba kreirati pomoću programa za particionisanje koji se isporučuje uz taj operativni sistem. Na taj način je korisnik siguran da su sve informacije specifične za datu particiju upisane na disk.

Program za particionisanje diskova koji se isporučuje uz Linux je fdisk (istoimeni program se isporučivao uz DOS i Windows 9x/ME). Detalji o njegovom korišćenju mogu se naći u on-line dokumentaciji (man pages). U nekim distribucijama Linux sistema (kao što je Red Hat) mogu se naći i programi cfdisk, koji je prijatniji za korišćenje i nudi lepši interfejs prema korisniku, i sfdisk, koji je moćniji i bogatiji opcijama, ali i teži za korišćenje.

fdisk

Program fdisk se pokreće na sledeći način:

```
# fdisk device
```

gde je device nod blok uređaja (diska) koga treba particionisati.

Program fdisk je interaktivni program - korisnik dolazi do rezultata navigacijom kroz sistem tekstualnih menija. Pri tom pomoć se može dobiti pomoću opcije m:

```
# fdisk /dev/sda

Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help):
```

fdisk omogućava korisniku da uradi sledeće stvari:

- prikazivanje particione tabele (p - print the partition table),
- pregled podržanih tipova particija (l - list known partition types),
- kreiranje primarnih, extended i logičkih particija (a - add a new partition),
- brisanje particija (d - delete a partition),
- promena tipa particija (t - change a partition's system id),
- postavljanje flega aktivne particije (a - toggle a bootable flag).

Pri tome promene se ne upisuju na disk dok korisnik ne napusti program pomoću opcije w - write table to disk and exit. Napuštanje programa pomoću opcije q - quit without saving changes ne povlači upisivanje promena na disk.

Ukoliko korisnik želi da promeni veličinu particije postupak je sledeći: kreiranje backup-a svih podataka sa particije, brisanje particije, kreiranje nove particije, povratak podataka na novu particiju. Ako je pri tome potrebno da se neka druga particija smanji, postupak je još složeniji. Program fdisk korisniku ne nudi opciju za jednostavnu promenu veličine particija (resizing) - promena veličine particija je složen postupak pomeranja granica i promene struktura u sistemima datoteka i izvodljiva je samo pomoću specijalnih programa kao što je Partitioner (isporučuje se u SuSE distribuciji Linux sistema). Ovaj postupak se ipak ne preporučuje bez sistema neprekidnog napajanja (UPS), jer nestanak struje povlači

gubitak svih podataka sa diska. Pod Windows operativnim sistemom program koji se snalazi sa promenom veličine FAT i NTFS particija je Partition Magic (proizvod kompanije Power Quest).

Na sistemima sa IDE diskovima boot particija (particija na kojoj se nalazi kernel) se mora nalaziti u okvirima prvih 1024 cilindra. Razlog je sledeći: prilikom podizanja sistema, disk se koristi najpre preko BIOS-a. Kernel se učitava u RAM memoriju sa boot particije, a zatim sistem ulazi u zaštićeno stanje (protected mode). Kako BIOS ima limit na 1024 cilindra, slika kernela se mora nalaziti u tim okvirima. Kad sistem uđe u zaštićeno stanje ovaj limit više nije validan. Ovaj limit ne važi za novije varijante BIOS-a i IDE diskova.

Prilikom kreiranja particija preporuka je da se particijama dodeli paran broj sektora, jer Linux sistemi datoteka koriste blokove veličine 1KB (dva sektora). Neparan broj sektora dovodi do neiskorišćenja poslednjeg sektora, što je ružna navika i neke verzije programa fdisk korisnika na to upozoravaju.

Program fdisk može korisniku na ekranu prikazati particione tabele ili veličinu particija bez ulaska u interaktivni režim rada - za to se koriste parametri -l i -s:

```
# fdisk –l [–u] [device ...]
# fdisk –s partition ...
```

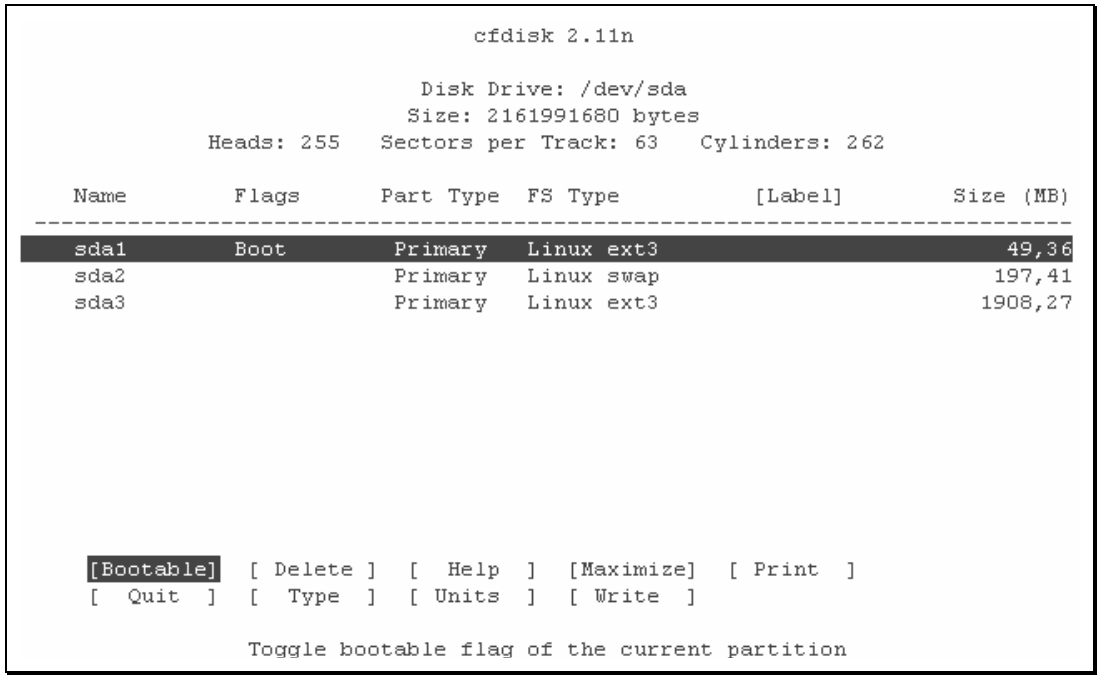
Ako se kao parametar navede -l program prikazuje particione tabele svih uređaja koji su navedeni kao argumenti (devices). Ako se nijedan uređaj ne navede kao argument, fdisk će prikazati particione tabele opisane u datoteci /proc/partitions (datoteka je opisana u poglavlju aktivno UNIX stablo - proc sistem datoteka). Veličine particija izražene su brojem cilindara (ukoliko se navede parametar -u veličine particija biće izražene brojem sektora).

Sa parametrom -s program prikazuje veličinu particija koje su navedene kao argumenti u sistemskim blokovima.

```
# /sbin/fdisk –s /dev/sda1 /dev/sda2
/dev/sda1: 48163
/dev/sda2: 192780
```

cfdisk

Da bi izlaganje bilo potpuno, ukratko ćemo pomenuti program cfdisk, čiji je radni ekran prikazan na slici 3.4.



Slika 3.4 *Radni ekran programa cfdisk*

Pomoću programa cfdisk korisnik može da obavi iste akcije koje se mogu obaviti i programom fdisk. Znači, korisnik može navigacijom kroz tekstualne menije da kreira nove i briše postojeće particije, označi aktivnu particiju, definiše tip particije i na kraju upiše particionu tabelu na disk. Pri tome, na ekranu je u svakom trenutku vidljiva tabela u kojoj su pregledno navedene i ukratko opisane sve kreirane particije i neiskorišćen prostor na disku.

Specijalne datoteke i particije diska

Svaka primarna, extended i logička particija predstavljena je jednom specijalnom datotekom sa direktorijuma /dev. Konvencija o imenima nodova za particije je sledeća: na ime diska treba dodati broj particije. Brojevima od 1-4 označavaju se primarne i extended particije, a brojevima većim od 5 logičke. Tako npr. /dev/hda1 predstavlja prvu particiju na primary master disku, a /dev/sdb7 treću logičku particiju na sedmom SCSI disku. U tabeli 3.5 dat je pregled nodova za sve particije:

Uređaj	Primarne particije	Logičke particije
IDE Primary Master	/dev/hda[1-4]	/dev/hda[5-16]
IDE Primary Slave	/dev/hdb[1-4]	/dev/hdb[5-16]
IDE Secondary Master	/dev/hdc[1-4]	/dev/hdc[5-16]
IDE Secondary Slave	/dev/hdd[1-4]	/dev/hdd[5-16]
Prvi SCSI disk	/dev/sda[1-4]	/dev/sda[5-16]
Drugi SCSI disk	/dev/sdb[1-4]	/dev/sdb[5-16]
Treći SCSI disk	/dev/sdc[1-4]	/dev/sdc[5-16]
Četvrti SCSI disk	/dev/sdd[1-4]	/dev/sdd[5-16]

Tabela 3.5 *Specijalne datoteke kojim su predstavljene particije*

Sistemi datoteka

UNIX sistem datoteka. Tipovi sistema datoteka. Kreiranje, aktiviranje i deaktiviranje sistema datoteka. Programi za rad sa sistemima datoteka. Diskovi bez sistema datoteka. Swap kao kvazi sistem datoteka.

Većina programa radi sa sistemima datoteka a mali broj programa sa particijama - ti programi neće raditi ukoliko sistem datoteka nije kreiran. Kada se podaci snimaju na disk, ne snimaju se direktno na particije, već strogo u sisteme datoteka. S druge strane, programi za rad s particijama (kao što su fdisk i programi za kreiranje sistema datoteka - npr. mkfs) su destruktivni po pitanju podataka koji na particijama postoje. Na primer, brisanjem jedne particije brišu se svi sistemi datoteka koji se na njoj nalaze, a samim tim i podaci na njima.

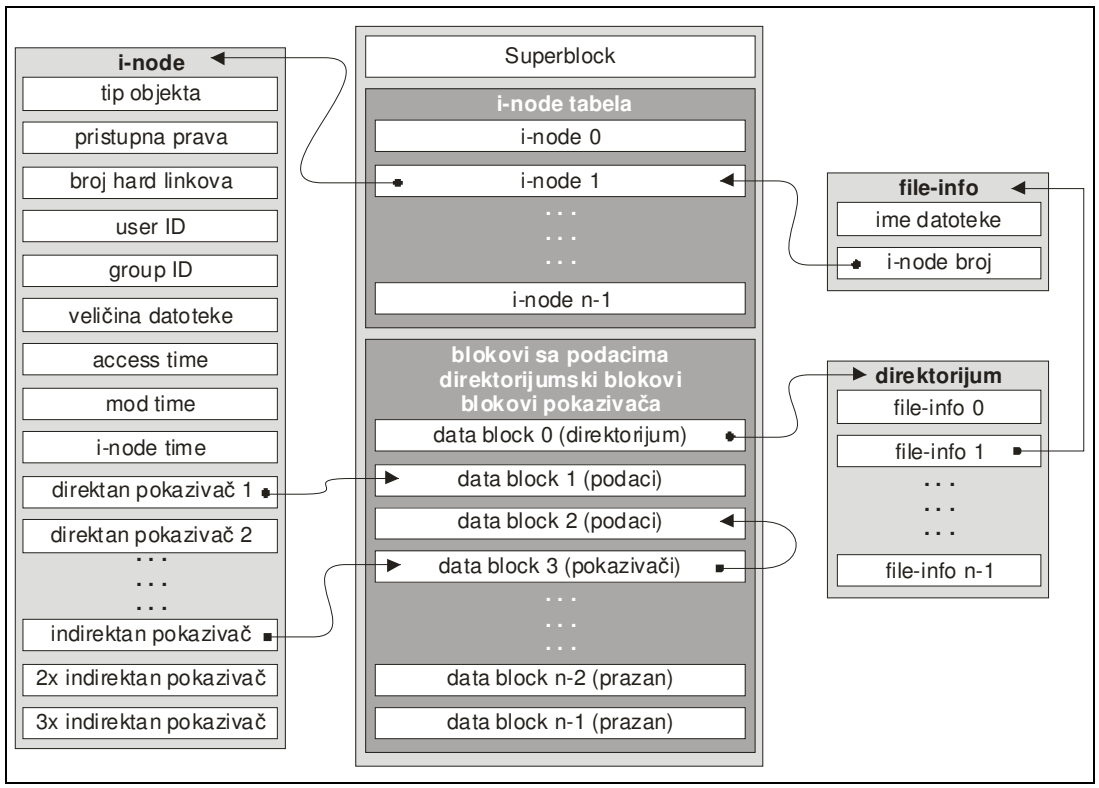
Da bi disk ili particija mogli da se iskoriste za skladištenje podataka potrebno je na njima kreirati sisteme datoteka. Sistem datoteka predstavlja način organizacije datoteka na sekundarnim memorijskim medijumima, tj. sistem datoteka je skup metoda i struktura podataka koje operativni sistem koristi za čuvanje datoteka. Sistem datoteka čine: zaglavlje (u kome se nalazi najmanje podataka, ali su ti podaci neophodni za funkcionisanje sistema datoteka), strukture za organizaciju podataka na medijumu (meta-data area) i sami podaci, odnosno datoteke i direktorijumi. Zaglavlje i meta-data area čine premašenje (overhead) sistema, ali bez njih sistem datoteka ne može da funkcioniše.

Sa korisničke tačke gledišta, zahvaljujući VFS-u, UNIX sve sisteme datoteka posmatra na isti način, bez obzira da li su istog ili različitog tipa, da li se nalaze na lokalnom disku računara ili na mreži. UNIX posmatra svaki sistem datoteka kao nezavisnu hijerarhijsku strukturu objekata (direktorijuma i datoteka) na čijem se vrhu nalazi root direktorijum (*/*). Objektima se pristupa pomoću relativne ili apsolutne putanje i imena objekta. U objekte UNIX sistema datoteka spadaju:

- regularne datoteke,
- direktorijumi (mogu se posmatrati kao specijalne datoteke koje sadrže objekte sistema datoteka, uključujući i poddirektorijume),
- hard linkovi (alternativna imena datoteka),
- simbolički link (prečice, odnosno datoteke čiji su sadržaji putanje i imena objekata na koji upućuju),
- blok i karakter specijalne datoteke (opisuju uređaje, odnosno drajvere u kernelu. Korišćenjem ovih datoteka mogu se vršiti ulazno-izlazne operacije na uređajima koje opisuju. UNIX drajver pomoću specijalne datoteke korisniku predstavlja uređaj kao tok bajtova (stream), odnosno datoteku),
- imenovani pipeline.

UNIX sistemi datoteka

Osnovna struktura svih UNIX sistema datoteka je slična. Tipičan UNIX sistem datoteka šematski je prikazan na slici 3.5:



Slika 3.5 *UNIX sistem datoteka*

UNIX sistem datoteka čine:

- zaglavlje (superblock),

- tabela indeksnih čvorova (i-node tabela),
- blokovi sa podacima (data blocks),
- direktorijumski blokovi (directory blocks),
- blokovi indirektnih pokazivača (indirection block).

Superblok (superblock) je zaglavlje sistema datoteka i sadrži informacije o sistemu datoteka u celini, kao što su njegova veličina, tip i zastavica čistoće (dirty flag). U superbloku se nalazi zaglavlje i-node tabele i zaglavlja listi slobodnih i-node čvorova i slobodnih blokova za podatke. Superblokovi svih aktivnih sistema datoteka keširaju se u RAM memoriji računara i periodično se upisuju na disk.

U okviru dela sa podacima nalaze se blokovi podataka, direktorijumski blokovi i blokovi indirektnih pokazivača. Sadržaj regularnih datoteka čini određeni broj blokova podataka. Direktorijumi se posmatraju kao specijalne datoteke čiji sadržaj čine direktorijumski blokovi koji sadrže datoteke, odnosno njihove specijalne informacije (file-info). Direktorijumski blokovi su tabele sastavljene od određenog broja file-info struktura, za svaku datoteku koja se tu nalazi. File-info struktura se u literaturi pominje i kao kontrolni blok datoteke - FCB (File Control Block). U skladu sa teorijom operativnih sistema, file-info struktura predstavlja jedan deo kontrolnog bloka datoteke. Svaki objekat koji se nalazi u direktorijumu (directory-entry) predstavljen je jednom file-info strukturom. Svaka file-info struktura sadrži ime objekta kog predstavlja i broj indeksnog čvora kojim je taj objekat u potpunosti opisan. Na ovaj način je omogućeno kreiranje hard linkova.

I-node (indeksni čvor) je osnovna struktura UNIX sistema datoteka koja u potpunosti opisuje jedan objekat. Shodno teorijskom pristupu, indeksni čvor je drugi deo kontrolnog bloka datoteke. Svaki UNIX sistem datoteka ima tabelu indeksnih čvorova. I-node sadrži sve informacije o objektu koji opisuje osim imena:

- tip objekta (npr. regularna datoteka, direktorijum ili simbolički link) i pristupna prava za tri vlasničke kategorije,
- broj hard linkova na dati objekat,
- user ID, odnosno ID korisnika koji je vlasnik objekta,
- group ID, odnosno ID grupe korisnika kojoj objekat pripada,
- veličinu objekta izraženu u bajtovima,
- vreme zadnjeg pristupa objektu (access time) u UNIX vremenskom formatu,
- vreme zadnje modifikacije objekta (mod time) u UNIX vremenskom formatu,
- vreme zadnje modifikacije indeksnog čvora objekta (i-node time) u UNIX vremenskom formatu,
- listu direktnih pokazivača na blokove sa podacima, koja je dovoljna da se adresiraju prvih 10-12 blokova podataka koji čine početak datoteke (broj zavisi od tipa sistema datoteka),
- listu indirektnih pokazivača (lista pokazivača na jednostruke, dvostruke i trostruke indirektne blokove).

Dodeljivanje prostora datotekama

Kompromis između veličine i-node tabele i brzine rada nastao je u originalnoj verziji UNIX sistema datoteka. Naime, većina UNIX datoteka je relativno male veličine. Umetanjem prvih 10-12 pokazivača na blokove sa podacima u i-node, i-node tabela će takođe biti relativno mala. Na taj način, manje datoteke se mogu potpuno opisati i-node čvorom. Prilikom alokacije prostora za veće datoteke koristi se dodatni blok pokazivača na blokove podataka (single indirection block). Za još veće datoteke dodatni prostor se može alocirati korišćenjem dvostrukih i trostrukih indirektnih pokazivača. Ova metoda dinamičke alokacije prostora je efikasna i kao takva se sa malim modifikacijama koristi u najnovijim verzijama UNIX i Linux sistema datoteka.

Na primeru ufs sistema datoteka prikazane su maksimalne veličine datoteka koje se mogu dobiti alokacijom prostora pomoću direktnih i 32-bitnih indirektnih pokazivača. Pretpostavlja se realna veličina sistemskog bloka (1KB). Jedan indirektni blok može sadržati najviše 1KB/4B=256 32-bitnih pokazivača:

- samo sa direktnim pokazivačima: 12 * 1 KB = 12 KB
- + indirektni pokazivač: 12 KB + 256 * 1 KB = 268 KB
- + dvostruki indirektni pokazivač: 268 KB + 256^2 * 1 KB ≈ 64 MB
- + trostruki indirektni pokazivač: 64 MB + 256^3 * 1 KB ≈ 16 GB

Na sledećem primeru prikazane su maksimalne veličine datoteka u sistemu datoteka sa sistemskim blokovima maksimalne veličine (8KB). Tada jedan indirektni blok može sadržati najviše 8KB/4B=2048 32-bitnih pokazivača:

- samo sa direktnim pokazivačima: 12 * 8 KB = 96 KB
- + indirektni pokazivač: 96KB + 2048 * 8 KB ≈ 16 MB
- + dvostruki indirektni pokazivač: 16 MB + 2048^2 * 8 KB ≈ 32 GB
- + trostruki indirektni pokazivač: 32 GB + 2048^3 * 8KB ≈ 64 TB

Rupe u datotekama

Neki UNIX sistemi datoteka dozvoljavaju kreiranje rupa u datotekama (hole) pomoću lseek() sistemskog poziva. Rupe u datoteci ne zauzimaju prostor na disku, a sistem datoteka ih simulira određenim brojem nula. Na ovaj način se smanjuje broj upotrebljenih blokova podataka za datoteke koje u svom binarnom obliku imaju velike nizove nula (male binarne datoteke, deljene biblioteke i neke baze podataka). Rupe se implementiraju upisivanjem specijalnih vrednosti na mestima adresa blokova podataka u indirektnim pokazivačima. Te vrednosti znače da ni jedan blok podataka nije upotrebljen za taj deo datoteke, već da se na tom mestu nalazi rupa.

Konvencija o imenima objekata sistema datoteka

U UNIX sistemima datoteka imena datoteka i direktorijuma formiraju se na sledeći način:

- Dozvoljeni karakteri su: sva velika slova (A-Z), sva mala slova (a-z), cifre (0-9), crta (-), underscore (_) i tačka (.). Opcije UNIX komandi počinju jednom (-) ili dvema crtama (--), tako da imena objekata ne treba počinjati tim karakterima. Ukoliko ime objekta počinje tačkom (.), on se smatra skrivenim i može se videti pomoću komande ls samo ako je zadata s parametrom -a (all). Datoteke koje služe za inicijalizaciju i podešavanje radnog okruženja najčešće su skrivene (.bash_profile, .bash_history);
- Zabranjeni karakteri su: ! " ' ` ; : / \ $ < > ( ) [ ] { } ~
- Ukoliko se umesto imena direktorijuma u apsolutnoj putanji navede ~ korisnik će biti preusmeren na home direktorijum. Na primer, komanda less ~/myfile.txt prikazaće sadržaj datoteke myfile.txt koja se nalazi u home direktorijumu korisnika koji je komandu zadao, bez obzira na tekući direktorijum.

UNIX u komandnoj liniji nema pojam ekstenzije, već samo imena objekta. Datoteke nmap.tar.gz i sample.mpg posmatraju se kao regularne datoteke, bez obzira na sadržaj, pri čemu se tip datoteke može odrediti pomoću komande file. Pojam ekstenzije se uvodi pod grafičkim radnim okruženjem, u smislu asociranja datoteka sa programom na osnovu ekstenzija (npr. datoteka sa ekstenzijom doc biće otvorena u Open Office Writeru, a ne u C kompajleru). Direktorijumi najčešće nemaju ekstenzije, iako ne postoji neko pravilo kojim bi se to eksplicitno zabranilo.

Maksimalne dužine imena datoteka i direktorijuma određene su sistemskom promenljivom NAME_MAX na osnovu tipa sistema datoteka na kom se objekat nalazi. Ova promenljiva ne može se promeniti, niti videti pomoću komandi env ili echo, ali je zato svaki korisnik može videti pomoću getconf sistemskog poziva.

```
# getconf NAME_MAX /
255
```

Ukoliko se ova komanda pokrene dvaput nad direktorijumima koji se nalaze na istom sistemu datoteka, isti rezultat se vraća u oba slučaja. Ukoliko se direktorijumi nalaze na različitim tipovima sistema datoteka, npr. ukoliko je /etc na ext2 sistemu datoteka, a /public na FAT 8.3 sistemu, komande getconf NAME_MAX /etc i getconf NAME_MAX /public vratiće različite rezultate.

Tipovi sistema datoteka

Razne varijante UNIX sistema i distribucija Linux sistema podržavaju različite domaće (native) i strane (foreign) sisteme datoteka. Najpoznatiji domaći sistemi datoteka su navedeni i mogu se aktivirati na većini UNIX sistema.

- minix - najstariji, i verovatno najpouzdaniji domaći UNIX sistem datoteka. Maksimalna veličina minix sistema datoteka je 64 MB, a imena datoteka ne mogu biti duža od 30 karaktera;
- xia - modifikovana varijanta minixa, ukida limite na veličinu sistema datoteka i broj karaktera u imenu datoteke, ali ne donosi nove mogućnosti. Retko se koristi, ali se smatra da je pouzdan;

- ext2 - Linux second extended, jako popularan sistem datoteka visokih performansi. Ovaj sistem datoteka je nekoliko godina predstavljao "Linux default";
- ext3 - može se posmatrati kao "ext2 + journaling". Potpuno je kompatibilan sa prethodnom verzijom (ext2), tako da se nadogradnja ostvaruje jednostavnim kreiranjem dnevnika. Za funkcionisanje ext3 sistema datoteka neophodna je podrška na nivou kernela. Ext3 podržava tri režima vođenja dnevnika - journal, ordered i writeback, koji na različite načine utiču na pouzdanost i performanse sistema;
- ReiserFS - journaling sistem datoteka solidnih performansi. U odnosu na ext3 ima veće premašenje, brži je pri radu sa malim datotekama, ali je relativno nestabilan pri radu sa velikim datotekama.

Dodatno, na UNIX-u postoji podrška za nekoliko tipova stranih sistema datoteka čime je omogućena relativno laka razmena datoteka sa drugim operativnim sistemima. Strani sistemi datoteka ponašaju se slično domaćim ali ne moraju imati sve funkcije domaćih sistema datoteka (npr. hard linkove) i mogu imati ograničenja koja na domaćim sistemima datoteka ne postoje (npr. stara verzija FAT sistema datoteka iz DOS-a ima ograničenje na dužinu imena datoteke na 8+3 karaktera).

- msdos - omogućava razmenu datoteka sa DOS i OS/2 FAT sistemom datoteka. Može se aktivirati za čitanje i pisanje (read-write).
- umsdos - proširenje msdos sistema datoteka pod Linux-om. Dodata je podrška za duga imena datoteka, vlasništvo, pristupna prava, linkove i specijalne datoteke. Može se koristiti kao Linux native sistem datoteka, a neke distribucije Linux sistema dozvoljavaju instalaciju operativnog sistema na njemu. Ovo oslobađa administratora sistema potrebe za kreiranjem posebnih Linux native particija, ukoliko se Linux instalira na DOS/Windows sistemu. Npr, Slackware Linux nudi korisniku instalaciju operativnog sistema na FAT sistemu datoteka u direktorijumu \Linux. Prilikom podizanja sistema, sa kernelom se u RAM učitava umsdos drajver i sistem dalje koristi direktorijum \Linux kao root sistem datoteka. Pri tome, LILO (LInux LOader) ne može da pokrene operativni sistem sa umsdos sistema datoteka, pa je potrebno koristi alternativni DOS/Windows program loadlin. Direktorijum \Linux je vidljiv iz DOS ili Windows sistema i zbog velikog broja malih datoteka može povećati slack.
- vfat - proširenje FAT sistema datoteka sa većim kapacitetom poznato pod imenom FAT32. Većina Windows 9x/ME sistema koristi FAT32.
- iso9660 - standard za CD-ROM sisteme datoteka. Može se koristiti sa Rock Ridge proširenjem koje dozvoljava duža imena datoteka i simboličke linkove.
- hpfs - OS/2 High Performance File System.
- ntfs - Windows NT sistem datoteka. Karakteriše ga postojanje dugih imena datoteka, simboličkih linkova (shortcuts), vlasništva, pristupnih prava, dnevnika transakcija, mount-point direktorijuma i kripto-zaštite. Linux sa kernelom 2.4.x dozvoljava da se ntfs aktivira u režimu čitanja (read-only) - kompajliranjem kernela moguće ga je aktivirati u režimu čitanja i pisanja (read-write), ali se to ne preporučuje. Stabilna podrška za pisanje na NTFS sistemu datoteka dodata je u

verzije Linux kernela 2.6.x, koje dozvoljavaju aktiviranje NTFS sistema datoteka u režimu čitanja i pisanja (read-write) bez ikakve prethodne intervencije. Programi za particionisanje diskova u nekim distribucijama (SuSE Linux Proffesional 9) mogu čak menjati veličine NTFS particija bez opasnosti od gubitka podataka.

- nfs - UNIX mrežni sistem datoteka koji omogućava deljenje lokalnog sistema datoteka između većeg broja umreženih računara i brz pristup udaljenim datotekama.
- smbfs - mrežni sistem datoteka koji omogućava deljenje lokalnog sistema datoteka sa umreženim računarima koji rade pod Windows operativnim sistemom. Koristi Windows protokol za deljenje datoteka.

Sistemi datoteka sa dnevnikom transakcija (Journaling)

Prilikom podizanja operativnog sistema proverava se integritet sistema datoteka. Gubitak integriteta najčešće se javlja kao posledica nasilnog zaustavljanja sistema, odnosno promena u objektima sistema datoteka koje nisu blagovremeno ažurirane u tabeli indeksnih čvorova, i može za posledicu imati gubitak podataka. Opasnost od gubitka podataka umanjuje se uvođenjem dnevnika transakcija koji prati aktivnosti vezane za promenu meta-data oblasti, odnosno i-node tabele, i objekata sistema datoteka. Dnevnik (journal, log) se ažurira pre promene sadržaja objekata i prati relativne promene u sistemu datoteka u odnosu na poslednje stabilno stanje. Transakcija se zatvara po obavljenom upisu i može biti ili u potpunosti prihvaćena ili odbijena. U slučaju oštećenja, izazvanog npr. nepravilnim gašenjem računara, sistem datoteka se može lako rekonstruisati povratkom na stanje poslednje prihvaćene transakcije.

Novije verzije Linux kernela uključuju podršku za rad sa visokoperformansnim journaling sistemima datoteka, poput ext3, ReiserFS, XFS i JFS sistema datoteka.

Ext3 sistem datoteka i režimi vođenja dnevnika transakcija

Sistem datoteka ext3 predstavlja ext2 sistem datoteka nadograđen podrškom za journaling. Jedan od osnovnih koncepata u dizajnu je potpuna kompatibilnost sa prethodnom verzijom - prelazak sa ext2 na ext3 ostvaruje se jednostavnim kreiranjem dnevnika, koji vodi evidenciju o izvršenim transakcijama, čime se umanjuje opasnost od gubitka podataka. Dnevnik transakcija umanjuje performanse sistema, pri čemu je pad performansi pri praćenju celokupne aktivnosti u sistemu datoteka znatno veći od pada performansi pri praćenju aktivnosti meta-data oblasti. Ext3 se može koristiti ukoliko je kernel preveden sa podrškom za ext3, što je podrazumevano u većini novijih distribucija Linuxa: Red Hat 7.2 i novije, SuSE 7.3 i novije. Tvorac ext3 sistema datoteka je Dr Stephen Tweedie.

U ext3 sistemu datoteka prisutna su tri režima vođenja dnevnika transakcija: journal, ordered i writeback.

- Journal je režim praćenja svih promena u sistemu datoteka, kako u meta-data oblasti tako i u objektima-datotekama, čime se pouzdanost sistema datoteka znatno uvećava na račun performansi. Redundansa koju ovaj režim rada unosi je velika, ali svakako na račun performansi.

- Ordered je režim praćenja promena u meta-data oblasti, pri čemu se promene u objektima sistema datoteka upisuju pre ažuriranja i-node tabele. Ovo je podrazumevani režim rada dnevnika, koji garantuje potpunu sinhronizaciju objekata sistema datoteka i meta-data oblasti. U odnosu na journal, ovaj režim karakteriše manja redundansa i veća brzina rada.
- Writeback je režim praćenja promena u meta-data oblasti, pri čemu se i-node tabela može ažurirati pre upisa promena u objekte sistema datoteka. Ovo je najbrži režim rada ali ne garantuje konzistenciju meta-data oblasti, odnosno sinhronizaciju objekata sistema datoteka meta-data oblasti, što može dovesti do neprijatnih situacija u sistemu datoteka kao što su pojave nove i stare verzije datoteke itd.

Reiser FS

Jedan od prvih sistema datoteka sa "journaling" opcijom je ReiserFS, verzija 3.6.x (prisutna u Linux kernelima počev od verzije 2.4). ReiserFS, koji je ime dobio po tvorcu, Hansu Reiseru, značajno povećava performanse pri radu sa malim datotekama (small file performance), koje su kod ostalih journaling sistema datoteka veoma slabe. Brojni testovi pokazuju da je ReiserFS 8 do 15 puta brži od ext2 pri radu sa datotekama manjim od 1KB. Dodatno, ReiserFS razrešava problem interne fragmentacije čime se povećava efikasnost iskorišćenja diskova. Ovako visoke performase pri radu sa malim datotekama ReiserFS postiže na osnovu optimizovanog B+ stabla (jedno po sistemu datoteka) i dinamičke alokacije indeksnih čvorova (za razliku od fiksne alokacije i-node koju koristi ext2). Dodatno, ReiserFS koristi promenljivu veličinu sistemskog bloka, a male datoteke se upisuju u svoj direktorijum zajedno sa svojom file-info strukturom. U dnevniku se ažuriraju samo promene u meta-data oblasti.
Loše osobine ReiserFS reflektuju se pri radu sa šupljim datotekama (sparse files), gde je ext2 daleko bolji. Takođe, ReiserFS radi sporije sa velikim datotekama u odnosu na ext2.

XFS

Sistem datoteka XFS je originalni proizvod kompanije Silicon Graphics, Inc (SGI), razvijen početkom devedesetih godina. Predstavlja robusni, puni 64-bitni sistem datoteka, sa brojnim kvalitetnim osobinama, prvenstveno namenjen za SGI IRIX, ali je napravljena varijanta i za Linux.

Prva novina u dizajnu XFS sistema datoteka je uvođenje alokacionih grupa, odnosno linearnih regiona jednake veličine, koji se definišu za svaki disk. Svaka alokaciona grupa ima svoju i-node tabelu i listu slobodnog prostora. Alokacione grupe su nezavisne i mogu učestvovati u paralelnim I/O operacijama i na taj način se omogućavaju istovremene paralelne I/O operacije na istom sistemu datoteka.

Interno, svaka alokaciona grupa koristi efikasna B+ stabla koja čuvaju informaciju o zonama slobodnog prostora i o slobodnim i-node čvorovima. XFS optimizuje alokaciju slobodnog prostora (koja je kritična po pitanju performansi upisa) putem odložene alokacije (delayed allocation).

Kao i ReiserFS, XFS koristi journal tehniku samo za meta podatke, ali journal može biti realizovan na više načina, obično kao poseban sistem datoteka.

Na bazi testova iz otvorene literature mogu se izvesti zanimljivi zaključci. XFS uglavnom pobeđuje u radu sa velikim datotekama, ali u testovima sa intenzivnim brisanjem datoteka pobeđuju ga i ReiserFS i ext3. Generalno gledano, teško je odrediti koji je sistema datoteka najbolji, zato što performanse drastično zavise od vrste test opterećenja.

JFS

JFS je puni 64-bitni "journaling" sistem datoteka koji je namenjen prvenstveno za IBM servere, ali je takođe portiran i za Linux operativni sistem. Kao i svi journaling sistemi obezbeđuje visoku pouzdanost, brz oporavak sistema datoteka i visoke performanse. Po pitanju journaling tehnike JFS upisuje u log samo metadata podatke.

JFS je extent baziran sistem datoteka, što mu omogućava da fleksibilno manipuliše datotekama na disku. Ekstent je sekvenca kontinualnih blokova koji se dodeljuju datoteci i koju specificiraju tri parametra <logical offset, length, physical>. Stablo je bazirano na B+ strukturi. JFS koristi različite veličine za sistemske blokove (512, 1024, 2048 i 4096 bajtova), što omogućava prilagođenje sistema datoteka prema potrebama.

Indeksni čvorovi i liste slobodnih blokova se održavaju dinamički, što svakako ima velike prednosti. Što se tiče direktorijuma, postoje dve moguće organizacije. Prva šema se koristi za male direktorijume i kod nje se sadržaj direktorijuma upisuje u njegov i-node, što drastično poboljšava performanse malih direktorijuma. Druga šema se koristi za velike direktorijume, koji se realizuju u formi optimizovanog B+ stabla, koje optimizuje pretraživanje, unos novih objekata i brisanje.

JFS se dobro snalazi i sa šupljim (sparse files) i sa gusto popunjenim datotekama (dense files). Kao puni 64-bitni sistem datoteka, odličan je u slučaju kada realizujete ogromne sisteme datoteka.

Koji sistem datoteka treba koristiti ?

Izbor sistema datoteka zavisi od faktora kao što su brzina, pouzdanost i kompatibilnost: ext2 i ext3 su stabilni domaći sistemi datoteka visokih performansi, reiser je vrlo pouzdan zahvaljujući dnevniku transakcija. Ovi sistemi datoteka nisu vidljivi iz lokalnog DOS/Windows sistema. Postojanje jedne FAT particije za razmenu podataka između Windows NT i Linux sistema je takođe dobro rešenje, ukoliko se koriste starije verzije kernela.

Kreiranje sistema datoteka

Sistemi datoteka se kreiraju, odnosno inicijalizuju pomoću programa mkfs. Program mkfs je front-end koji poziva posebne programe pomoću kojih se kreiraju različiti tipovi sistema datoteka. Sintaksa programa mkfs je različita za različite distribucije UNIX sistema, a najbitnije opcije, zajedničke za sve verzije programa mkfs, ovde su objašnjene. Dodatna uputstva o programu mkfs mogu se naći u on-line dokumentaciji (man pages).

Sintaksa programa mkfs je:

```
# mkfs [-t fstype] [-c | -l bblist] device
```

gde je:

device — specijalna datoteka koja predstavlja particiju na kojoj se kreira sistem datoteka. Tom datotekom se kasnije predstavlja i sistem datoteka.

-t fstype — opcija kojom se specificira tip sistema datoteka koji je potrebno kreirati. fstype može biti ext2, ext3, reiser, msdos ili bilo koji drugi tip za koji u operativnom sistemu postoji podrška.

-c — zastavica kojom se programu mkfs nalaže da pre kreiranja sistema datoteka ispita površinu medijuma na kojoj se kreira taj sistem datoteka i inicijalizuje listu neispravnih blokova.

-l bblist — opcija kojom se specificira datoteka sa inicijalnom listom neispravnih blokova. Ne treba koristiti opcije -c i -l zajedno.

Dati su primeri korišćenja komande mkfs. Prvi primer ilustruje kreiranje ext2 sistema datoteka na disketi kapaciteta 1.44MB. Disketa je najpre formatirana programom fdformat sa zastavicom -v (bez provere ispravnosti), a zatim je pomoću mkfs na disketi kreiran ext2 sistem datoteka i proverena površina medijuma. Primer ilustruje ponašanje programa mkfs - kako je kao parametar naveden tip ext2, mkfs je pokrenuo program mke2fs čija je isključiva namena kreiranje ext2 sistema datoteka.

```
# fdformat -v /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done

# mkfs -t ext2 -c /dev/fd0H1440
mke2fs 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
360 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group
Checking for bad blocks (read-only test): done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

Kreiranje sistema datoteka na particiji diska slično je prethodnom primeru, s tim što formatiranje najčešće nije potrebno, ali particionisanje jeste. U ovom slučaju je kreiranje liste neispravnih blokova komandom badblocks bolje rešenje od upotrebe zastavice -c programa mkfs. Lista se kasnije može iskoristiti prilikom provere integriteta sistema datoteka (program fsck). Dat je primer kreiranja ext2 sistema datoteka na drugoj primarnoj particiji Primary Master diska.

```
# badblocks /dev/hda2 > /tmp/bad-block-list1
# mkfs -t ext2 -l /tmp/bad-block-list1 /dev/hda2
```

Parametri UNIX sistema datoteka

Prilikom kreiranja moguće je navesti parametre sistema datoteka. Većinu parametara koje mkfs predlaže ne treba menjati, osim gustine indeksnih čvorova (broj i-nodova po jedinici prostora u sistemu datoteka) i broja cilindara u grupi (za ufs sisteme datoteka).

Svaka datoteka u sistemu datoteka koristi jedan indeksni čvor. Indeksni čvor (i-node) je struktura podataka koja potpuno opisuje jednu datoteku ili direktorijum i zauzima oko 128 bajtova prostora na disku. Broj indeksnih čvorova se ne menja dinamički - npr. ukoliko je na sistemu datoteka ostalo pet slobodnih i-node čvorova, korisnik će moći da kreira najviše pet novih datoteka, bez obzira na slobodni prostor. Sistem datoteka neće kreirati automatski još pet slobodnih i-node čvorova. Ako je prosečna veličina datoteke u sistemu datoteka 1MB ili više, standardna gustina indeksnih čvorova (1 i-node za svakih 4KB prostora) biće velika - veliki broj indeksnih čvorova ostaće neiskorišćen, čak i ako je sistem datoteka pun. Premašenje ovakvog sistema je veliko. S druge strane, ako se u sistemu datoteka nalazi veliki broj malih datoteka (npr. prosečne veličine 1-2KB), standardna gustina (1 i-node za svakih 4KB prostora) biće mala - u sistemu datoteka neće biti slobodnih indeksnih čvorova, bez obzira na slobodni prostor. Korisnici će ostati bez slobodnih i-node struktura pre nego što ostanu bez slobodnog prostora i neće moći da kreiraju nove datoteke. Zato je prilikom kreiranja sistema datoteka potrebno odrediti prosečnu veličinu datoteka na sistemu datoteka, a na osnovu toga gustinu i-node čvorova.

U ufs sistemima datoteka, datoteke se grupišu u grupe cilindara radi bržeg pristupa. Grupu cilindara čine 1-32 cilindra (podrazumevana vrednost je 16), pri čemu se brzina sistema smanjuje kada se broj cilindara u grupi povećava. Ako grupu čini 1 cilindar, sistem će raditi najbrže, ali će premašenje sistema biti najveće (svaka cilindarska grupa sadrži skup meta-data struktura uključujući backup kopiju superbloka). Povećanje broja cilindara u grupi smanjiće overhead, ali i brzinu sistema.

Veličina bloka za čitanje i pisanje kreće se u opsegu 1-8KB. Podrazumevana vrednost za Linux sisteme datoteka je 1KB. Veća vrednost uvećava performanse - broj indirektno adresiranih blokova podataka je manji, a kapacitet pročitan u jednom ciklusu ulazno-izlaznih operacija je veći. Manje sistemske blokove (512 bajta) treba koristiti na sistemima datoteka manjeg kapaciteta jer je i slack manji. Neke varijante UNIX sistema datoteka dozvoljavaju fragmentisanje blokova (block fragment) - jedan blok se deli na nekoliko fragmenata u koje se mogu smeštati manje datoteke ili poslednji fragmenti većih datoteka. Na ovaj način se smanjuje neiskorišćeni prostor na disku.

Parametri sistema datoteka podešavaju se pomoću argumenata programa kojim se kreira sistem datoteka (npr. mke2fs, mkdosfs), koji se mogu navesti i kao argumenti front-end programa mkfs. Parametri ext2 sistema datoteka podešavaju se pomoću sledećih argumenata programa mke2fs:

-b block-size	Specificira se veličina sistemskog bloka u bajtovima (blokovi mogu biti veličine 1024, 2048 and 4096 bajtova). Ako se ne navede, mke2fs određuje veličinu bloka na osnovu veličine sistema datoteka.
-f fragment-size	Specificira se veličina fragmenata (block-fragments) u bajtovima

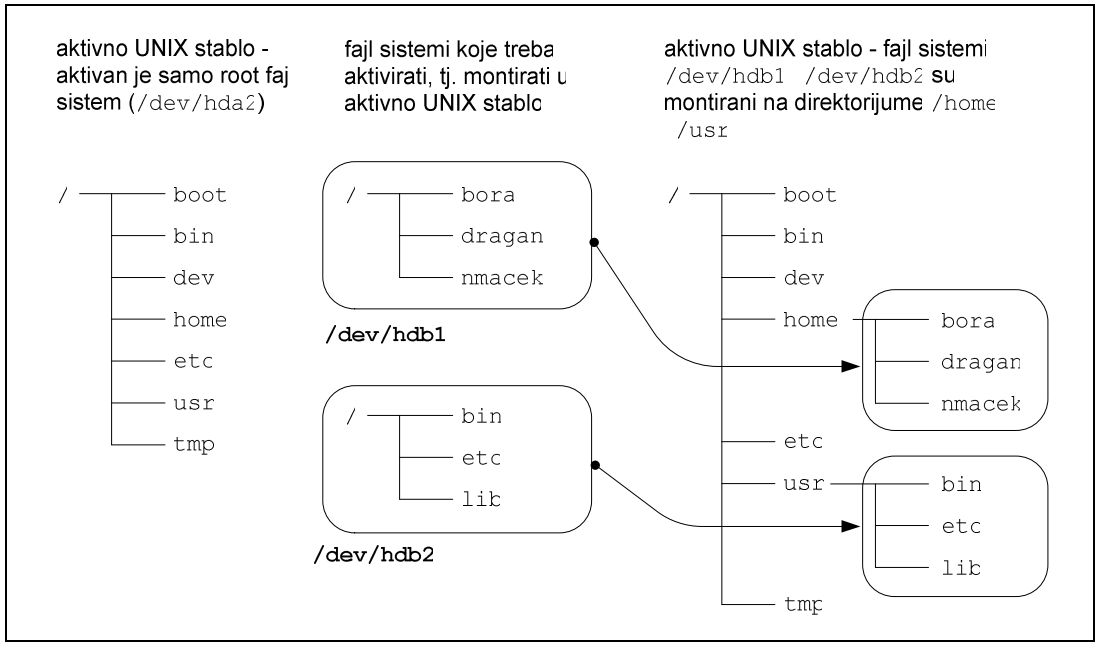
-i bytes-per-inode	Određuje na koliko će bajtova biti kreiran jedan i-node. Ova vrednost ne sme biti manja od veličine sistemskog bloka.
-N no-of-inodes	Specificira se tačan broj indeksnih čvorova koji će biti kreirani u i-node tabeli. Ova opcija nadjačava podrazumevanu vrednost za gustinu indeksnih čvorova i opciju -b.
-m percentage	Određuje procenat sistemskih blokova koji će biti rezervisani za root korisnika. Podrazumevana vrednost je 5%.

Aktiviranje i deaktiviranje sistema datoteka

Montiranje sistema datoteka na aktivno UNIX stablo

Pre korišćenja sistem datoteka treba aktivirati. Za razliku od DOS/Windows sistema, na kojima se svi FAT sistemi datoteka aktiviraju prilikom podizanja sistema, na UNIX-u se sistemi datoteka aktiviraju montiranjem na mount-point direktorijume. Montiranje sistema datoteka može se obaviti ručno (pomoću komande mount) ili automatski, prilikom podizanja sistema, a kasnije se sistem datoteka može deaktivirati komandom umount. Montiranjem sistema datoteka na mount-point direktorijume stvara se aktivno UNIX stablo koje čine svi aktivirani sistemi datoteka.

Na slici 3.6 data su tri sistema datoteka (npr. sistemi datoteka na particijama /dev/hda2, /dev/hdb1 i /dev/hdb2). Postupak ilustruje stvaranje aktivnog UNIX stabla - sistemi datoteka /dev/hdb1 i /dev/hdb2 montirani su na mount-point direktorijume /home i /usr root sistema datoteka /dev/hda2.



Slika 3.6 *Montiranje sistema datoteka*

Montiranje sistema datoteka sa slike 3.6 može se uraditi pomoću sledećih komandi:

```
# mount /dev/hdb1 /home
# mount /dev/hdb2 /usr
```

Nakon izvršenja ovih komandi sistemi datoteka su aktivirani, i korisnici mogu pristupati njihovom sadržaju preko direktorijuma /home i /usr aktivnog UNIX stabla. Na primer, listanje sadržaja root direktorijuma sistema datoteka /dev/hdb1 može se obaviti komandom ls /home.

Standardna sintaksa komande mount je:

```
# mount [-r] [-t fstype] devicenode mountpoint
```

Nakon izvršenja komande kernel će montirati sistem datoteka koji se nalazi na uređaju ili particiji čiji je nod devicenode na direktorijum mountpoint. Direktorijum mountpoint mora biti kreiran u aktivnom UNIX stablu pre montiranja sistema datoteka i preporučljivo je da bude prazan. Ukoliko nije prazan, prethodni sadržaj, vlasnik i pristupna prava biće nevidljivi za korisnike sve dok je montirani sistem datoteka aktivan. Putanja direktorijuma mountpoint postaće putanja ka root direktorijumu montiranog sistema datoteka.

Komanda mount zahteva da se navedu dva argumenta:

- nod za uređaj (npr.CD-ROM ili flopi) ili particiju diska (primarnu ili logičku) na kojoj se nalazi sistem datoteka - u literaturi se spominje i kao nod za sistem datoteka,
- direktorijum na koji će se montirati sistem datoteka (mount-point).

Prilikom aktiviranja UNIX će pokušati da sam prepozna tip sistema datoteka. Ukoliko u tome ne uspe moguće je pomoću parametra -t eksplicitno navesti tip sistema datoteka. Na primer, montiranje diskete na kojoj se nalazi FAT sistem datoteka može se uraditi sledećom komandom:

```
# mount -t msdos /dev/fd0 /media/floppy
```

Bez parametra -r komanda mount će pokušati da aktivira sistem datoteka za čitanje i pisanje, ukoliko za to postoji podrška u kernelu (npr. starije verzije kernela neće dozvoliti aktiviranje ntfs sistema datoteka u režimu čitanja i pisanja). Ako se sistem datoteka nalazi na medijumu kao što je CD-ROM ili postoji potreba da se zabrani pisanje na sistem datoteka, sistem datoteka se pomoću opcije -r može aktivirati samo u režimu čitanja (read-only). Kernel će tada zaustaviti sve pokušaje pisanja na sistem datoteka i neće menjati vreme zadnjeg pristupa u indeksnim čvorovima datoteka koje se nalaze na tom sistem datoteka.

root i user sistemi datoteka

Svi objekti aktivnih sistema datoteka organizovani su u jedinstvenu strukturu stabla - aktivno UNIX stablo. Root sistem datoteka (/) nastaje prilikom instalacije operativnog sistema. To je prvi sistem datoteka koji se aktivira prilikom podizanja sistema - montira se na root direktorijum aktivnog UNIX stabla, i u toku rada se ne može deaktivirati. Sistem se ne može podići ako root sistem datoteka ne može da se aktivira. Da bi kernel znao gde se root sistem datoteka nalazi, nod root sistema datoteka treba specificirati u boot manageru

(kao što su LILO i GRUB). Alternativno rešenje je da se u izvornom kodu (source) kernela navede putanja root sistema datoteka, pa da se kernel prevede (compile). Root sistem datoteka se prvo aktivira u read-only režimu da bi se pomoću programa fsck proverio integritet sistema datoteka, a zatim se ponovo aktivira u režimu čitanja i pisanja. Na root sistemu datoteka nalaze se sistemski podaci i njegova struktura je strogo određena. Modifikovanje ove strukture može biti vrlo destruktivan proces (npr. brisanje direktorijuma /bin znači brisanje velikog broja komandi sa sistema, a promenom imena direktorijuma /etc dobija se sistem bez konfiguracionih datoteka). Zbog toga obični korisnici imaju redukovani pristup root sistemu datoteka - čitanje datoteka i navigacija po stablu. Pun pristup ima samo superuser root.

User (korisnički) sistem datoteka se po potrebi može aktivirati i deaktivirati u toku rada, ukoliko je korisnik za to dobio dozvole od superusera. Na user sistemu datoteka nalaze se korisnički podaci, njegovu strukturu određuju korisnici i ona se kasnije može menjati. Pristupna prava u okviru user sistema datoteka određuje root.

/etc/fstab i auto-mount

Sistemi datoteka koji se aktiviraju automatski prilikom podizanja operativnog sistema (auto-mount) opisani su u datoteci /etc/fstab (filesystem table). Ova datoteka sadrži statičke informacije o sistemu datoteka. Dat je primer datoteke /etc/fstab:

```
# cat /etc/fstab
LABEL=/         /           ext3      defaults,usrquota       1 1
LABEL=/boot     /boot       ext3      defaults                1 2
none            /dev/pts    devpts    gid=5,mode=620          0 0
LABEL=/home     /home       ext3      defaults,usrquota       1 2
/dev/sda3       /shares     ext3      defaults                1 2
none            /proc       proc      defaults                0 0
none            /dev/shm    tmpfs     defaults                0 0
LABEL=/usr      /usr        ext3      defaults,usrquota       1 2
LABEL=/var      /var        ext3      defaults,usrquota       1 2
/dev/sda2       swap        swap      defaults                0 0
/dev/cdrom      /cdrom      iso9660   noauto,owner,kudzu,ro   0 0
/dev/fd0        /floppy     auto      noauto,owner,kudzu      0 0
```

Svaki sistem datoteka opisan je jednom linijom datoteke. Svaka linija datoteke ima šest polja, odvojenih razmaknicama ili tab karakterima. Navodimo redom značenje ovih polja.

- Prvo polje (fs_spec) opisuje sisteme datoteka koje treba aktivirati. Lokalni sistemi datoteka opisani su nodovima za diskove i izmenljive medijume (/dev/sda2, /dev/cdrom) ili imenom sistema datoteka (LABEL=/home), a mrežni sistemi datoteka imenom servera i imenom deljenog direktorijuma (npr. kyuss:misc_docs).
- Drugo polje (fs_file) opisuje mount-point direktorijum sistema datoteka. Za swap u ovo polje treba upisati swap ili none.
- Treće polje (fs_vfstype) opisuje tip sistema datoteka - npr. ext2, ext3, iso9660, swap. Za izmenljive medijume koji podržavaju rad sa nekoliko tipova sistema datoteka (kao što su flopi diskovi) u ovo polje treba upisati auto - kada se sistem datoteka sa izmenljivog medijuma aktivira, izvršiće se automatska detekcija tipa sistema datoteka.

- U četvrtom polju (fs_mntops) navedene su zapetom razdvojene opcije sa kojima će se sistem datoteka aktivirati. Opcija noauto zabraniće aktiviranje datog sistema datoteka prilikom podizanja operativnog sistema i aktiviranje sistema datoteka komandom mount -a. Sistem datoteka sa opcijom noauto može se aktivirati samo ručno. Opcija user dozvoliće svim korisnicima da aktiviraju taj sistem datoteka, što inače može da obavi samo superuser root. Ako je navedena opcija usrquota, sistem datoteka biće aktiviran sa limitom iskorišćenja prostora na disku (user quota). Ako je navedena opcija kudzu, sistem datoteka biće uključen u listu uređaja na kojima se vrši pretraživanje drajvera za plug-n-play uređaje. Opcija ro dozvoliće aktiviranje sistema datoteka isključivo u režimu čitanja.
- Peto polje (fs_freq) određuje da li će sistem datoteka biti uključen u listu sistema datoteka za back-up, odnosno dump (vrednost polja je 1) ili ne (vrednost polja je 0)
- Šesto polje (fs_passno) opisuje red kojim će program fsck proveriti integritet sistema datoteka pri podizanju operativnog sistema. Za root sistem datoteka fs_passno je 1, a za ostale sisteme datoteka, 2. Pri tom, sistemi datoteka na istom uređaju biće provereni sekvencijalno, a sistemi datoteka na različitim uređajima paralelno, radi uštede vremena. Ako je fs_passno 0, integritet sistema datoteka neće biti proveren.

Drugi način za aktiviranje svih sistema datoteka opisanih u /etc/fstab je komanda mount -a. Ovom komandom će svi sistemi datoteka opisani u /etc/fstab, koji nemaju atribut noauto, biti montirani na mount-point direktorijume navedene u drugoj koloni iste datoteke.

Pregled aktiviranih sistema datoteka

Svaki korisnik UNIX sistema može pomoću komande mount bez argumenata odrediti koji su sistemi datoteka aktivirani na sistemu, odnosno šta je montirano na aktivno UNIX stablo. Komanda mount ove podatke čita iz datoteke /etc/mtab (mount table). Na osnovu toga može se odrediti:

- broj i vrsta diskova na sistemu,
- kako su diskovi particionisani,
- na kom se disku nalazi root sistem datoteka,
- koji su operativni sistemi prisutni na računaru .

Ova interpretacija ima smisla samo ako su svi sistemi datoteka aktivirani. Ukoliko ima neaktinih sistema datoteka običan korisnik ne može na osnovu rezultata komande mount odrediti tačan broj diskova u sistemu, ne može tačno odrediti kako su diskovi particionisani, i koji se sistemi datoteka na njima nalaze. Dodatne informacije o sistemima datoteka mogu se dobiti na osnovu datoteke /etc/fstab, koju svi korisnici mogu da pročitaju. Dodatne informacije o particionisanju diskova (pomoću programa fdisk -l) mogu dobiti samo korisnici koji imaju prava čitanja nad direktorijumom /sbin, programom /fdisk i nodovima za diskove i particije na njima - /dev/hda, /dev/hda1 itd.)

Dodatno, ukoliko korisnik ima pravo da pročita nod sistema datoteka, komanda mount -l prikazaće i njegovo ime - label (samo za ext2, ext3 i XFS). Imena sistema datoteka mogu se zadavati pomoću specijalnih komandi koje pripadaju setovima programa za administraciju različitih tipova sistema datoteka (npr. ime ext2 sistema datoteka zadaje se pomoću komande e2label, a ime XFS sistema datoteka pomoću komande xfs_admin.)

U sledećem primeru dat je rezultat izvršenja komande mount na jednom UNIX serveru:

```
# mount -l
/dev/sdb3 on / type ext3 (rw,usrquota) [/]
none on /proc type proc (rw)
/dev/sda1 on /boot type ext3 (rw) [/boot]
none on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/sdb2 on /home type ext3 (rw,usrquota) [/home]
/dev/sda3 on /shares type ext3 (rw) []
none on /dev/shm type tmpfs (rw)
/dev/sdb1 on /usr type ext3 (rw,usrquota) [/usr]
/dev/sdb5 on /var type ext3 (rw,usrquota) [/var]
```

Na osnovu ovoga korisnici mogu biti sigurni:

- da na serveru postoje najmanje dva SCSI diska, pri čemu se ne može odrediti da li postoje i IDE diskovi,
- da je prvi SCSI disk podeljen na najmanje tri particije, pri čemu se ne može odrediti da li na njemu postoje extended i logičke particije; da je drugi SCSI disk podeljen na najmanje tri particije, od kojih je jedna sigurno extended, i da u toj particiji postoji najmanje jedna logička particija,
- da se root sistem datoteka nalazi u trećoj primarnoj particiji drugog SCSI diska,
- da je na računaru prisutan Linux, pri čemu se prisustvo ostalih operativnih sistema ne može odrediti (npr. ne postoje FAT/NTFS sistemi datoteka koji ukazuju na moguće postojanje DOS/Windows operativnih sistema, niti HPFS sistemi datoteka koji ukazuju na moguće postojanje OS/2 operativnog sistema).

Deaktiviranje sistema datoteka

Za razliku od root sistema datoteka, koji se aktivira pri podizanju operativnog sistema i koji se kasnije ne može deaktivirati, korisnički (user) sistemi datoteka se po potrebi može deaktivirati, odnosno demontirati sa aktivnog UNIX stabla. Deaktiviranje sistema datoteka vrši se komandom umount, koja zahteva da se navede jedan argument: nod za sistem datoteka ili mount-point direktorijum.

Tako se, na primer, deaktiviranje sistema datoteka /dev/hdb3 koji je montiran na direktorijum /inst_packages može izvršiti komandom:

```
# umount /dev/hdb3
```

ili:

```
# umount /inst_packages
```

Pre zamene medijuma u uređajima poput flopi-diska (uređaji sa izmenljivim medijumima), potrebno je deaktivirati sistem datoteka na njima. CD-ROM ne dozvoljava da se medijum izbaci iz uređaja dok se sistem datoteka na njemu ne deaktivira, ali kao read-only uređaj ne može da ošteti medijum (osim fizički) u slučaju nasilne izmene. Ovakva vrsta zaštite ne postoji na flopi diskovima koji dozvoljavaju pisanje po izmenljivim medijumima. Ako se disketa izvadi iz flopi diska bez prethodnog deaktiviranja sistema datoteka, može doći do logičkog oštećenja medijuma. Logičko oštećenje medijuma nastaje kao posledica informacija koje se čuvaju u write-behind kešu - sistem u potpunosti prazni write-behind keš prilikom deaktiviranja sistema datoteka. Bez deaktiviranja, velika je verovatnoća da deo informacija neće biti upisana na medijum, koji na taj način može postati čak i neupotrebljiv.

Dozvole za aktiviranje sistema datoteka

Procedure aktiviranja i deaktiviranja sistema datoteka zahtevaju privilegije superusera. Razlog je u sledećem: korisnik root je sistemski korisnik, odnosno deo sistema, i kao takvom data su mu apsolutna prava. Pristup root nalogu imaju osobe od poverenja, sa višegodišnjim iskustvom u administraciji UNIX sistema. Ostali korisnici se ne smatraju delom sistema, ne može im se ukazati visok nivo poverenja i najčešće nemaju dovoljno iskustva u administraciji.

Ukoliko postoji potreba regularnim korisnicima može se omogućiti aktiviranje sistema datoteka:

- davanjem lozinke superusera. Ovo je najlakše rešenje i treba ga praktikovati samo ako su mere sigurnosti i kontrole pristupa nepotrebne, što je slučaj sa neumreženim personalnim računarima;
- korišćenjem programa sudo (konfiguraciona datoteka programa sudo je /etc/sudoers);
- korišćenjem paketa mtools koji korisnicima omogućava rad sa MS-DOS disketama bez montiranja na aktivno UNIX stablo;
- davanjem dozvola svim korisnicima za montiranje sistema datoteka u datoteci /etc/fstab. Korisnici će moći da aktiviraju samo one sisteme datoteka za koje je u polju fs_mntops navedena opcija user.

Provera i oporavak sistema datoteka

Sistem datoteka je kompleksna struktura podataka, koju osim podataka (datoteka i direktorijuma) čine i strukture za organizaciju podataka na medijumu: zaglavlje (superblock) i meta-data area u vidu i-node tabele. Fizički defekti, odnosno neispravnost površine diska mogu da prouzrokuju nepovratan gubitak podataka. Preliminarna sigurnost se postiže prijavljivanjem defekata programu mkfs prilikom kreiranja sistema datoteka. Naknadno je potrebno proveravati površinu diska programom badblocks, i ažurirati listu neispravnih blokova u okviru sistema datoteka programom fsck. Logički defekti se odnose na neispravnosti u zaglavlju sistema datoteka i i-node tabeli, i najčešće nastaju kao posledica keširanja diskova, odnosno kao posledica informacija koje se čuvaju u write-

behind kešu. Ukoliko sistem u potpunosti ne isprazni write-behind keš (sistem u potpunosti prazni write-behind keš u periodima slabe aktivnosti diska i prilikom deaktiviranja sistema datoteka), deo informacija neće biti upisan na medijum. Ovakve situacije se javljaju prilikom nasilnog zaustavljanja sistema (nestanak struje, nasilni reboot).

U oba slučaja posledice direktno zavise od tipa informacije koja se nalazi na neispravnom bloku, odnosno od tipa informacije u write-behind kešu koja nije upisana na disk. Ukoliko se radi o podacima iz zaglavlja sistema datoteka, sistem datoteka gubi integritet, a postoji mogućnost i da će ceo sistem datoteka biti neupotrebljiv. Ukoliko se radi o podacima koji pripadaju i-node tabeli, korisnik neće moći da pristupi datoteci ili direktorijumu, iako sami podaci postoje na disku. Ukoliko se radi o regularnim datotekama delovi datoteka mogu biti nepovratno oštećeni. Pouzdanost sistema datoteka može se povećati korišćenjem dnevnika transakcija, naprednih tehnika kao što su disk mirroring (RAID level 1) i izvora neprekidnog napajanja (Uninterruptible Power Supply - UPS).

Logički defekti i provera integriteta sistema datoteka

Integritet sistema datoteka može se proveriti pomoću programa fsck (filesystem check). fsck će otkloniti manje "kvarove" u sistemu datoteka, i upozoriti korisnika na postojanje problema koji se ne mogu otkloniti. U najčešće uzroke logičkih kvarova sistema datoteka koji se mogu otkloniti spadaju nestanak struje i korisničke greške (npr. neispravno gašenje sistema). Posledice kvara na samom hardveru (na disku ili disk kontroleru) najčešće se ne mogu sanirati - verovatnoća gubitka podataka u slučaju otkaza hardvera je velika. Sisteme datoteka sa logičkim defektima treba aktivirati samo u režimu čitanja, npr. u svrhe pravljenja backupa. Aktiviranje neispravnih sistema datoteka u režimu za čitanje i pisanje može biti destruktivno - pri upisu podataka na sistem datoteka izmeniće se meta-data strukture koje su već neispravne, i na taj način može doći do dodatnog gubitka podataka.

fsck se automatski pokreće prilikom podizanja većine UNIX sistema - na taj način se neispravnosti u auto-mount sistemima datoteka detektuju i otklanjaju pre korišćenja. Vreme potrebno za proveru integriteta raste sa veličinom sistema datoteka. Kako se logički defekti retko javljaju na sistemima datoteka koji su deaktivirani pre gašenja sistema, UNIX koristi sledeće mehanizme pomoću kojih smanjuje vreme potrebno za podizanje sistema:

- ukoliko postoji datoteka /etc/fastboot, ni jedan sistem datoteka se ne proverava prilikom podizanja sistema,
- u superbloku UNIX native sistema datoteka uveden je fleg čistoće (dirty-flag) koji se resetuje pri svakom deaktiviranju sistema datoteka, i pomoću kog se određuje koji su sistemi datoteka ispravni, a koje treba automatski proveriti prilikom sledećeg podizanja UNIX sistema,
- integritet journaling sistema datoteka se ne proverava.

Integritet ostalih sistema datoteka korisnici mogu proveriti pokretanjem programa fsck. Program fsck zaobilazi standardne rutine za pristup sistemima datoteka i tretira disk kao uređaj, tako da sistem datoteka čiji se integritet provera ne sme biti aktiviran.

```
# fsck /dev/sdb2
```

```
fsck 1.27 (8-Mar-2002)
e2fsck 1.27 (8-Mar-2002)
/home: clean, 9380/126720 files, 213122/253023 blocks
```

lost+found direktorijum

Prilikom kreiranja novog sistema datoteka, mkfs kreira direktorijum lost+found. U ovaj direktorijum fsck smešta pokazivače ka i-node strukturama koje nisu evidentirane ni u jednom direktorijumu, ili na njih ukazuju oštećene dir-info strukture. Program fsck ne može da sazna pravo ime oštećenih objekata tako da im dodeljuje imena na osnovu broja indeksnog čvora, u obliku #inode-number. Na ovaj način se povećava stepen pouzdanosti sistema - datoteka čiji i-node nije evidentiran ni u jednom direktorijumu se ne briše, već se notira u lost+found direktorijumu, tako da se kasnije može upotrebiti ukoliko ima korisnog sadržaja.

Ukoliko direktorijum lost+found ne postoji fsck će prilikom provere integriteta obrisati sve objekte čiji i-node nije regularno evidentiran u nekom direktorijumu.

Korišćenje lost+found direktorijuma je poželjna mera predostrožnosti na sistemima datoteka koji ne podržavaju rad sa dnevnikom transakcija. Ukoliko takav sistem datoteka postoji, potrebno je nakon svakog nasilnog zaustavljanja sistema proveriti sadržaj lost+found direktorijuma i rekonstruisati korisne podatke. Kako se atributi objekata sistema datoteka (osim imena objekta) čuvaju u i-node strukturama, svi objekti u lost+found direktorijumu imaće originalnog vlasnika, originalna prava pristupa i ostale atribute.

Fizički defekti i provera ispravnosti površine diska

Komanda badblocks se može koristiti za proveru ispravnosti površine diska koju sistem datoteka zauzima. Kao rezultat izvršenja komande na ekranu se prikazuju brojevi svih neispravnih blokova diska koji se analizira - ova lista se kasnije može iskoristiti kao parametar programa fsck čime se lista neispravnih blokova aranžira u meta-data strukturi sistema datoteka. Operativni sistem dalje neće koristiti one blokove sistema datoteka koji se nalaze na neispravnim blokovima diska, čime se vraća pouzdanost sistemu datoteka. Sledeći primer ilustruje komande kojima se ova provera može izvesti:

```
# badblocks /dev/sda2 > bad-block-list
# fsck -t ext2 -l bad-block-list /dev/sda2
```

Defragmentacija

Datoteka se prilikom upisa u sistem datoteka ne može uvek smestiti u strogo sekvencijalni niz blokova. Takve datoteke se nazivaju fragmentisanim i za njihovo čitanje je potrebno više vremena, s obzirom da je potrebno više puta pomeriti glave za čitanje i pisanje na samom disku. Neki sistemi datoteka, uključujući ext2 i ext3, koriste napredne tehnike za umanjenje fragmentacije. Osnovna ideja tehnike koju ext2 sistem datoteka primenjuje je sledeća: prilikom upisa datoteka uvek se kao sledeći blok za upis bira blok najbliži bloku na kom se trenutno vrši upis, tako da se smanjuje korišćenje mehanike diska, a samim tim i sistem datoteka brže radi. Fragmentacija predstavlja problem na sistemima sa lošom

tehnikom keširanja diskova, a kako se teško može izbeći, mora biti svedena na neki prihvatljivi minimum.

Ovaj problem se rešava defragmentacijom sistema datoteka, odnosno aranžiranjem datoteka u sistemu datoteka u kontinualne sekvence blokova. Većina sistema datoteka koristi specifične programe za defragmentaciju (za defragmentaciju ext2 sistema datoteka koristi se program defrag). Pre defragmentacije bilo kog sistema datoteka preporučuje se izrada backupa.

Ostali programi za rad sa sistemima datoteka

Od ostalih programa koji služe za rad sa sistemima datoteka pomenućemo samo najbitnije, koji se mogu koristiti na svim tipovima sistema datoteka i neke koji su specijalno namenjeni za rad sa ext2 sistemima datoteka.

Program df (disk free) prikazuje količinu slobodnog prostora na jednom ili više sistema datoteka, izraženu u sistemskim blokovima. Ukoliko se navede s parametrom -h, količina slobodnog prostora se prikazuje u čitljivom formatu (human readable format), odnosno u KB ili MB.

```
# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sdb3             372M   34M  319M  10% /
/dev/sda1              45M  7.4M   35M  17% /boot
/dev/sdb2             973M   24M  899M   3% /home
/dev/sdb1             2.6G  440M  2.0G  18% /usr
/dev/sdb5             251M   98M  139M  42% /var
/dev/sda3             1.7G  1.4G  268M  85% /shares
```

Program du (disk used) prikazuje količinu prostora na sistemu datoteka koju zauzimaju direktorijumi, počev od tekućeg direktorijuma ili od direktorijuma koji je naveden kao parametar. Na ovaj način mogu se pronaći korisnici čije datoteke zauzimaju najviše prostora na sistemu datoteka. Ukoliko se navede s parametrom -h količina zauzetog prostora se prikazuje u čitljivom formatu (human readable format), odnosno u KB ili MB.

```
# du -h /home
16k     /home/lost+found
24k     /home/nmacek
7.0M    /home/dragan
24k     /home/bora
7.1M    /home
```

Komanda sync upisuje u sistem datoteka sve neupisane blokove koji se nalaze u bafer kešu. Ovu komandu je retko potrebno pokrenuti zato što je u sistemu aktivan specijalan demon proces kojim je automatizovana sinhronizacija keša. Komanda se pokreće ukoliko je sistem potrebno odmah oboriti, čime se keš potpuno sinhronizuje sa sistemom datoteka, tako da su pri sledećem podizanju Linux sistema sistemi datoteka "čisti".

Program tune2fs služi za podešavanje parametara ext2 sistema datoteka. U parametre koji se ovim programom mogu podesiti spadaju: ime sistema datoteka (label), broj blokova rezervisanih za superusera, najduže vreme između dve provere integriteta (interval between checks), najveći dozvoljeni broj aktiviranja sistema datoteka između dve provere

integriteta (max mount count), ponašanje kernela ukoliko se pri radu sa sistemom datoteka detektuje greška (error behavior) i parametri vezani za korišćenje dnevnika transakcija (journal).

Program dumpe2fs prikazuje informacije o ext2 sistemu datoteka na osnovu podataka koje čita iz zaglavlja, odnosno superbloka. Program funkcioniše na principu Berkeley dumpfs programa koji prikazuje informacije o BSD FFS sistemu datoteka.

```
# dumpe2fs -h /dev/sda1
dumpe2fs 1.27 (8-Mar-2002)
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          64690278-d43b-4faf-b198-318a458e29cc
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal filetype needs_recovery
sparse_super
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              12048
Block count:              48163
Reserved block count:     2408
Free blocks:              39140
Free inodes:              12022
First block:              1
...
```

debugfs je interaktivni program koji korisniku omogućava direktan pristup strukturama sistema datoteka, tako da se može koristiti za njihov oporavak, ukoliko fsck to ne može da uradi automatski. debugfs omogućava korisniku da izvrši razne operacije nad objektima sistema datoteka, kao i nad sistemom datoteka u celini, poput markiranja zastavice čistoće, inicijalizacije, povezivanja i-node i dir-info struktura, i povratka obrisanih datoteka. Detaljan spisak komandi debugfs programa dobija se unošenjem komande help.

```
# debugfs
debugfs 1.27 (8-Mar-2002)
debugfs:  help
Available debugfs requests:

show_debugfs_params, params   Show debugfs parameters
open_filesys, open            Open a filesystem
close_filesys, close          Close the filesystem
feature, features             Set/print superblock features
dirty_filesys, dirty          Mark the filesystem as dirty
init_filesys                  Initalize a filesystem
show_super_stats, stats       Show superblock statistics
...
```

Rad sa diskovima bez sistema datoteka

Iako se diskovi i drugi masovni memorijski medijumi (na primer, CD-ROM) najčešće koriste u formi sistema datoteka, postoje izuzeci. Swap particija na sebi nema klasičan

sistem datoteka. Disketa se može iskoristiti kao medijum za arhiviranje, pri čemu tar (tape arhiver) sadržaj upisuje direktno na disketu, zaobilazeći rutine za rad sa sistemima datoteka. Linux boot disketa najčešće ne sadrži sistem datoteka, već se na njoj nalazi minimalni kernel u specijalnom formatu. Hard diskovi se takođe mogu koristiti u ovom obliku. Na primer, ako je jedan disk u sistemu oštećen, poželjno je napraviti kopiju istog na drugom disku pre pokušaja oporavka. Kopija se može napraviti komandom dd koja pristupa medijumu zaobilazeći sistem datoteka. Na primer:

```
# dd if=/dev/hdb of=/dev/hdc
```

Korišćenje medijuma bez sistema datoteka ima svoje prednosti: nema gubitaka koje unose strukture za organizaciju podataka, tako da se veći deo medijuma može iskoristiti za čuvanje korisničkih podataka. Takođe, kompatibilnost medijuma je uvećana - tar format disketne arhive je identičan na većini operativnih sistema, dok se domaći sistemi datoteka uglavnom razlikuju.

Komanda dd (disk-to-disk copy)

Komanda dd konvertuje i kopira datoteku ili određeni deo medijuma. Koristi se uglavnom za kopiranje disketa i traka ili za konvertovanje sadržaja datoteke. Komanda dd pristupa fizičkom medijumu bez obzira na njegov sadržaj (zaobilazi sistem datoteka), čime je omogućeno kopiranje medijuma koji nisu ni u jednom od formata koje UNIX prepoznaje. Sintaksa komande dd je:

```
# dd [option=value]
```

pri čemu se komanda može zadati sa sledećim parametrima:

if=input_file	dd ulazne podatke čita iz datoteke input_file
of=output_file	dd podatke upisuje u datoteku output_file
bs=n	specificira se veličina ulaznog i izlaznog bloka u bajtima (block size)
ibs=n	specificira se veličina ulaznog bloka u bajtima
obs=n	specificira se veličina izlaznog bloka u bajtima
cbs=n	specificira se veličina bloka za konverziju u bajtima
skip=n	broj blokova koji se preskaču pre kopiranja na ulazu
seek=n	broj blokova koji se preskaču pre kopiranja na izlazu
count=n	specificira se broj blokova koje dd treba da iskopira
conv=t1[,t2,..]	Konverzije datoteka u ascii ili ebcidic karakter set, u mala slova (lcase), velika slova (ucase) i obrtanje bajtova – bswap (byte swap)

Broj blokova i bajtova može biti naveden sa nekim od sledećih multiplikativnih sufiksa: kD (1000), k (1024), MD (1,000,000), M (1,048,576), GD (1,000,000,000), G (1,073,741,824).

U nastavku teksta dat je primer koji pokazuje jednostavan način kopiranja diskete. Najpre se sadržaj originalne diskete upiše u pomoćnu datoteku /tmp/floppydisk1, zatim se originalna disketa zameni praznom i u komandi zamene parametri za ulaznu i izlaznu datoteku (if i of). Za disketu od 1.44MB koristićemo veličinu bloka 96kB.

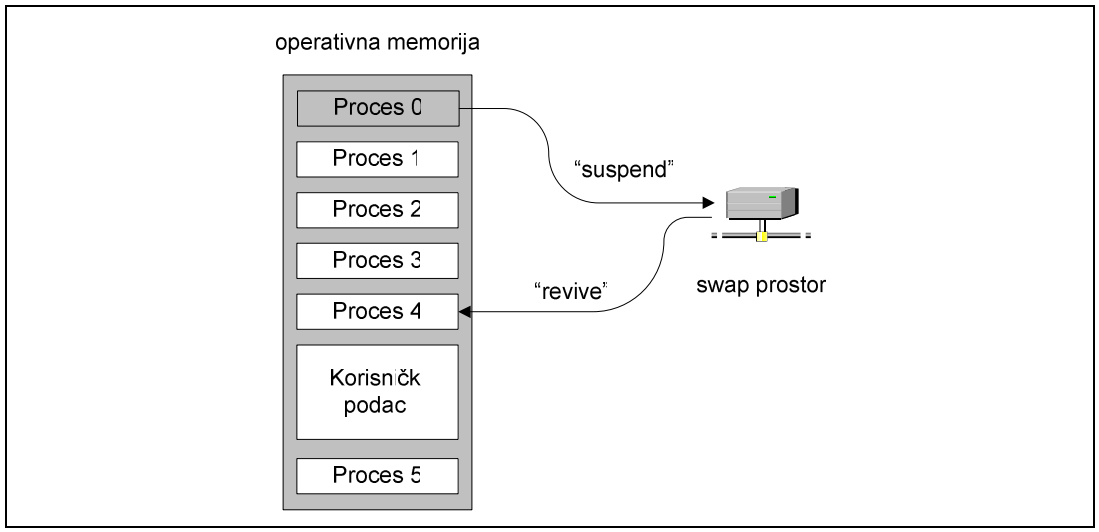
```
# dd if=/dev/fd0H1440 of=/tmp/floppydisk1 bs=96k
15+0 records in
15+0 records out
# dd if=/tmp/floppydisk1 of=/dev/fd0H1440 bs=96k
15+0 records in
15+0 records out
```

Provera ispravnosti sadržaja diskete bez montiranja na aktivno Linux stablo može se izvršti na sledeći način:

```
# dd if=dev/fd0H1440 of=/dev/null bs=96k
```

Virtuelna memorija (swap)

Virtuelna memorija (swap) je deo diska ili sistema datoteka koji služi za privremeno skladištenje neaktivnih procesa, čime se prividno povećava količina operativne memorije. Funkcionisanje swap prostora simbolički je prikazano na slici 3.7.



Slika 3.7 *Funkcionisanje swap prostora*

Swap se može realizovati na dva načina: u formi datoteka u postojećem sistemu datoteka i kao kvazi-sistem datoteka na posebnoj particiji. Obe varijante imaju svoje prednosti i svoje mane. Swap u formi datoteka omogućava veću fleksibilnost administracije sistema - veličina swap datoteke može se lako povećati, što u slučaju swap particije nije tako jednostavno. Swap prostor u formi posebne particije funkcioniše brže nego swap datoteka, jer se zaobilaze rutine za pristup objektima sistema datoteka.

Linux zahteva kreiranje jedne swap particije prilikom instalacije i najčešće sam preporučuje veličinu swap prostora. Preporučuje se kreiranje jedne swap particije dva puta veće od količine sistemske memorije, a swap prostor se naknadno može uvećati kreiranjem swap datoteka.

Swap particija se naknadno može kreirati pomoću programa fdisk ili nekog drugog programa za rad sa particijama (cfdisk, sfdisk), pri čemu kreirana particija treba da bude tipa 82 (Linux swap). Nakon toga je potrebno pokrenuti program mkswap koji će kreirati logičku strukturu swap prostora, i aktivirati swap prostor komandom swapon.

```
# mkswap /dev/sda2
# swapon /dev/sda2
```

Swap se po potrebi može isključiti komandom swapoff.

```
# swapoff /dev/sda2
```

Swap datoteka se kreira na sledeći način: najpre se u postojećem sistemu datoteka kreira prazna datoteka bez rupa željene veličine, a zatim se u okviru nje kreira logička struktura swap prostora. Swap datoteke se mogu aktivirati i deaktivirati na isti način kao i swap particije, komandama swapon i swapoff, pri čemu se kao parametri programa navode imena swap datoteka sa apsolutnom putanjom.

```
# dd if=null of=/swap_file count=100M
# mkswap /swap_file
# swapon /swap_file
```

Informacije o količini i zauzetosti operativne memorije i swap prostora mogu se dobiti pomoću programa free.

```
# free
             total       used       free     shared    buffers     cached
Mem:         62176      60160       2016          0      10876      18016
-/+ buffers/cache:      31268      30908
Swap:       192772       8980     183792
```

Aktivno UNIX stablo

Značajni direktorijumi i datoteke u aktivnom UNIX stablu

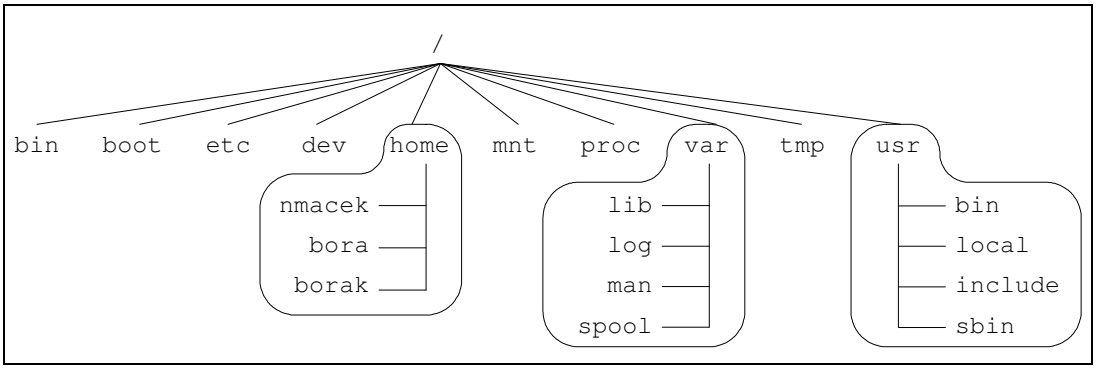
U ovom poglavlju opisani su značajni delovi aktivnog UNIX stabla definisani standardom Filesystem Hierarchy Standard v2.1. FHS standard definiše organizaciju aktivnog UNIX stabla i podelu stabla na nekoliko sistema datoteka specifične namene koje treba kreirati na odvojenim particijama ili diskovima. Datoteke su u aktivnom stablu grupisane prema nameni - sve komande se nalaze na jednom mestu, administrativni alati na drugom, dokumentacija na trećem, itd. Većina Linux distribucija sa manjim odstupanjem prati ovaj de facto standard, koji nije obavezan - stablo se može podeliti na veći broj sistema

datoteka zavisno od veličine diskova, a može biti i realizovano na samo jednom sistemu datoteka.

Po FHS standardu aktivno stablo čine sledeći sistemi datoteka:

- / (root), čiji se koreni direktorijum poklapa sa korenim direktorijumom aktivnog UNIX stabla. Root sistem datoteka sadrži osnovnu direktorijumsku strukturu stabla i sve datoteke neophodne za podizanje sistema i dovođenje sistema u stanje u kom ostali sistemi datoteka mogu biti aktivirani;
- /usr, u kome se nalazi većina korisničkih komandi, biblioteke, dokumentacija (man pages) i ostale relativno nepromenljive datoteke neophodne za normalno funkcionisanje sistema;
- /var, u kome se nalaze promenljive datoteke, poput spool direktorijuma (npr. red za štampu), log datoteka i nekih privremenih datoteka;
- /home, u kome se nalaze korisnički podaci odnosno lični direktorijumi svih korisnika sistema. Realizacijom ovog direktorijuma u vidu posebnog sistema datoteka olakšavaju se postupci arhiviranja podataka.

Osnovna struktura aktivnog UNIX stabla prema FHS standardu prikazana je na slici 3.8.



Slika 3.8 *Struktura aktivnog UNIX stabla prema FHS standardu*

root sistem datoteka

Root sistem datoteka sadrži osnovnu direktorijumsku strukturu stabla i sve datoteke neophodne za podizanje operativnog sistema. Sadržaj root sistema datoteka je najčešće dovoljan za jednokorisnički rad (singleuser mode) koji se koristi u svrhe administracije ili oporavka sistema, tako da se u okviru njega nalaze i osnovni administrativni alati. Najčešće se nalazi na lokalnom disku, a može biti smešten i na mreži, ili realizovan u formi ramdiska (sistem datoteka u operativnoj memoriji). Root direktorijum ne sadrži datoteke, osim simboličkog linka /vmlinuz na sliku jezgra (boot image).

U nastavku teksta dat je opis značajnih direktorijuma root sistema datoteka.

- /bin - najčešće korišćene komande koje mogu koristiti regularni korisnici.

- /sbin - komande namenjene superuseru, ali ih po potrebi mogu koristiti i obični korisnici ukoliko im se za to daju dozvole. /sbin se ne nalazi u putanji regularnih korisnika, ali se nalazi u putanji superusera.
- /etc - konfiguracione datoteke.
- /root - lični direktorijum korisnika root (superuser).
- /lib - deljene biblioteke neophodne za rad programa iz root sistema datoteka.
- /lib/modules - kernel moduli.
- /dev - specijalne datoteke koje predstavljaju uređaje (nodes), koje se kreiraju prilikom instalacije Linux sistema, a imena dobijaju na osnovu specijalnih konvencija. Dodatno se mogu kreirati pomoću /dev/MAKEDEV skript datoteke, što se radi isključivo ukoliko je direktorijum /dev oštećen.
- /tmp - privremene datoteke. Poželjno je da programi pokrenuti nakon podizanja sistema koriste /var/tmp za smeštanje privremenih datoteka, jer se na taj način root sistem datoteka održava urednim.
- /boot - datoteke koje koristi boot loader (npr. LILO), uključujući i slike kernela. Poželjno je ovaj direktorijum realizovati u formi odvojenog sistema datoteka koji će se nalaziti u okvirima prvih 1024 cilindara diska.
- /mnt - direktorijum u kome se nalaze mount-point direktorijumi (npr. /mnt/winc, /mnt/floppy, /mnt/cdrom). U nekim distribucijama Linux sistema ovaj direktorijum je preimenovan u /media.

/etc direktorijum

/etc direktorijum sadrži većinu konfiguracionih datoteka, uključujući i konfiguracione datoteke mrežnog okruženja. Značajne datoteke su opisane u daljem tekstu, a funkcija ostalih se može odrediti pomoću programa man koji prikazuje dokumentaciju o programu koji sadrži tu datoteku.

- /etc/fdprm - tabela parametara flopi diskova (floppy disk parameter table). Datoteka koju koristi program setfdprm.
- /etc/fstab - tabela auto-mount sistema datoteka (filesystem table).
- /etc/group - datoteka u kojoj su opisane sve korisničke grupe.
- /etc/init - konfiguraciona datoteka init procesa.
- /etc/issue - tekst koji proces getty prikazuje pre linije za unošenje korisničkog imena prilikom prijavljivanja na sistem (login prompt).
- /etc/magic - spisak magičnih brojeva. Ovu datoteku koristi program file prilikom određivanja tipa datoteke pomoću testa magičnih brojeva.
- /etc/motd - poruke od kojih se jedna prikazuje nakon uspešnog prijavljivanja na sistem (message of the day). Sadržaj ove datoteke određuje administrator sistema, koji u ovu datoteku može upisati poruku koju treba da vide svi korisnici sistema.

- /etc/mtab - tabela aktiviranih sistema datoteka (mount table). Koriste je razni programi kojima trebaju informacije o aktiviranim sistemima datoteka (npr. df)
- /etc/rc, /etc/rc.d ili /etc/rc?.d - skript datoteke koje se pokreću prilikom podizanja sistema ili promene nivoa izvršenja, ili direktorijumi u kojima se te skript datoteke nalaze.
- /etc/passwd - datoteka u kojoj su opisani svi korisnici sistema.
- /etc/printcap - baza podataka o mogućnostima štampača (printer capabilities).
- /etc/profile - datoteka koja se izvršava po pokretanju Bourne-Again shella. Ovo je globalna datoteka, tj. izvršavaće se bez obzira na to koji se korisnik prijavljuje na sistem.
- /etc/securetty - identifikacija "sigurnih" terminala, odnosno terminala s kojih root sme da se prijavi na sistem (secure getty). U smislu povećanja sigurnosti poželjno je zabraniti prijavljivanje superusera na sistem preko mreže. Tada se root može prijaviti sa mreže kao regularan neprivilegovani korisnik, a zatim pokrenuti program su (switch user) čime ostvaruje privilegije superusera.
- /etc/shadow - šifrovane lozinke korisnika sistema.
- /etc/shells - spisak komandnih interpretera kojima se veruje, tj. koje korisnici mogu navesti kao podrazumevane terminale komandom chsh.
- /etc/termcap - baza podataka o mogućnostima terminala (terminal capabilities).

/usr sistem datoteka

U /usr sistemu datoteka nalazi se većina korisničkih komandi, prateće biblioteke i dokumentacija (man pages). /usr sistem datoteka je relativno veliki, s obzirom da je većina programa koji pripadaju distribuciji Linux sistema tu instalirana. Lokalno instaliran softver smešta se u /usr/local, tako da se olakšava nadogradnja (upgrade) Linux distribucije novijom verzijom. Ovaj sistem datoteka može se nalaziti na mreži i računari ga mogu aktivirati u režimu čitanja kao mrežni sistem datoteka, čime se štedi na utrošenom prostoru na lokalnim diskovima i olakšava administracija. Tada je dovoljno izvršiti promene na jednom mestu i one će biti vidljive na svim računarima.

U nastavku teksta je dat opis značajnih direktorijuma /usr sistema datoteka.

- /usr/X11R6 - X Windows sistem.
- /usr/bin - većina korisničkih programa (user binaries).
- /usr/sbin - komande za administraciju raznih serverskih programa i ostale komande koje nisu neophodne za rad u jednokorisničkom režimu rada (superuser binaries).
- /usr/share/man, /usr/share/info, /usr/share/doc - on-line dokumentacija (man pages, GNU info, ...).
- /usr/include - zaglavlja za programe pisane u C programskom jeziku.

- /usr/lib - relativno nepromenljive prateće datoteke raznih programa, uključujući i neke konfiguracione datoteke.
- /usr/local - mesto za instalaciju softvera koji ne spada u Linux distribuciju.

/var sistem datoteka

U /var sistemu datoteka se nalaze datoteke koje se menjaju prilikom regularnog funkcionisanja sistema, poput spool direktorijuma i log datoteka. Ovaj direktorijum je specifičan za svaki sistem (radnu stanicu ili server) i mora se realizovati lokalno, a ne kao mrežni sistem datoteka.

- /var/cache/man - keš u kome se čuvaju formatirane man stranice.
- /var/lib - promenljive prateće datoteke raznih programa, uključujući i neke konfiguracione datoteke.
- /var/local - Promenljivi podaci koji pipadaju softveru instaliranom u /usr/local/bin direktorijumu (u ovakav softver spadaju npr. programi koje nije instalirao superuser). Lokalno instaliran softver koristi i ostale poddirektorijume /var sistema datoteka, npr. /var/lock.
- /var/lock - lock datoteke, odnosno indikatori korišćenja resursa, koje kreiraju razni programi koji koriste neke resurse sistema. Programi koji prate ovu konvenciju mogu pre korišćenja resursa da dobiju informaciju o njegovom zauzeću.
- /var/log - log datoteke koje kreiraju razni programi. Primeri ovih datoteka su /var/log/wtmp, u kojoj su zabeležena prijavljivanja (login) i odjavljivanja (logout) sa sistema i /var/log/messages, u koju syslog upisuje poruke kernela i sistemskih programa. Veličina nekih log datoteka brzo raste, tako da ove datoteke povremeno treba očistiti.
- /var/mail - datoteke koje predstavljaju poštanske sandučiće (mailbox). U zavisnosti od stepena odstupanja od FHS standarda, neke distribucije čuvaju ove datoteke u direktorijumu /var/spool/mail.
- /var/run - datoteke u kojima se čuvaju informacije o sistemu koje su validne do sledećeg podizanja sistema (reboot). Na primer, /var/run/utmp sadrži informacije o korisnicima koji su trenutno prijavljeni na sistem.
- /var/spool - spool direktorijumi za poštu (/var/spool/mail) i redovi za štampače (var/spool/lpd, /var/spool/cups).
- /var/tmp - privremene datoteke koje su prevelike da bi bile smeštene u /tmp ili treba da postoje na disku duže nego što bi to bilo moguće u direktorijumu /tmp.

/proc sistem datoteka

/proc sistem datoteka omogućava lak pristup strukturama podataka kernela, na osnovu čega se mogu dobiti informacije o sistemu (npr. o procesima, odakle potiče i naziv). proc

nije sistem datoteka u pravom smislu te reči, već simbolička predstava ovih struktura jednim kvazi-sistemom datoteka. Nijedna datoteka sa direktorijuma proc (uključujući veliku datoteku kcore) ne zauzima mesto na disku. Sve ove datoteke, koje se mogu videti pomoću standardnih alata za rad sa datotekama, kreira kernel u operativnoj memoriji računara. Većina datoteka proc sistema datoteka data je u formatu koji je prilagođen korisniku, odnosno u formatu koji je jednostavan za čitanje i razumevanje.

U nastavku teksta je dat opis značajnih datoteka i direktorijuma /proc sistema datoteka.

- /proc/1 - direktorijum u kome se nalaze datoteke sa informacijama o prvom procesu (init). Za svaki proces na /proc sistemu datoteka postoji direktorijum sa imenom rednog broja procesa u kom je taj proces opisan.
- /proc/cpuinfo - informacije o procesoru, kao što su proizvođač, model i performanse.
- /proc/devices - spisak blok i karakter uređaja koje podržava aktivni kernel.
- /proc/filesystems - sistemi datoteka za čije korišćenje je kernel konfigurisan.
- /proc/kcore - slika operativne memorije sistema. Može se iskopirati na drugo mesto u aktivnom stablu, čime se na realnom sistemu datoteka kreira slika operativne memorije (memory dump).
- /proc/kmsg - poruke koje kernel generiše a koje se dalje prosleđuju syslog procesu.
- /proc/meminfo - informacije o korišćenju operativne i swap memorije.
- /proc/modules - informacije o aktivnim modulima kernela.
- /proc/net - statusne informacije mrežnih protokola.
- /proc/uptime - vreme rada sistema (od poslednjeg podizanja operativnog sistema).
- /proc/version - verzija kernela.

3

KORISNICI I GRUPE

Svi višekorisnički operativni sistemi poseduju mehanizam autentifikacije korisnika pomoću korisničkih naloga. Korisnički nalog (user account) čini korisničko ime (username), jedinstveno za taj sistem, korisnički profil, mailbox, kao i sve datoteke i resursi koji tom korisniku pripadaju.

Korisnički nalozi

Kreiranje korisničkih naloga. Datoteka /etc/passwd. Globalni profil i inicijalno okruženje za nove korisnike Brisanje korisnika. Privremena zabrana rada korisnicima.

Svakom korisniku Linux sistema dodeljen je jedinstven celobrojni identifikator - UID (user ID) na osnovu kog kernel identifikuje korisnike. Ovaj metod predstavljanja kernelu karakterističan je za većinu operativnih sistema, uzevši u obzir da procesor brže radi sa brojnim vrednostima. Posebna baza podataka, koja radi u korisničkom režimu rada, dodeljuje tekstualna imena ovim numeričkim vrednostima, odnosno uparuje UID sa konkretnim korisničkim imenom. Dodatno, u bazi se nalaze i informacije o korisniku, kao što su opis, lokacija ličnog direktorijuma (home) i podrazumevani komandni interpreter (shell).

Na UNIX sistemima postoje dve vrste korisnika:

- sistemski korisnici, koji nastaju prilikom instalacije operativnog sistema i služe za specijalne namene, a ne za prijavljivanje na sistem. Jedini sistemski korisnik koji se može prijaviti na sistem je superuser root. Root ima sve privilegije i služi isključivo za administraciju sistema;
- regularni korisnici, koji služe za prijavljivanje na sistem. Regularne korisnike kreira superuser.

Kreiranje korisničkih naloga

Kreiranje regularnih korisničkih naloga je proces koji obuhvata: dodavanje informacija o korisniku u bazu, kreiranje ličnog direktorijuma, i po potrebi, podešavanje inicijalnog radnog okruženja (profila) korisnika. U većinu Linux distribucija uključeni su programi za kreiranje korisničkih naloga iz komandne linije (adduser i useradd) ili grafičkog okruženja (KUser).

Program adduser je interaktivni program koji nakon unošenja korisničkog imena kreira lični direktorijum i postavlja inicijalno okruženje, a zatim zahteva unošenje lozinke i osnovnih podataka o korisniku, koje unosi u bazu. Za razliku od njega, useradd zahteva da se podaci o korisniku navedu kao parametri u komandnoj liniji, da se unapred kreira lični direktorijum i u njega iskopiraju datoteke koje predstavljaju inicijalno okruženje. Prednost ovog načina kreiranja korisnika je mogućnost integracije u skript (npr. kreiranje korisnika na osnovu postojeće datoteke koju je šef poslao administratoru) a nedostatak zadavanje inicijalne lozinke koja je vidljiva na ekranu. Sledećim primerom prikazano je kreiranje korisnika jsmith ovim programima:

```
# adduser
Enter a username to add: jsmith
Adding user jsmith...
Adding new group jsmith (1051).
Adding new user jsmith (1051) with group jsmith.
Creating home directory /home/jsmith.
Copying files from /etc/skel
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for jsmith
Enter the new value, or press return for the default
        Full Name []: John Smith
        Room Number []: 409
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [y/n] y

# useradd -d /home/jsmith -u 1051 -p johnny jsmith
```

/etc/passwd

Baza podataka koja opisuje sve korisnike UNIX sistema realizovana je u obliku jednostavne tekstualne datoteke /etc/passwd (u literaturi se spominje i kao password file). Datoteka /etc/passwd ima strukturu tabele - jedan korisnik je opisan jednom linijom datoteke (ekvivalent zapisu baze), koju čini sedam polja razdvojenih dvotačkom:

- korisničko ime (username),
- šifrovana lozinka,
- UID,

- GID (numerički identifikator primarne grupe),
- opisni podaci (puno ime, brojevi telefona itd...),
- lični direktorijum (home),
- komandni interpreter (shell).

Korisnik kreiran komandom adduser u prethodnom primeru predstavljen je u datoteci /etc/passwd sledećom linijom:

```
jsmith:x:1051:1051:John Smith,409,,:/home/jsmith:/bin/bash
```

Svaki korisnik može videti sadržaj ove datoteke i na taj način se, na primer, može informisati o drugim korisnicima sistema. To znači da je i polje u kom se nalazi lozinka korisnika javno dostupno - zato se lozinke svih korisnika čuvaju u šifrovanom obliku, čime se postiže viši nivo sigurnosti. Ove metode zaštite nisu dovoljne jer je standardni algoritam crypt (modifikovani DES) koji se koristi za šifrovanje slab - kriptoanaliza se relativno lako može izvesti, naročito u slučajevima slabih lozinki, odnosno lozinki koje su kratke ili se mogu naći u standardnim rečnicima. Dodatnu sigurnost uvodi System V UNIX - lozinke se šifruju jačim algoritmom koji sistem podržava, a šifrat se skriva u datoteku /etc/shadow (shadow passwords).

/etc/shadow

U datoteci /etc/shadow, kojoj može da pristupi samo root, nalaze se šifrovane lozinke svih korisnika sistema i informacije o vremenskim ograničenjima. Datoteka /etc/shadow se može kreirati pomoću komande pwconv, koja na osnovu postojeće /etc/passwd i eventualno /etc/shadow kreira novu shadow datoteku. Obrnuti proces, odnosno uklanjanje shadow datoteke obavlja se komandom pwunconv.

Svaka linija u datoteci određuje parametre lozinke jednog korisnika i sastoji se od osam polja razdvojenih dvotačkom. Polja redom imaju sledeće značenje:

- ime korisnika, navedeno kao u /etc/passwd;
- šifrovana lozinka korisnika. Ako je ovo polje prazno lozinka nije potrebna za prijavljivanje na sistem;
- datum poslednje izmene lozinke (izražen u broju dana počev od 1. januara 1970 godine). Sva ostala polja u datoteci predstavljaju relativne veličine u odnosu na ovaj broj;
- najmanji broj dana važenja lozinke (ukoliko se stavi 0, znači da nema ograničenja);
- najveći broj dana važenja lozinke (prazno polje znači da ograničenje ne postoji);
- broj dana pre dana isteka lozinke u kojima se korisnik upozorava da treba da promeni lozinku;
- broj neaktivnih dana (broj uzastopnih dana u kojima se korisnik nije prijavljivao na sistem) nakon koga se nalog automatski zaključava;
- datum prestanka važenja lozinke.

Korisnik jsmith iz prethodnog primera biće u datoteci /etc/shadow opisan sledećom linijom:

```
jsmith:$1$mou5Sbvb$XyStBj/5og982wl6QuiIp.:12500:0:99999:7:::
```

Vremenska ograničenja lozinki se mogu pregledati i promeniti komandom chage (Change Age):

```
$ chage -m 10 -M 40 -W 20 -I 7 jsmith
$ chage -l jsmith
Minimum:        10
Maximum:        40
Warning:        20
Inactive:       7
Last Change:            jun 23, 2004
Password Expires:       avg 02, 2004
Password Inactive:      avg 09, 2004
Account Expires:        Never
```

Članstvo u grupama

Svaki korisnik UNIX sistema mora pripadati najmanje jednoj grupi, a dodatno može pripadati većem broju grupa i u svima je ravnopravan član. Primarna grupa korisnika je obavezan atribut svakog korisnika - to je grupa čiji je GID naveden u datoteci /etc/passwd u liniji kojom je korisnik opisan, odnosno grupa kojoj se dodeljuju objekti sistema datoteka koje korisnik kreira. Na UNIX sistemima postoje dve vrste grupa:

- sistemske, koje nastaju prilikom instalacije UNIX sistema i služe za specijalne namene. Grupa root spada u sistemsku grupu - korisnici koji su članovi ove grupe imaju relativno visoka ovlašćenja;
- regularne, koje kreira superuser, a koje se koriste u svrhe lakše administracije.

Regulisanjem članstva u grupama kontroliše se pristup resursima sistema (datotekama, štampačima itd...).

Sve grupe su opisane u datoteci /etc/group na sledeći način - jedna grupa je opisana jednom linijom datoteke, koju čine četiri polja razdvojenih dvotačkom:

- ime grupe,
- grupna lozinka,
- GID (group ID) - jedinstveni numerički identifikator grupe,
- spisak članova grupe u formatu: user1, user2, ... usern.

Dodela identifikatora korisnicima i grupama (UID i GID)

Na većini sistema koji nisu umreženi nije potrebno voditi računa o numeričkim vrednostima korisnika i grupa. Ove vrednosti su bitne u slučajevima kada je računar povezan na mrežu i kada je u upotrebi mrežni sistem datoteka (NFS), koji korisnike identifikuje na osnovu numeričkih vrednosti. U tom slučaju je potrebno eksplicitno dodeliti korisnicima iste vrednosti UID na svim sistemima, što se ne može ostvariti

automatskom dodelom UID pomoću programa za kreiranje korisnika. Uniformna dodela identifikatora korisnicima se može realizovati metodama centralizovane autentifikacije, kao što je NIS (Network Information Server). Dodatno je potrebno izbeći one UID koji su već korišćeni jer će korisnici koji ih dobiju moći da pristupe resursima starih korisnika (kao što su datoteke i mail).

Inicijalno okruženje - etc/skel direktorijum

Nakon kreiranja ličnog direktorijuma potrebno je kreirati inicijalno okruženje (profil) korisnika. Profil se inicijalizuje na osnovu datoteka koje se nalaze u direktorijumu /etc/skel. Administrator sistema u ovaj direktorijum smešta datoteke koje će činiti podrazumevani profil za sve novokreirane korisnike. Na primer, moguće je kreirati datoteku /etc/skel/.bash_profile koja će biti uključena u sve nove profile i koja će sadržati niz komandi koje je potrebno pokrenuti pri prvom prijavljivanju na sistem (podešavanje promenljivih, aliasa, putanje itd.) Korisnici dalje mogu profile prilagođavati svojim potrebama.

Osim ličnog profila postoje i globalne konfiguracione datoteke, kao što je /etc/profile u koju je potrebno upisati konfiguracione parametre zajedničke za sve korisnike.

Na ovaj način je omogućeno kreiranje relativno malog inicijalnog profila u /etc/skel direktorijumu. Dodatna prednost globalnih konfiguracionih datoteka je laka promena parametara okruženja koja će imati uticaj na sve korisnike sistema (parametri se menjaju u jednoj datoteci a ne u profilima svih korisnika).

Kreiranje korisničkih naloga bez upotrebe pomoćnih programa

Korisnički nalozi mogu se kreirati upotrebom programa specijalne namene, ili ručno, prateći sledeće korake:

- dodavanje informacija o novom korisniku u datoteku /etc/passwd. Ova datoteka se ne sme otvoriti standardnim editorom (kao što su vi, emacs ili joe), već isključivo pomoću vipw editora. vipw datoteku zatvara za upis, tako da druge komande ne mogu istovremeno da promene njen sadržaj. Komanda se navodi bez argumenata. Prijavljivanje na sistem ne treba dozvoliti pre kraja kompletne procedure kreiranja korisnika - u polje za lozinku treba uneti *, i na taj način privremeno zabraniti prijavljivanje korisnika na sistem;
- učlanjivanje korisnika u grupe pomoću vigr editora;
- kreiranje ličnog direktorijuma;
- kopiranje profila iz /etc/skel direktorijuma;
- dodela vlasništva nad ličnim direktorijumom i podešavanje pristupnih prava;
- dodela inicijalne lozinke komandom passwd, nakon čega se korisnik može prijaviti na sistem.

U primeru je dat jednostavan skup komandi kojima se može kreirati korisnički nalog jsmith:

```
# vipw
# vigr
# mkdir /home/jsmith
# cp -R /etc/skel/* /home/jsmith
# chown -R jsmith /home/jsmith
# chmod -R 700 /home/jsmith
# passwd jsmith
```

Promena parametara korisničkih naloga

Na većini UNIX sistema postoji nekoliko komandi kojima se mogu menjati razni parametri korisničkih naloga, odnosno adekvatna polja u datoteci /etc/passwd. Te komande su:

- chfn, kojom se menja puno ime korisnika;
- chsh, kojom se menja podrazumevani komandni interpreter;
- passwd, kojom se menja lozinka korisnika.

Superuser može korišćenjem ovih komandi promeniti osobine bilo kog naloga (kao što je prikazano u sledećem primeru). Regularni korisnici mogu promeniti isključivo osobine svog korisničkog naloga. Dodatna mera predostrožnosti koja se preporučuje je zabrana pristupa ovim komandama korisnicima koji nemaju iskustvo u radu sa UNIX-om.

```
# chfn jsmith
Changing the user information for jsmith
Enter the new value, or press return for the default
        Full Name [John Smith]: John Smith Jr.
        Room Number [409]: 425
        Work Phone []: 39xx450
        Home Phone []: 44xx012
        Other []:
# chsh jsmith
Changing the login shell for jsmith
Enter the new value, or press return for the default
        Login Shell [/bin/bash]: /bin/csh
# passwd jsmith
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Ostali administrativni zadaci vezani za korisničke naloge (promena korisničkog imena, lokacije ličnog direktorijuma i članstva u grupama) obavljaju se bez upotrebe specijalnih programa, odnosno promenom adekvatnih podataka u datotekama /etc/passwd i /etc/group pomoću vipw i vigr editora. Prilikom promene korisničkog imena moguć je otkaz autentifikacije kod servisa koji korisnike identifikuju na osnovu korisničkih imena, a ne na osnovu UID (npr. mail server), tako da je na tim serverima potrebno takođe načiniti izmene.

Privremena zabrana prijavljivanja na sistem

Ponekad je potrebno zabraniti prijavljivanje korisnika na sistem bez brisanja korisničkog naloga (npr. ukoliko korisnik ne plati za usluge ili ukoliko postoji osnovana sumnja da je njegova lozinka otkrivena). Najjednostavniji način je privremena promena lozinke korisnika - na taj način korisnik neće moći da se prijavi na sistem, ali o zabrani prijavljivanja neće dobiti nikakve informacije. Ovaj metod zabrane najbrže se realizuje, ali i donosi najviše problema administratoru (korisnik mora na neki drugi način da bude obavešten o zabrani - telefonom ili pismenim putem).

Bolji metod je zamena komandnog interpretera u datoteci /etc/passwd jednostavnim programom (shell skript ili preveden C program) koji će korisniku na ekranu ispisati poruku o privremenoj zabrani korišćenja sistema - npr. razlog i trajanje zabrane, kao što je prikazano u sledećem primeru:

```
# cat > /home/jsmith/warning
#!/usr/bin/tail +2
Prijavljivanje na sistem vam je privremeno zabranjeno.
Molimo vas da se javite administratoru sistema.
<CTRL-D>

# vipw
jsmith:x:1051:1051::/home/jsmith:/home/jsmith/warning
```

Ukoliko korisnik pokuša da se prijavi na sistem dobiće sledeće upozerenje:

```
login as: jsmith
jsmith@tulip's password:
Prijavljivanje na sistem vam je privremeno zabranjeno.
Molim vas da se javite administratoru sistema.
```

Uklanjanje korisnika

Uklanjanje korisnika, odnosno korisničkog naloga, sa sistema podrazumeva:

- brisanje svih datoteka, poštanskih sandučića, poslova za štampu i zakazanih poslova (cron jobs i at jobs). Prilikom brisanja potrebno je pronaći korisničke datoteke na celom aktivnom stablu, a ne samo na ličnom direktorijumu korisnika. Dodatno se preporučuje uvođenje privremene zabrane prijavljivanja na sistem (npr. jednostavnom zamenom lozinke) pre samog brisanja korisničkih datoteka;
- brisanje relevantnih linija iz datoteka /etc/passwd i /etc/group, pomoću vipw i vigr editora.

Korisnici se mogu obrisati i pomoću nekog administrativnog programa, poput deluser, pri čemu važe ista pravila: najpre je potrebno izvršiti zabranu prijavljivanja, zatim obrisati sve datoteke i na kraju pokrenuti program deluser koji će ukloniti korisnički nalog.

```
# passwd jsmith
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

```
# find / -user jsmith -exec rm {} \;
# deluser jsmith
Removing user jsmith...
done.
```

Identifikacija korisnika

Identifikacija korisnika i dobijanje informacija o korisničkom nalogu. Privremeno prijavljivanje na sistem pod drugim imenom. Stvarni i efektivni identifikatori korisnika (RUID i EUID).

Dve osnovne komande pomoću kojih se može odrediti ko je prijavljen na sistem su who i finger. Komanda who prikazuje korisničko ime, terminal (line), vreme prijavljivanja (login-time) i host računar (from) za sve korisnike koji su prijavljeni na sistem. Ukoliko se komanda zada sa parametrom -H rezultat će biti prikazan sa zaglavljem. Ukoliko se zada sa parametrom -q prikazuju se samo imena i ukupan broj korisnika prijavljenih na sistem.

```
# who -H
NAME     LINE     TIME         COMMENT
root     pts/0    Mar 24 18:50 (nicotine.internal.vets.edu.yu)
jsmith   pts/1    Mar 24 19:50 (lab409.internal.vets.edu.yu)
# who -q
root jsmith
# users=2
```

Komanda finger daje sličan rezultat - prikazuje korisnike prijavljene na sistem, a pomoću nje se mogu dobiti i detaljne informacije o korisnicima iz /etc/passwd datoteke, bez obzira da li su oni trenutno prijavljeni na sistem ili ne. Dodatno se mogu dobiti i informacije o korisnicima udaljenih sistema (npr: finger coyote@acme.com), ali se takvi pokušaji najčešće završe porukom "connection refused".

```
# finger
Login    Name          Tty      Idle   Login Time
jsmith   John Smith    pts/1       1   Mar 25 15:48  (nicotine)
root     root         *pts/0           Mar 25 15:47  (nicotine)

# finger jsmith
Login: jsmith                           Name: John Smith Jr.
Directory: /home/jsmith                 Shell: /bin/bash
Office: 425, 39xx450                    Home Phone: 44xx012
Last login Wed Mar 24 17:28 (CET) on pts/1 from nicotine
No mail.
No Plan.
```

Privremeno prijavljivanje na sistem pod drugim imenom

Korisnik se privremeno može prijaviti na sistem pod drugim imenom pomoću komande su (switch user) i na taj način pristupiti resursima koji pripadaju drugom korisniku. Najčešće

se koristi ukoliko je potrebno izvršiti promenu datoteka koje pripadaju drugom korisniku, promenu pristupnih prava ili pokretanje nekog programa. Administratori koriste ovu komandu da bi razrešili neki problem sa korisničkim nalogom ili u svrhe testiranja autorizacije i ponašanja naloga sa izvesnim aplikacijama.

Sintaksa komande su je sledeća:

```
# su [-] [username]
```

Od korisnika koji pokreće komandu su (ukoliko to nije root) zahteva se da unese i lozinku za korisnički nalog koji želi privremeno da koristi. Nakon unošenja lozinke korisnik ima sve privilegije tog naloga. Povratak na originalni korisnički nalog vrši se komandom exit.

Sledeći primer ilustruje korišćenje komande su:

```
$ whoami
jsmith
$ pwd
/home/jsmith
$ su nmacek
Password:
$ whoami
nmacek
$ pwd
/home/jsmith
$ exit
exit
```

Ako se komanda su zada bez argumenata, promenljive koje čine radno okruženje korisnika i tekući direktorijum se ne menjaju. Ukoliko je potrebno da se prilikom privremenog prijavljivanja na sistem pročitaju nove inicijalizacione datoteke specifičnog korisnika, potrebno je zadati komandu sa parametrom "-" pre korisničkog imena (npr. su - nmacek). Na taj način će se izvršiti postavljanje promenljivih i prelazak na home direktorijum tog korisnika.

Ukoliko se prilikom zadavanja komande su ne navede korisničko ime, korisnik će biti privremeno prijavljen kao superuser root (pod uslovom da zna lozinku). Dodatno, ukoliko treba postaviti okruženje superusera, odnosno ukoliko postoji potreba za pokretanjem programa koji su dostupni isključivo superuseru i nalaze se u njegovoj putanji (PATH), komanda se zadaje sa parametrom "-". Administratori UNIX/Linux sistema najčešće koriste root nalog na ovaj način - root nalog ne treba koristiti za obavljanje poslova koji nisu administrativne prirode i koji se mogu obaviti sa nalogom regularnog korisnika.

```
$ whoami
jsmith
$ pwd
/home/jsmith
$ su -
Password:
# whoami
root
# pwd
/root
```

Stvarni i efektivni identifikatori korisnika (RUID i EUID)

ID korisnika koji je inicijalno prijavljen na UNIX sistem (pomoću procesa login) je stvarni identifikator korisnika (RUID - Real User ID). Ukoliko se korisnik privremeno prijavi na sistem pod drugim imenom pomoću komande su, njegov ID se privremeno menja. U cilju diferenciranja inicijalno i privremeno prijavljenih korisnika uvodi se efektivni identifikator korisnika (EUID - Effective User ID). Za RUID i EUID važi sledeće:

- RUID je ID korisnika koji je inicijalno prijavljen na sistem i ne menja se tokom rada, bez obzira da li je korisnik pokrenuo komandu su i pomoću nje se prijavio pod drugim imenom,
- EUID je jednak RUID ukoliko korisnik nije pokrenuo komandu su, odnosno UID korisnika pod čijim imenom je privremeno prijavljen, ukoliko je izvršena zamena identiteta komandom su.

Sledeći primer ilustruje komande who am i i id pomoću kojih se može odrediti identitet korisnika koji je privremeno prijavljen pod drugim nalogom. Komande who am i i id prikazuju RUID i EUID korisnika respektivno.

```
$ id
uid=1051(jsmith) gid=1051(jsmith) groups=1051(jsmith)
$ who am i
jsmith   pts/1        Mar 25 16:09 (nicotine.internal.vets.edu.yu)
$ su -
Password:
# id
uid=0(root) gid=0(root) groups=0(root)
# who am i
jsmith   pts/1        Mar 25 16:09 (nicotine.internal.vets.edu.yu)
```

4

KONTROLA PRISTUPA NA NIVOU SISTEMA DATOTEKA

Pristup resursima pod mrežnim operativnim sistemima je strogo kontrolisan. Sistem datoteka je fundamentalni resurs svake radne stanice ili servera, a kontrola pristupa datotekama i direktorijumima (dodela ovlašćenja za pristup i zaštita od neovlašćenog pristupa) ključna komponenta ozbiljnih zaštitnih polisa.

Vlasnički odnosi i prava pristupa

Prava pristupa za datoteke i direktorijume. Kategorije pristupnih prava. Vlasnički odnosi.

Jedna od najznačajnijih komponenti svake ozbiljne zaštitne polise je kontrola pristupa na nivou sistema datoteka. Kontrolom pristupa na nivou sistema datoteka određuju se:

- skup korisnika koji mogu pristupiti objektima, odnosno datotekama i direktorijumima,
- nivo pristupa, odnosno skup akcija koje autorizovani korisnici mogu izvršiti nad tim objektima.

Kontrola pristupa na nivou sistema datoteka zasniva se na vlasničkim odnosima, odnosno vlasništvu nad objektima (pripadnost objekta korisnicima i grupama) i pravima pristupa. Prava pristupa se dodeljuju svakoj datoteci i direktorijumu i na osnovu tih prava se određuje nivo pristupa tim objektima.

Prava pristupa u sistemu datoteka

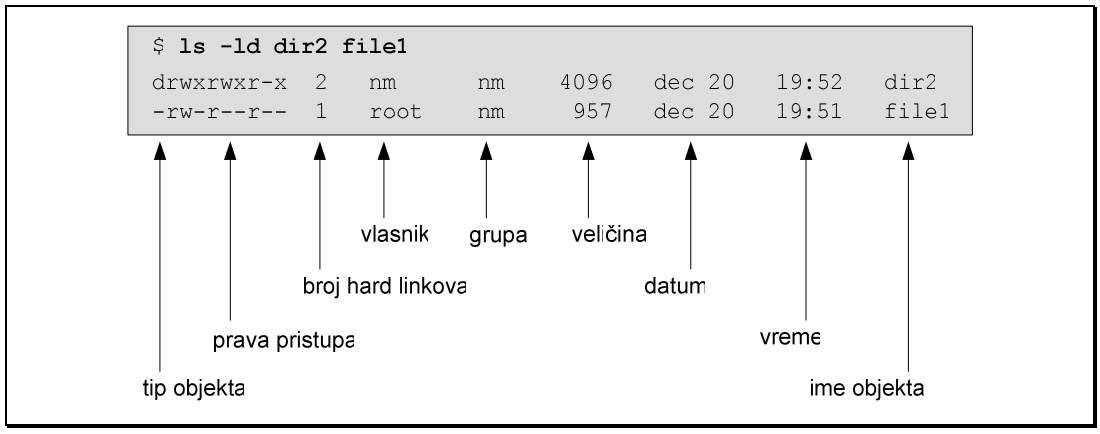
Pravima pristupa određuje se skup korisnika koji mogu pristupati datotekama i direktorijumima i skup akcija koje ti korisnici mogu da izvrše nad njima. Prava pristupa za

datoteke i direktorijume najlakše se mogu odrediti pomoću komande ls (list) sa parametrom -l (long), kao što je prikazano u sledećem primeru.

```
$ ls -ld
total 5
drwxrwxr-x   2   jsmith   nm        4096   dec 20   19:52   dir1
drwxrwxr-x   2   nm       nm        4096   dec 20   19:52   dir2
-rw-r--r--   1   root     nm         957   dec 20   19:51   file1
-rw-r--r--   1   root     root     13597   dec 20   19:52   file2
-rw-r--r--   1   nm       nm        1326   dec 20   19:52   file3
```

Komanda ls –l prikazuje listing sadržaja direktorijuma u dugom (long) formatu. Za svaki objekat prikazuje se potpuni UNIX kontekst, odnosno sve informacije o objektu. Ako se u argumente komande ls uključi i -a (all) opcija, prikazuju se sve datoteke, uljučujući i skrivene (datoteke koje počinju tačkom). Dugački listing komande ls –l obuhvata sledeće informacije za svaku datoteku ili direktorijum:

- tip objekta (d za direktorijum, - za regularnu datoteku, l za simbolički link, p za pipe, c za karakter uređaj, b za blok uređaj),
- prava pristupa za tri vlasničke kategorije,
- broj hard linkova na datoteku,
- korisničko ime vlasnika objekta,
- ime grupe kojoj objekat pripada,
- veličina objekta,
- datum,
- vreme,
- ime objekta .



Slika 4.1 *Dugačak listing (komanda ls -l)*

Kategorije pristupnih prava

Prvi karakter u polju od 10 karaktera u dugačkom listingu nije deo prava pristupa, već ukazuje na tip datoteke. Ukoliko je prvi karakter (–) (dash), to znači da je reč o običnoj, regularnoj datoteci. Ukoliko je prvi karakter d, to znači da je reč o direktorijumu. Postoje i drugi tipovi datoteka i za svaki od njih postoji poseban znak na prvoj poziciji, kao što su b (blok uređaj - block special file), c (karakter uređaj - character special file), l (simbolički link), p (imenovani pipe).

Sledećih devet znakova predstavljaju prava pristupa objektu za tri vlasničke kategorije, a to su vlasnik, grupa i ostali. Prva tri karaktera definišu prava pristupa vlasnika, druga tri prava pristupa grupe kojoj datoteka pripada i poslednja tri karaktera prava pristupa za ostatak sveta.

U nastavku teksta dato je kratko objašnjenje vlasničkih kategorija.

- Vlasnik (owner) najčešće je korisnik koji je kreirao objekat, osim ukoliko superuser ne promeni vlasništvo. U tom slučaju, vlasnik je korisnik kome je vlasništvo dodeljeno. Vlasnik objekta može biti bilo koji korisnik sistema, regularan ili sistemski.
- Grupa (group) je korisnička grupa kojoj je datoteka formalno priključena. Za razliku od korisnika koji mogu pripadati većem broju grupa, objekti sistema datoteka mogu pripadati samo jednoj grupi, koja može biti regularna ili sistemska. Najčešće je to primarna grupa korisnika koji je objekat kreirao. Superuser naknadno može promeniti pripadnost objekta grupi. Datoteke i direktorijumi ne priključuju se grupama u datoteci /etc/group (što bi bilo nepraktično zbog velikog broja objekata koji su priključeni grupama), već se grupa kojoj objekat pripada upisuje u njegov i-node.
- Ostali (others, public) su svi korisnici koji nisu ni vlasnik objekta, niti pripadaju grupi kojoj objekat pripada.

Prava pristupa za svaku vlasničku kategoriju eksplicitno se dodeljuju svakom objektu prilikom kreiranja, a kasnije se mogu promeniti. Potpuni skup prava za svaku vlasničku kategoriju čine:

- pravo čitanja (r - read),
- pravo upisa (w - write),
- pravo izvršavanja (x - execute).

Ova prava imaju strogu poziciju u svim vlasničkim kategorijama: uvek je prva pozicija pravo čitanja, druga pozicija pravo upisa i treća pravo izvršavanja. Ukoliko se na odgovarajućoj poziciji umesto simbola kojim je pravo predstavljeno (r,w ili x) nalazi crtica, pravo je ukinuto. Značenje prava za datoteke i direktorijume bitno se razlikuje.

U nastavku teksta data je interpretacija pristupnih prava skrivene datoteke .bash_profile (inače, datoteka .bash_profile se izvršava nakon prijavljivanja korisnika na sistem i formira okruženje za korisnika)

```
$ ls -la ~/.bash_profile
```

```
-rw-r--r--   1   nm   nm   509   Mar 10   17:21   .bash_profile
```

Prvih deset karaktera predstavljaju tip datoteke i prava pristupa. Kao što je već pomenuto, prvi karakter ne spada u prava pristupa već određuje tip datoteke. U ovom slučaju to je (-), što znači da je reč o običnoj, regularnoj datoteci. Znak (-) na bilo kojoj drugoj poziciji, ukazuje na suspenziju prava

Sledeća tri karaktera određuju prava pristupa za vlasnika datoteke (vlasničko pravo), odnosno prava koja vlasnik ima nad datotekom. U slučaju datoteke .bash_profile vlasnik datoteke je korisnik nm, što se vidi iz trećeg polja. Prava vlasnika nad datotekom .bash_profile su rw-, što znači da je može čitati (pravo r) i modifikovati (pravo w), ali je ne može izvršavati (znak - na trećoj poziciji). Vlasnici najčešće nemaju pravo izvršavanja nad datotekama koje se ne mogu izvršavati. U ovom slučaju, datoteka ima formu shell programa ali se izvršava isključivo prilikom prijavljivanja na sistem, tako da korisniku nm nije dodeljeno pravo izvršavanja.

Sledeći skup od tri karaktera definiše prava pristupa grupe (grupno pravo). Sistem administratori kreiraju grupe i definišu pripadnost korisnika grupama. Grupno pravo se primenjuje na sve korisnike koji pripadaju istoj grupi kojoj pripada i datoteka. U slučaju datoteke .bash_profile, koja pripada grupi staff, grupno pravo je r--, što znači da svi korisnici koji pripadaju grupi staff mogu da pročitaju datoteku (pravo r), ali je ne mogu modifikovati ni izvršavati.

Poslednji skup od tri karaktera predstavlja prava pristupa za sve korisnike koji nisu ni vlasnik niti pripadaju istoj grupi kojoj datoteka pripada (javno pravo pristupa). U ovom slučaju, grupno pravo je r--, što znači da svaki korisnik koji pripada ovoj vlasničkoj kategoriji može čitati datoteku (pravo r), ali je ne može modifikovati niti izvršavati.

Sledeći primer ilustruje prava pristupa sistemskim direktorijumima /bin i /root.

```
$ ls -ld /bin /root
drwxr-xr-x   2   root   root   2048   Apr  1    20:16   /bin
drwxr-x---   7   root   root   1024   Apr 20    15:43   /root
```

Sistemski direktorijum bin sadrži najčešće korišćene UNIX komande. Svim korisnicima sistema dato je pravo korišćenja direktorijuma bin. Svi korisnici sistema mogu da se pozicioniraju na direktorijum, mogu da pročitaju sadržaj i pokrenu komande koje se u njemu nalaze. Pravo upisa dato je jedino superuseru.

Sistemski direktorijum root je home direktorijum superusera, koji nad njim ima sva prava. Članovi grupe root mogu da pročitaju sadržaj direktorijuma i da se pozicioniraju na njega, dok je ostalim korisnicima pristup direktorijumu zabranjen.

Problem unije vlasničkih kategorija

Prava pristupa određuju se prema vlasničkoj kategoriji kojoj korisnik pripada. U slučajevima unije vlasničkih kategorija (vlasnik datoteke pripada grupi kojoj pripada datoteka) vlasništvo se određuje po pravilima konkretnog UNIX sistema.

- Kod starijih UNIX sistema, pristupna prava vlasnika koji pripada grupi kojoj pripada objekat određena su unijom vlasničkog i grupnog prava.

- Kod UNIX sistema koji prate POSIX standard (POSIX compliant), kao što je Linux, pristupna prava vlasnika koji pripada grupi kojoj pripada i datoteka određena su isključivo vlasničkim pravom.

U svakom slučaju, korisnik se u odnosu na objekat u sistemu datoteka može naći u 3+1 vlasničkoj poziciji: vlasnik (owner), grupa (group), ostatak sveta (others), i vlasnik i grupa istovremeno.

Značenje prava pristupa za datoteke i direktorijume

Značenje pristupnih prava različito je za datoteke i direktorijume. U odnosu na datoteku, pristupna prava imaju sledeće značenje:

read (r) korisnik može pročitati sadržaj datoteke, odnosno može prikazivati datoteku na ekranu, štampati je ili kopirati;

write (w) korisnik može modifikovati sadržaj datoteke. Napomena: ovo pravo ne znači da korisnik može da obriše datoteku! Korisnik može obrisati datoteku samo ako mu je dato pravo upisa nad roditeljskim direktorijumom;

execute (x) korisnik može izvršavati datoteku, pod uslovom da se radi o shell programu ili o datoteci u binarnom izvršnom formatu;

Pristupna prava u odnosu na direktorijum su malo složenija i imaju sledeće značenje:

read (r) korisnik može pročitati sadržaj direktorijuma, što znači da korisnik može da izvrši komandu ls. Napomena: pravo r nad direktorijumom nije dovoljno da korisnik prikaže dugački listing direktorijuma (ls -l). Za to je neophodno i x pravo nad direktorijumom;

write (w) korisnik može modifikovati sadržaj direktorijuma, odnosno dodavati nove datoteke i brisati postojeće, kreirati i brisati poddirektorijume. Napomena: ovo pravo ne znači da korisnik može da obriše direktorijum! Korisnik može obrisati direktorijum samo ako mu je dato pravo upisa nad roditeljskim direktorijumom;

execute (x) korisnik se može pozicionirati na direktorijum (komandom cd), može prikazivati dugački listing (ls -l) sadržaja i pretraživati direktorijum (find).

Napomena: da bi se direktorijum mogao praktično koristititi autorizovanim korisnicima treba dati prava r i x. Ovo važi isključivo za Linux.

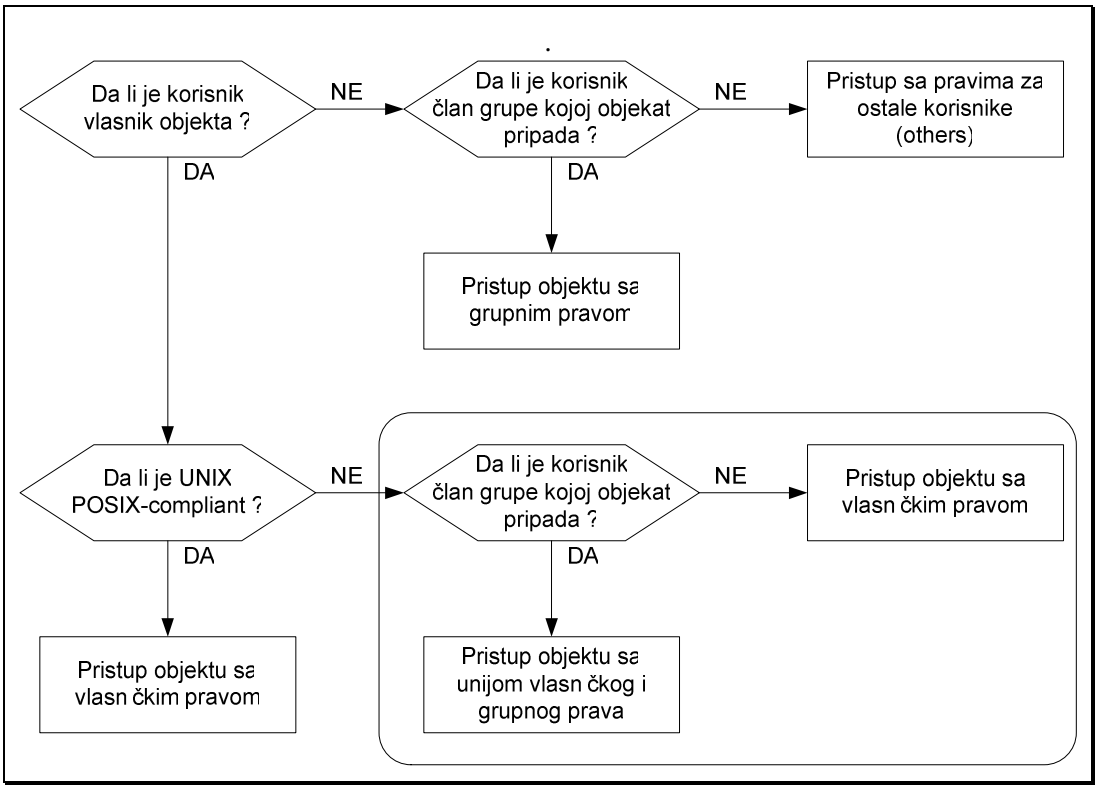
Određivanje pristupa za datoteke i direktorijume

Svim datotekama i direktorijumima dodeljen je korisnički identifikator (UID) i grupni identifikator (GID). Kernel razrešava vlasničke odnose na osnovu ovih identifikatora, a ne na osnovu imena korisnika i grupa, čime se troši manja količina memorije, a i sam postupak kraće traje. Komanda ls -n će prikazati numeričke UID i GID, kao što je prikazano u sledećem primeru. Primeri takođe ilustruju funkcionisanje komandi id i groups, koje su pogodne za analizu vlasničkih odnosa. Komanda id prikazuje UID i ime

tekućeg korisnika, a komanda groups GID primarne grupe i imena svih grupa kojima tekući korisnik pripada.

```
$ ls -ln
-rw-rw-r--   1   859   861    20   dec 23   14:04   kyuss
-rw-rw-r--   1   859   861    20   dec 23   15:20   stoner
$ id
uid=859(nm) gid=861(nm) groups=861(nm),0(root)
$ groups
nm root
```

Sve sistemske procese kreira operativni sistem na zahtev ili samog sistema ili nekog korisnika. Svaki proces ima svoj UID i GID, koji se poklapa sa UID i GID korisnika koji je proces inicirao. Kada korisnik putem svog procesa pokuša da čita, modifikuje ili izvršava datoteku, upoređuju se UID i GID procesa i datoteke (najpre UID, a potom GID). Na osnovu ovog poređenja korisnik se postavlja u odgovarajuću vlasničku kategoriju, a zatim se na osnovu pristupnih prava te vlasničke kategorije određuje da li korisnik ima ili nema prava da obavi željenu akciju. Sledeći dijagram ilustruje logiku određivanja da li korisnički proces ima pravo da pristupi datoteci na željeni način:



Slika 4.2 *Logika određivanja pristupnih prava*

Podrazumevana prava pristupa (umask)

Nakon kreiranja novog objekta UNIX postavlja takozvana podrazumevana (inicijalna) prava pristupa. U nastavku teksta opisan je postupak formiranja podrazumevanih pristupnih prava na primeru nove datoteke i novog direktorijuma, koji su kreirani komandama touch i mkdir.

```
$ touch myfile
$ mkdir mydir
$ ls -l
drwxrwxr-x   2   nm   nm   4096   dec 23   14:33   mydir
-rw-rw-r--   1   nm   nm      0   dec 23   14:33   myfile
```

Komande takođe ilustruju postupak dodele inicijalnog vlasništva: korisnik koji kreira objekat postaje njegov vlasnik, nakon čega se objekat formalno dodeljuje primarnoj grupi vlasnika. Inicijalna prava pristupa dodeljuju se na osnovu vrednosti promenljive umask, koja je specifična za svakog korisnika. Vrednost promenljive umask postavlja se prilikom prijavljivanja na sistem, a naknadno se može promeniti. Trenutnu vrednost promenljive umask svaki korisnik može videti i promeniti pomoću istoimene komande. Prilikom promene vrednosti potrebno je zadati komandu umask praćenu sa tri spojene oktalne cifre (svaka cifra u opsegu od 0 do 7), čime se definiše nova vrednost promenljive.

```
$ umask
0002
$ umask 022
$ umask
0022
```

Inicijalna prava datoteka i direktorijuma određuju se različitim formulama. Inicijalna prava direktorijuma predstavlja vrednost 777 od koje je oduzeta vrednost promenljive umask: perm(dir)=777-umask. To znači da su za vrednost promenljive umask=027 inicijalna prava za direktorijum 750.

Datotekama se inicijalno ne dodeljuju prava izvršavanja ni u jednoj vlasničkoj kategoriji, tako da se podrazumevana prava računaju po nešto složenijoj formuli. Vrednost 666 se logički množi sa komplementom promenljive umask: perm(file)= 666&(^umask), čime se formiraju prava za datoteku. To znači da su za vrednost promenljive umask=027 inicijalna prava za datoteku 640.

Tipični primeri za umask vrednosti, kao i podrazumevana prava za datoteke i direktorijume su:

umask vrednost	podrazumevana prava za datoteku	podrazumevana prava za direktorijum
077	600 (`rw-------`)	700 (`rwx------`)
027	640 (`rw-r-----`)	750 (`rwxr-x---`)
022	644 (`rw-r--r--`)	755 (`rwxr-xr-x`)
007	660 (`rw-rw----`)	770 (`rwxrwx---`)
002	664 (`rw-rw-r--`)	775 (`rwxrwxr-x`)
000	666 (`rw-rw-rw-`)	777 (`rwxrwxrwx`)

Tabela 4.1 *Vrednosti umask promenljive i podrazumevana prava pristupa*

Eksperimentisanje sa komandama touch, mkdir, ls -l i umask je najbolji način da korisnik shvati kako se umask vrednosti upotrebljavaju za kontrolu pristupa.

```
$ umask 066
$ touch myfile
$ mkdir mydir
$ ls -l
drwx--x--x   2   nm   nm   4096   dec 23   14:41   mydir
-rw-------   1   nm   nm      0   dec 23   14:41   myfile
$ rm -r myfile mydir
$ umask 027
$ touch myfile
$ mkdir mydir
$ ls -l
drwxr-x---   2   nm   nm   4096   dec 23   14:42   mydir
-rw-r-----   1   nm   nm      0   dec 23   14:42   myfile
```

Komanda umask na prvi pogled izgleda konfuzno, ali je vrlo značajna. Korisnik koji ne izabere adekvatnu vrednost umask promenljive mora često da koristi komande za promenu pristupnih prava, ukoliko želi da zaštiti svoje datoteke i direktorijume. Ukoliko je značaj podataka veliki,potrebno je koristiti dodatne mehanizme zaštite, poput šifrovanja pojedinačnih datoteka ili skladištenja značajnih datoteka na kripto sistemima datoteka.

Promena vlasništva i pristupnih prava

Promena pristupnih prava komandom chmod. Promena vlasničkih odnosa komandama chown i chgrp. Postavljanje specijalnih atributa: setuid, setgid i sticky bit.

Inicijalna prava pristupa dodeljuju se svakom novokreiranom objektu u sistemu datoteka. Iako su inicijalna prava pristupa uglavnom dovoljno restriktivna, korisnici mogu imati potrebu da pojedinačnim datotekama ili direktorijumima promene prava pristupa. Dodatno, korisnici mogu "pokloniti" datoteku ili direktorijum drugom korisniku, ukoliko su nezadovoljni inicijalnim vlasničkim odnosima i ukoliko konkretan sistem to dozvoljava.

Promena pristupnih prava

Prava pristupa objektima mogu se promeniti komandom chmod (change mode). Inicijalna prava datoteke najčešće su takva da sve vlasničke kategorije imaju pravo čitanja, što znači da mogu da pročitaju, kopiraju i odštampaju datoteku. Međutim korisnici često ukidaju pravo čitanja vlasničkim kategorijama grupe i ostatka sveta ukoliko su datoteke poverljive, odnosno dodeljuju pravo upisa, ukoliko su datoteke javno dostupne većem broju korisnika. Shell programi (shell script) su sledeći primer datoteka kojima se menjaju inicijalna prava pristupa. Kada korisnik kreira shell program, inicijalna prava pristupa ne uključuju pravo izvršavanja ni za jednu vlasničku kategoriju. Da bi mogao direktno da izvršava shell script datoteku korisnik mora da joj dodeli x pravo, kao što je učinjeno u sledećem primeru:

```
$ touch myscript
$ ls -l myscript
-rw-r--r--   1   nm   nm   0   dec 23   14:49   myscript
$ chmod u=rwx myscript
$ ls -l myscript
-rwxr--r--   1   nm   nm   0   dec 23   14:49   myscript
```

Prava pristupa mogu promeniti isključivo vlasnici datoteka i direktorijuma, dok root kao superuser može da promeni pristupna prava svakom objektu. Komanda chmod može se pokrenuti u simboličkom (relative) ili oktalnom (absolute) režimu.

U simboličkom režimu dodaju se ili oduzimaju prava izabranim vlasničkim kategorijama. Ovaj režim je pogodan za pojedinačno postavljanje ili ukidanje prava većem broju datoteka. U oktalnom režimu se korišćenjem oktalnih brojeva (0-7) postavljaju prava pristupa za sve vlasničke kategorije.

Simbolički režim

Simbolički režim se koristi za postavljanje jednog ili više pristupnih prava u jednoj ili više vlasničkih kategorija. Ovaj režim se naziva relativnim zbog toga što korisnik dodeljuje ili oduzima prava u odnosu na postojeća, dok se postojeća prava koja nisu specificirana argumentom komande ne menjaju. Korisnik može takođe specificirati sva prava pristupa u svim vlasničkim kategorijama. Format komande u simboličkom modu je:

```
$ chmod [-R] symbolic_mode objectname
```

odnosno:

```
$ chmod [-R] category op permissions[,...] objectname
```

Argument symbolic_mode sastoji se od tri komponente:

- komponenta category opisuje vlasničku kategoriju na koju se komanda odnosi: vlasnik (u), grupa (g), others (o), sve kategorije (a);
- komponenta op je operator koji dejstvuje na prava: dodela prava (+), ukidanje prava (-), dodela tačno određenih prava (=);

- komponenta permissions su prava pristupa koja se dodeljuju ili oduzimaju: čitanje (r), modifikacije (w) i izvršavanje (x).

Dobra osobina simboličkog režima je što korisnik može dodeliti ili ukinuti prava koja želi, bez poznavanja trenutnih prava. Komanda je jako korisna za dodelu, odnosno oduzimanje prava većem broju datoteka sa različitim trenutnim skupom prava.

Sledeći primer ilustruje korišćenje simboličkog režima: najpre je komandom touch kreirana datoteka myfile, sa inicijalnim pravima pristupa 644, a zatim su komandama chmod izvršene sledeće aktivnosti:

- dodata prava upisa kategorijama group i others,
- oduzeto pravo upisa vlasniku,
- dodeljen egzaktan skup prava (r i w) vlasniku i ukinuto pravo upisa kategorijama group i others,
- svima ukinuta sva prava.

```
$ touch myfile
$ ls -l myfile
-rw-r--r--   1   nm   nm   0   dec 23   15:25   myfile
$ chmod go+w myfile
$ ls -l myfile
-rw-rw-rw-   1   nm   nm   0   dec 23   15:25   myfile
$ chmod u-w myfile
$ ls -l myfile
-r--rw-rw-   1   nm   nm   0   dec 23   15:25   myfile
$ chmod u=rw,go-w myfile
$ ls -l myfile
-rw-r--r--   1   nm   nm   0   dec 23   15:25   myfile
$ chmod a= myfile
$ ls -l myfile
----------   1   nm   nm   0   dec 23   15:25   myfile
```

Parametar -R se koristi za rekurzivnu promenu pristupnih prava direktorijuma i svih objekata (poddirektorijuma i datoteka) koji se u njemu nalaze. Ukoliko se navede parametar -R, argument objectname mora biti direktorijum. Sledeći primer ilustruje rekurzivnu promenu pristupnih prava.

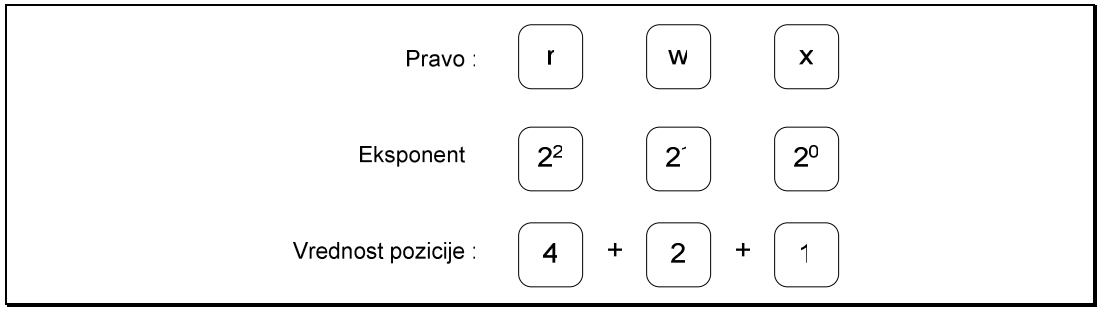
```
$ ls -ld parent_dir
drwxr-xr-x   2   nm   nm   4096    Apr 28   09:10   parent_dir
$ ls -l parent_dir
parent_dir:
total 0
-rw-r--r--   1   nm   nm      0   Apr 28   09:09   dir1
-rw-r--r--   1   nm   nm      0   Apr 28   09:10   dir2
-rw-r--r--   1   nm   nm      0   Apr 28   09:09   file1
-rw-r--r--   1   nm   nm      0   Apr 28   09:09   file2
$ chmod -R o-rx parent_dir
$ ls -ld parent_dir
drwxr-xr-x   2   nm   nm   4096    Apr 28   09:10   parent_dir
$ ls -l parent_dir
parent_dir:
total 0
```

```
-rw-r--r--   1   nm   nm       0   Apr 28   09:09   dir1
-rw-r--r--   1   nm   nm       0   Apr 28   09:10   dir2
-rw-r--r--   1   nm   nm       0   Apr 28   09:09   file1
-rw-r--r--   1   nm   nm       0   Apr 28   09:09   file2
```

Oktalni režim

Komandom chmod u oktalnom režimu dodeljuju se prava pristupa svim vlasničkim kategorijama istovremeno. Prava koja korisnik navede kao argument komande eksplicitno zamenjuju postojeća prava (prethodna prava se ne prolongiraju), tako da se ovaj režim naziva apsolutnim. Komanda zatheva da se u ovom režimu kao argument navedu tri oktalne cifre od kojih svaka predstavlja prava pristupa za jednu vlasničku kategoriju.

Suština oktalnog režima je u binarno-eksponencijalnoj reprezentaciji prava. Prava se predstavljaju binarno, pri čemu dodeljena prava imaju vrednost 2, a ukinuta vrednost 0. Svako pravo se zatim stepenuje, pri čemu se eksponent određuje na osnovu pozicije prava u vlasničkoj kategoriji (eksponent za pravo čitanja je 2, za pravo upisa 1, za pravo izvršavanja 0). Na kraju se vrednosti saberu i time se dobija oktalni broj kojim su određena prava pristupa za jednu vlasničku kategoriju. Ovaj postupak je ilustrovan slikom 4.3.



Slika 4.3 *Predstavljanje prava oktalnim brojevima*

Moguće oktalne vrednosti sa odgovarajućim pravima opisane su sledećom tabelom:

Oktalna vrednost	Suma prava po binarnoj vrednosti	Odgovarajuća prava	Definicija
7	4 + 2 + 1	`r w x`	čitanje, izmena i izvršavanje
6	4 + 2 + 0	`r w -`	čitanje i izmena
5	4 + 0 + 2	`r - x`	čitanje i izvršavanje
4	4 + 0 + 0	`r - -`	samo čitanje
3	0 + 2 + 1	`- w x`	izmena i izvršavanje
2	0 + 2 + 0	`- w -`	samo izmena
1	0 + 0 + 2	`- - x`	samo izvršavanje
0	0 + 0 + 0	`- - -`	bez prava pristupa

Sintaksa komande chmod u oktalnom režimu je slična sintaksi komande u simboličkom režimu:

```
$ chmod [-R] absolute_mode objectname
```

Apsolutna prava formiraju se pomoću tri oktalne cifre kojima su predstavljena prava pristupa za vlasnika, grupu i ostatak sveta. Parametar -R se, kao i u simboličkom režimu, koristi za rekurzivnu promenu pristupnih prava direktorijuma i svih objekata koji se u njemu nalaze. U tom slučaju argument objectname mora biti direktorijum

Napomena: Kada se koristi oktalni režim moraju se navesti sve tri oktalne cifre, odnosno pristupna prava za sve tri vlasničke kategorije, u tačnom redosledu (vlasničko pravo - grupno pravo - pravo za ostatak sveta).

Sledeća dva primera ilustruju korišćenje oktalnog režima komande chmod. Prvi primer prikazuje zamenu postojećih pristupnih prava čitanja i upisa u svim kategorijama (666) pravima čitanja i izvršavanja (555). Drugi primer ilustruje ukidanje prava izmene datoteke denywrites kategorijama group i others.

```
$ ls -l betatest
-rw-rw-rw-   1   nm   nm   0   dec 23   15:25   betatest
$ chmod 555 betatest
$ ls -l betatest
-r-xr-xr-x   1   nm   nm   0   dec 23   15:25   betatest
$ ls -l denywrites
-rwxrwxrwx   1   nm   nm   0   dec 23   15:25   denywrites
$ chmod 755 denywrites
$ ls -l denywrites
-rwxr-xr-x   1   nm   nm   0   dec 23   15:25   denywrites
```

Promena vlasničkih odnosa

UNIX postavlja inicijalne vlasničke odnose prilikom kreiranja datoteke ili direktorijuma. Korisnik koji kreira objekat postaje njegov vlasnik, a objekat se formalno pridružuje primarnoj grupi vlasnika.

Promena vlasnika

Komandom chown komande (change owner) root kao superuser može da promeni vlasnika objekta, a ukoliko konkretan sistem to dozvoljava, to može učiniti i vlasnik. Regularni korisnici Linux sistema mogu promeniti vlasničke odnose samo ako na sistemu nije aktiviran mehanizam disk kvote (disk quota), kojim se korisnicima ograničava iskorišćenje prostora na diskovima. Poklanjanje datoteka u slučaju sistema sa kvotom moglo bi da izazove prepunjenje disk kvote za konkretnog korisnika. Kada se za datoteku promeni vlasništvo, prava pristupa starog vlasnika određena su kategorijama group i others. Sledeće komande prikazuju sintaksu za promenu vlasništva:

```
# chown [-R] new_owner objectname
```

Parametar -R koristi se za rekurzivnu promenu vlasništva nad direktorijumom i svim objektima koji se u njemu nalaze. Ukoliko se navede parametar -R argument objectname mora biti direktorijum. Primer ilustruje promenu vlasništva jedne datoteke.

```
$ whoami
nm
$ ls –l myfile
-rw-r--r--   1   nm       nm   0   Apr 28   12:07   myfile
$ chown jsmith myfile
chown: changing ownership of `myfile': Operation not permitted
$ su
Password: ******
# chown jsmith myfile
# exit
exit
$ ls –l myfile
-rw-r--r--   1   jsmith   nm   0   Apr 28   12:07   myfile
```

Primarne i sekundarne grupe

Svaki korisnik UNIX i Linux sistema mora pripadati najmanje jednoj grupi, čiji je GID naveden u opisu korisnika u datoteci /etc/passwd. Primarnu grupu korisnika određuje sistem administrator prilikom kreiranja korisničkog naloga. Primarna grupa je ona grupa kojoj operativni sistem dodeljuje datoteke i direktorijume koje korisnik kreira.

Sekundarne grupe su sve ostale grupe kojima korisnik pripada. Na ovaj način korisnicima je omogućeno da istovremeno pripadaju većem broju grupa u cilju ostvarivanja svojih prava pristupa. Pripadnost sekundarnim grupama reguliše se u datoteci /etc/group. Zavisno od verzije UNIX sistema korisnici mogu biti pripadnici do 8 ili do 16 sekundarnih grupa. Korisnik je ravnopravan član svih grupa.

Promena grupe kojoj objekat pripada

Nakon kreiranja objekat se formalno priključuje (odnosno, dodeljuje) primarnoj grupi korisnika koji je objekat kreirao. Superuser može bilo koju datoteku dodeliti nekoj grupi. Na Linux sistemima vlasnici mogu objekat dodeliti svojoj primarnoj grupi, dok na nekim UNIX sistemima vlasnik može datoteku pokloniti bilo kojoj grupi.

Datoteka se dodeljuje drugoj grupi komandom chgrp, čija je sintaksa:

```
$ chgrp [–R] new_group objectname
```

Parametar -R koristi se za rekurzivnu promenu grupnog vlasništva nad direktorijumom i svim objektima koji se u njemu nalaze. Ukoliko se navede parametar -R, argument objectname mora biti direktorijum.

Specijalni atributi datoteka i direktorijuma

setuid (SUID) i setgid (SGID)

Prilikom izvršavanja nekih programa postoji potreba za privremenim prisvajanjem identiteta drugog vlasnika ili grupe, što se na UNIX sistemima realizuje postavljanjem korisničkog bita (set user ID - setuid) i grupnog bita (set group ID - setgid).

Kada korisnik pokrene program kome je postavljen setuid bit, realni vlasnički odnosi se menjaju efektivnim. Korisnik koji izvršava program u procesu koji nastaje dobija identitet vlasnika izvršnog programa. Izvršavanje programa sa identitetom vlasnika može biti jako korisno. Na primer, običnom korisniku se na ovaj način može dozvoliti da kreira backup podataka bez otkrivanja lozinke superusera. Dovoljno je postaviti setuid bit programu za arhiviranje, nakon čega će korisnik moći da pročita sve datoteke i kreira backup.

Napomena: ukoliko je vlasnik programa superuser, postavljanje setuid bita može biti krajnje opasno. Na primer, ako rm (čiji je vlasnik root) ima postavljen setuid, tada svako ko ga pokrene dobija identitet superusera, što znači da može da obriše svaku datoteku.

Ukoliko je setuid bit izvršnog programa postavljen u kategoriji vlasnika se na mestu prava x nalazi oznaka s. Bit se postavlja komandom chmod, tako što se kategoriji vlasnika dodaje pravo s (chmod u+s progfile), a uklanja oduzimanjem prava s kategoriji vlasnika.

Sledeći primer ilustruje postavljanje setuid bita:

```
$ ls -l public_backup
-rwxr-xr-x   1   nm   nm   2344   dec 23   15:25   public_backup
$ chmod u+s public_backup
$ ls -l public_bekap
-rwsr-xr-x   1   nm   nm   2344   dec 23   15:25   public_backup
```

Grupni bit (setgid) postavlja se i funkcioniše na sličan način - realna grupa korisnika se u procesu zamenjuje efektivnom grupom, odnosno grupom kojoj izvršni program sa postavljenim setgid bitom pripada. Ukoliko je setgid bit izvršnog programa postavljen, u kategoriji grupe se na mestu prava x nalazi oznaka s. Bit se postavlja komandom chmod, tako što se kategoriji grupe dodaje pravo s (chmod g+s progfile), a uklanja oduzimanjem prava s kategoriji grupe.

Sledeći primer ilustruje postavljanje setgid bita:

```
$ ls -l public_backup
-rwxr-xr-x   1   nm   nm   2344   dec 23   15:25   public_backup
$ chmod g+s public_backup
$ ls -l public_bekap
-rwsr-sr-x   1   nm   nm   2344   dec 23   15:25   public_backup
```

Sticky bit (t)

Na javno dostupnim direktorijumima tipična prava pristupa su 777, što znači da svi korisnici mogu da kreiraju nove i brišu postojeće datoteke, bez obzira kome te datoteke pripadaju. Postavljanjem sticky bita za direktorijum uvodi se sledeće ograničenje: bez obzira na pravo upisa koje korisnik ima nad tim direktorijumom, on u njemu ne može

obrisati tuđe datoteke (odnosno datoteke kojima on nije vlasnik). Tipičan primer je sistemski direktorijum /tmp. Sticky bit se postavlja i ukida komandom chmod tako što se u simboličkom režimu svim vlasničkim kategorijama dodeli pravo t (chmod +t directory), a ukida oduzimanjem prava t. Sledeći primer ilustruje postavljanje, identifikaciju i ukidanje sticky bita:

```
$ ls -l public_dir
-rwxrwxrwx   1   nm    nm   4096   dec 23   15:25   public_dir1
$ chmod +t public_dir1
$ ls -l public_dir1
-rwxrwxrwt   1   nm    nm   4096   dec 23   15:25   public_dir1
$ chmod -t public_dir1
-rwxrwxrwx   1   nm    nm   4096   dec 23   15:25   public_dir1
```

Dodatno, sticky bit se može postaviti komandom chmod u oktalnom režimu, navođenjem cifre 1 pre pristupnih prava (chmod 1777 directory).

```
$ ls -l public_dir2
-rwxrwxrwx   1   nm   nm   4096   dec 23   15:25   public_dir2
$ chmod 1777 public_dir2
$ ls -l public_dir2
-rwxrwxrwt   1   nm   nm   4096   dec 23   15:25   public_dir2
```

Specijalni atributi datoteka na ext2/ext3 sistemu datoteka

Specijalni atributi datoteka na ext2 i ext3 sistemu datoteka su:

A	Don´t update access time - zabranjuje izmenu vremena poslednjeg pristupa (štedi se na disk I/O operacijama na laptop sistemima);
a	Append only - dozvoljava isključivo dodavanje novih podataka u datoteku, ali ne i izmenu ili brisanje starih, ukoliko je datoteka otvorena u režimu čitanja. Samo superuser može postaviti ili obrisati ovaj atribut;
c	Compressed - datoteka sa atributom c smešta se na disk u komprimovanom obliku, pri čemu kompresiju obavlja kernel. Prilikom čitanja, kernel najpre obavlja dekompresiju datoteke;
d	No dump - datoteka nije kandidat za backup koji se vrši komandom dump;
i	Immutable - datoteka ne može biti modifikovana ili obrisana, ne može joj se promeniti ime niti se može kreirati link koji ukazuje na tu datoteku;
j	Data journalling - svi podaci se prvo ažuriraju u dnevniku, a zatim u samoj datoteci, pod uslovom da se koristi ordered ili writeback režim dnevnika. Ovo je podrazumevano u slučaju journal dnevnika;
s	Secure deletion - Prilikom brisanja datoteke u sve blokove koji čine datoteku upisuju se nule;
S	Sync - prilikom izmene sadržaja datoteke promene se odmah upisuju na disk (izmene sadržaja datoteke se ne keširaju);

t Tail-merge - neiskorišćeni fragmenti poslednjeg bloka datoteke ne mogu se dodeliti drugoj datoteci na korišćenje;

u Undelete - prilikom brisanja datoteke čuva se njen sadržaj, čime je omogućen povratak obrisane datoteke;

Specijalni atributi mogu se dodeliti datotekama komandom chattr, čija je opšta sintaksa:

```
# chattr [-R] mode files
```

Parametar mode se zadaje u simboličkom režimu (slično kao pristupna prava u simboličkom režimu komande chmod). Format parametra mode je:

```
+-=[ASacdistu]
```

Operator + znači da se postojećoj listi atributa dodaju atributi navedeni u komandnoj liniji, dok se operatorom - od postojeće liste atributa oduzimaju atributi navedeni u komandnoj liniji. Operator = se koristi za dodelu tačno određenog skupa atributa datoteci. Atributi su predstavljeni slovima "ASacdijsu".

Opcija -R se koristi za rekurzivnu promenu atributa celokupnog sadržaja direktorijuma (datoteka i poddirektorijuma) koji je naveden kao argument komande.

Specijalni atributi datoteka mogu se videti pomoću komande lsattr. Na primer, ovako izgleda datoteka file1 koja ima flegove c,d i i (Compressed, No Dump i Immutable):

```
$ lsattr file1
---i-d-c------ file1
```

Disk kvote

Šta su disk kvote ? Administracija kvota na Linux sistemu.

Najjednostavnije rečeno, kvota je ograničenje koje sistem administrator dodeljuje korisnicima, a koje se tiče stepena iskorišćenja prostora na sistemima datoteka. Kvote se mogu postaviti na sledeći način:

- limitiranjem broja indeksnih čvorova (odnosno broja datoteka) koje korisnici ili grupe mogu imati u okviru sistema datoteka za koji je kvota postavljena,
- limitiranjem broja blokova na disku (vrednost u kilobajtima) koji se mogu dodeliti korisnicima ili grupama.

Kvota ograničava korisnike, odnosno sprečava ih da koriste neograničenu količinu slobodnog prostora u sistemu datoteka, što je jako značajno ukoliko se radi o nekom file serveru. Takođe, kvotama se mogu ograničiti veličine mail boxova korisnika. Na primer, kvota od 10MB može se dodeliti svim korisnicima za particiju /var, nakon čega korisnici mogu u direktorijumu /var/spool/$LOGNAME sačuvati najviše 10MB.

Administracija kvota na Linux sistemu

Preduslovi za postavljanje kvota

Prva stvar koju superuser treba da uradi je da obezbedi podršku za kvotu u kernelu. Ukoliko kernel to ne podržava potrebno je prevesti novo jezgro (videti poglavlje o konfigurisanju Linux kernela). U verziji 2.4 Linux kernela podrška za kvotu uključuje se potvrdnim odgovorom na pitanje:

```
Filesystems – Quota support (CONFIG_QUOTA) [N/y/?] y
```

Nakon toga, potrebno je proveriti da li je quota paket instaliran na sistem (videti poglavlje o instaliranju softverskih paketa - Red Hat Package Manager):

```
# rpm -q quota
package quota is not installed
```

Ukoliko nije, potrebno je da se instalira sa instalacionog diska. Na primer, u slučaju Red Hat Linux distribucije kvota se sa CD-ROM uređaja (montiran na direktorijum /mnt/cdrom) može instalirati na sledeći način:

```
# rpm -Uvh /mnt/cdrom/RedHat/RPMS/quota-version.i386.rpm
```

Postavljanje kvota

Kvota se može postaviti za korisnike, grupe ili korisnike i grupe. Za sve primere koji slede kvote su postavljene za /home sistem datoteka (uređaj /dev/sda2).

Postavljanje kvota vrši se u nekoliko koraka. Najpre je potrebno modifikovati datoteku /etc/fstab, koja sadrži informacije o različitim sistemima datoteka na Linux sistemu. Kvota se mora postaviti posebno za svaki sistem datoteka, odnosno mora biti dopisana u svaku liniju datoteke /etc/fstab koja predstavlja sistem datoteka za koji želimo da postavimo kvote.

Primeri redom ilustruju uključivanje podrške za kvotu:

- za korisnike

```
LABEL=/home /home ext2 defaults,nosuid,usrquota 1 2
```

- za grupe

```
LABEL=/home /home ext2 defaults,nosuid,grpquota 1 2
```

- za grupe i korisnike

```
LABEL=/home /home ext2 defaults,nosuid,usrquota,grpquota 1 2
```

Nakon modifikacije sadržaja datoteke /etc/fstab potrebno je obavestiti sistem o izmenama, odnosno reaktivirati sistem datoteka za koji se postavljaju kvote:

```
# mount -oremount /home/
```

Datoteke quota.user i quota.group

Sledeći korak u administraciji kvota predstavlja kreiranje datoteka quota.user, odnosno quota.group, koje redom predstavljaju tabele kvota za korisnike i grupe. U root direktorijumu sistema datoteka za koji se kvota postavlja potrebno je kreirati datoteku:

- quota.user, ukoliko se kvota dodeljuje korisnicima,
- quota.group, ukoliko se kvota dodeljuje grupama,
- obe datoteke, ukoliko se kvota dodeljuje i korisnicima i grupama.

Vlasnik ovih datoteka mora biti superuser. Ove datoteke se za potrebe našeg primera kreiraju u direktorijumu /home, jer kvote postavljamo za home direktorijume korisnika, odnosno za /home sistem datoteka:

```
# touch /home/quota.user
# chmod 600 /home/quota.user
# touch /home/quota.group
# chmod 600 /home/quota.group
```

Komanda touch kreiraće nove prazne datoteke quota.user i quota.group u /home/ direktorijumu. Komanda chmod dodeljuje read i write pravo vlasniku (root), dok su ostalima ukinuta sva prava (600, odnosno -rw-------).

Napomena: potrebno je kreirati samo jednu datoteku ukoliko se kvota dodeljuje korisnicima ili grupama. U slučaju da se kvota dodeljuje i korisnicima i grupama, kreiraju se obe datoteke.

Nakon kreiranja potrebnih datoteka superuser može dodeliti kvote korisnicima i/ili grupama pomoću alata edquota, kao što je prikazano u sledećem primeru.

Dodeljivanje kvote korisniku

Pretpostavljamo da na sistemu postoji korisnički nalog jsmith. Sledećim komandama pokreće se editor za podešavanje kvote za korisnika jsmith na svakoj particiji za koju je kvota postavljena:

```
# edquota –u jsmith
Quotas for user jsmith:
/dev/sda2: blocks in use: 0, limits (soft = 0, hard = 0)
inodes in use: 0, limits (soft = 0, hard = 0)
```

Značenje linija je sledeće:

blocks in use: x	ukupan broj blokova (u kilobajtima) koje je korisnik upotrebio na particiji. Parametar kontroliše sam operativni sistem;
inodes in use:	ukupan broj datoteka koje je korisnik smestio na particiju. Parametar kontroliše sam operativni sistem.

Korisniku jsmith može se dodeliti kvota od 5MB na particiji /dev/sda2 na sledeći način:

```
Quotas for user jsmith:
/dev/sda2: blocks in use: 0, limits (soft = 5000, hard = 6000)
```

```
inodes in use: 0, limits (soft = 0, hard = 0)
```

Značenje parametara soft i hard limit je sledeće:

soft limit	određuje najveću količinu prostora na sistemu datoteka koju korisnik može da iskoristi za smeštanje svojih datoteka. Kako je u ovom primeru soft=5000, korisnik jsmith će na particiju /dev/sda2 (home sistem datoteka) moći da smesti najviše 5MB svojih datoteka;
hard limit	apsolutno ograničenje. Korisnik ne može ni na koji način da pređe ovaj limit. Hard limit vrednost ima smisla samo ako je postavljen parametar grace period.

Parametrom grace period (period milosti) postavlja se vremenska granica pre nasilne primene vrednosti parametra soft limit. Na primer, ovaj parametar se može koristiti da upozori korisnike o novoj administrativnoj polisi kojom će se korisnicima dodeliti kvota od 5MB prostora nakon sedam dana. Komandom edquota -t određuju se softtime ograničenja za svaki sistem na kom je uključena kvota. Grace period parametar može se definisati za iskorišćeni prostor i ukupan broj datoteka na sledeći način:

```
# edquota -t
Time units may be: days, hours, minutes, or seconds
Grace period before enforcing soft limits for users:
/dev/sda2: block grace period: 7 days, file grace period: 7 days
```

Dodeljivanje kvote grupama

Sledeći primer ilustruje dodelu kvota grupi allusers, pod pretpostavkom da data grupa postoji na sistemu:

```
# edquota -g allusers
Quotas for group allusers:
/dev/sda2: blocks in use: 0, limits (soft = 5000, hard = 6000)
inodes in use: 0, limits (soft = 0, hard = 0)
```

Postupak dodele kvota grupama sličan je postupku dodele kvota korisnicima i svodi se na dodelu vrednosti parametrima soft limit, hard limit i grace period.

Dodeljivanje jednakih kvota većem broju korisnika

Specijalnom opcijom -p alata edquota većem broju korisnika mogu se dodeliti kvote identične kvotama koje su dodeljene nekom drugom korisniku. Na primer, svim korisnicima sistema čiji je UID 500 ili veći, mogu se dodeliti kvote identične kvoti korisnika jsmith.

```
# edquota -p jsmith `awk -F: '$3 > 499 {print $1}' /etc/passwd`
```

Ostali alati za rad sa kvotama

Komande koje su prethodno navedene najčešće se koriste, ali za administraciju kvota postoji mnogo više komandi (quota, repquota, quotactl, quotaon ...).

Program quota prikazuje zauzeće diska od strane određenog korisnika ili grupe i ograničenja u sistemu datoteka. Komanda se zadaje na sledeći način:

```
# quota –u user
# quota –g group
```

gde su user i group imena korisnika ili grupe čije kvota parametre program prikazuje.

Alat repquota daje sažete informacije o zauzeću diska i kvotama za navedeni sistem datoteka, trenutnom broju datoteka i zauzeću diska za svakog korisnika kome su dodeljene kvote. Program se pokreće na sledeći način:

```
# repquota option
```

Moguće opcije su:

-a prikazuje izveštaj o svim sistemima datoteka koji su u datoteci /etc/fstab naznačeni kao read-write za koje su postavljene kvote

-g prikazuje izveštaj o kvotama za grupe

-u prikazuje izveštaj o kvotama za korisnike (podrazumevana opcija)

Alat quotacheck analizira potrošnju datoteka i direktorijuma na odgovarajućem sistemu datoteka i na osnovu toga kreira odgovarajuće datoteke quota.user i quota.group. Sintaksa komande je:

```
# quotacheck [–u] [–g] [–a|filesystem]
```

Alatima quotaon i quotaoff mogu se aktivirati i deaktivirati kvote za određeni sistem datoteka. Quotaon zahteva da na sistemu budu prisutne datoteke quota.user ili quota.group, nakon čega se može zadati komanda:

```
# quotaon [–ug] filesystem
```

Sličnom komandom kvote se mogu deaktivirati:

```
# quotaoff [–ug] filesystem
```

Parametri -u i -v odnose se na korisničke i grupne kvote. Ukoliko se umesto ovih parametara navede -a, aktiviraće se ili deaktivirati kvote na svim read-write sistemima datoteka za koje je u datoteci /etc/fstab naznačena upotreba kvota:

```
# quotaon –a
# quotaoff –a
```

Na ovaj način kvote se aktiviraju prilikom podizanja sistema.

5

RAD SA DATOTEKAMA IZ KOMANDNE LINIJE

Grafičko radno okruženje korisnike UNIX i Linux sistema obezbeđuje potpunim skupom alata za rad sa datotekama. Ovoj grupi alata pripadaju razni file manager alati kao što su Konqueror i Krusader, razni tekst editori i slični alati koji su jednostavni za korišćenje. Grafičko radno okruženje opterećuje procesor i povećava rizik u smislu sigurnosti sistema, tako da se, po pravilu, ne instalira na serverima. Tada sistem administratorima na raspolaganju ostaje komandni interpreter (shell) i prateći skup alata za rad sa datotekama, čije je poznavanje neophodan uslov za uspešnu administraciju UNIX i Linux servera.

Komandni interpreter (shell)

Šta je komandni interpreter koje su mu funkcije? Najčešće korišćeni komandni interpreteri.

Shell je interfejs između korisnika i kernela, odnosno jezgra operativnog sistema. Shell prihvata komande koje korisnik zadaje, zatim ih interpretira i potom ih izvršava, pri čemu po potrebi pokreće odgovarajuće programe. Na UNIX sistemima postoji više različitih komandnih interpretera, a korisnici u toku rada po potrebi mogu preći iz jednog u drugi.

Funkcije komandnog interpretera

Komandni interpreter je proces koji obavlja sledeće funkcije u cilju obezbeđivanja interfejsa između korisnika i operativnog sistema:

- interpretaciju komandne linije,
- pokretanje programa,
- redirekciju ulaza i izlaza,
- povezivanje komandi u pipeline,

- zamenu imena datoteka,
- rukovanje promenljivim i kontrolu okoline,
- shell programiranje.

Interpretacija komandne linije

Dve osnovne prednosti korišćenja komandnih interpretera u odnosu na grafičko radno okruženje su: fleksibilnost u radu i mogućnost pristupa serveru sa udaljenog terminala (na kom ne postoji grafičko okruženje). Veliki broj administrativnih programa (programi za upravljanje operativnim sistemimom i uređajima) zahtevaju da korisnik poznaje i razume UNIX komande. U nekim slučajevima programi koji rade iz komandne linije su jedini dostupan alat.

Kad se korisnik prijavi na sistem u kontekstu tekućeg login procesa izvršava se proces shell, odnosno komandni interpreter. Na ekranu se prikazuje komandni prompt (shell prompt), a to je najčešće znak $, ukoliko se na sistem prijavi običan korisnik, odnosno #, ukoliko se na sistem prijavi root. Kada korisnik zada neku komandu (odnosno otkuca neki tekst i pritisne Enter), shell to pokušava da interpretira. Tekst unet u shell prompt naziva se komandna linija (command line), čiji je opšti oblik:

```
$ command [opcije] [argumenti]
```

Znak $ je odzivni znak komandnog interpretera (shell prompt) i prikazuje se na ekranu svaki put kad je komandni interpreter spreman da od korisnika prihvati novu komandu. Zatim sledi command, odnosno komanda koja se izvršava. Komanda može biti interna (ugrađena u shell) ili eksterna (realizovana kao poseban program koji se nalazi u sistemskoj putanji). Opcije i argumenti su parametri koje shell prenosi komandi, pri čemu su argumenti najčešće obavezni i predstavljaju ime neke datoteke, direktorijuma, korisnika ili, na primer, identifikator procesa.

Ime komande, opcije i argumenti razdvajaju se razmakom. Shell interpretira razmak kao graničnik i na osnovu toga razdvaja argumente i opcije od imena komande. U jednu komandnu liniju može se uneti najviše 256 karaktera. Imena većine UNIX komandi po pravilu se formiraju od malih slova (izuzeci su razni shell programi, poput /dev/MAKEDEV).

Više UNIX komandi mogu se navesti u istoj komandnoj liniji ukoliko su razdvojene znakom tačka-zarez.

U nastavku teksta dati su primeri komandne linije:

```
$ cal               # samo komanda
$ df /dev/sda       # komanda (fd) i argument (/dev/sda)
$ cp 1.txt 2.txt    # komanda (cp) i dva argumenta (1.txt i 2.txt)
$ date -u           # komanda (date) i opcija (-u)
$ ls -l /etc        # komanda (ls), opcija (-l) i argument (/etc)
$ clear ; date      # dve komande koje se izvršavaju jedna za drugom
```

Opcije su osetljive na velika i mala slova (case-sensitive) i mogu se navesti u jednom od sledeća dva oblika:

-x znak minus (-) praćen jednim slovom,

--option dva znaka minus (--) praćena punim imenom opcije.

Dodatno, komandni interpreter eliminiše nepotrebne informacije (komentare i beli prostor) iz komandne linije. Na primer, shell će sledeću komandnu liniju interpretirati kao komandu sa šest argumenata između kojih će ukloniti beli prostor (whitespace):

```
$ echo A   B   C   1 2 3
A B C 1 2 3
```

U sledećem primeru se beli prostor ne uklanja jer je argument komande echo stavljen pod apostrofe i interpretira se kao niz karaktera:

```
$ echo 'A   B   C   1 2 3'
A   B   C   1 2 3
```

Inicijalizacija programa

Nakon interpretacije komandne linije shell inicira izvršenje zadate komande. Ukoliko komanda nije interna (ugrađena u shell, poput komande cd) shell traži izvršnu datoteku koja odgovara imenu komande u direktorijumima navedenim u sistemskoj putanji. Nakon toga shell pokreće program i prosleđuje mu argumente i opcije navedene u komandnoj liniji.

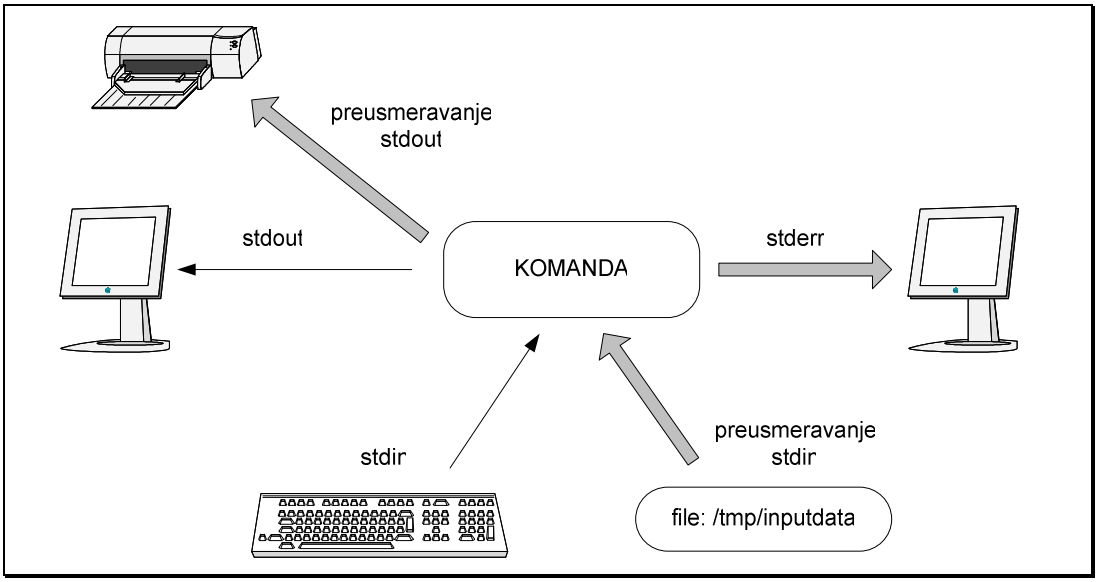
Napomena: Ukoliko se izvršna datoteka nalazi u tekućem direktorijumu ili u nekom direktorijumu koji nije u sistemskoj putanji, ime komande se mora zadati sa putanjom. Sledeći primeri ilustruju pokretanje programa koji se nalaze u tekućem direktorijumu i direktorijumu /usr/sbin:

```
$ ./myscript
$ /usr/sbin/useradd
```

Redirekcija ulaza i izlaza

Operacije koje procesor izvršava ponekad zahtevaju izvršenje ulazno/izlaznih operacija. Na primer, korisnik pomoću tastature unosi podatke programu za obračun plata, nakon čega program vrši obračun i na ekranu prikazuje rezultat i po potrebi obaveštenja o greškama.

UNIX komande primaju podatke sa standardnog ulaza (stdout), rezultate izvršenja šalju na standardni izlaz (stderr), a poruke o greškama na standardni uređaj za greške (stderr). Većina UNIX komandi koristi tastaturu kao standardni ulaz, a monitor kao standardni izlaz i uređaj za greške. UNIX omogućava da se ulaz i izlaz komande preusmere. To znači da komanda može čitati ulazne podatke iz datoteke (na primer, shell skript koji kreira 100 korisničkih naloga), odnosno da se izlaz komande može sačuvati kao datoteka ili odštampati radi kasnijeg čitanja. Na slici 5.1 prikazana je komanda čiji je standardni ulaz preusmeren na datoteku, a standardni izlaz na štampač.



Slika 5.1 *Redirekcija standardnog ulaza i izlaza*

Ulaz komande preusmerava se pomoću znaka < (manje od) na sledeći način:

```
$ command < inputdevice
```

Inputdevice može biti datoteka ili ulazni uređaj (preusmeriti ulaz na izlazni uređaj kao što je štampač nema nikakvog smisla).

Na primer, moguće je standardni ulaz sa tastature zameniti nekom tekstualnom datotekom:

```
$ wc -l < /tmp/jsnmith.dat
```

Izlaz se češće preusmerava od ulaza. Komande poput ls i sort koriste monitor kao standardni izlaz, odnosno prikazuju rezultat izvršenja na monitoru bez pauze nakon svakog punog ekrana. Ukoliko postoji potreba da se rezultat izvršenja ovih komandi pregleda kasnije ili se zbog brzog skrolovanja uopšte ne može pregledati, potrebno je preusmeriti izlaz ovih komandi na datoteku ili štampač. Za redirekciju se koristi znak > (veće od). Ukoliko se redirekcija vrši u postojeću datoteku datoteka se briše, a zatim se kreira nova u koju se smešta rezultat izvršenja komande. Ukoliko korisnik želi da se rezultat izvršenja doda na postojeću datoteku bez brisanja njenog sadržaja, za redirekciju izlaza koristi se znak >>. Sledeći primer ilustruje redirekciju izlaza na štampač, u novu datoteku i postojeću datoteku i kreiranje prazne datoteke.

```
$ sort kyuss.txt > /dev/lp0
$ ls -l /home/jsmith > myfile
$ ls -l /tmp/jsmith >> myfile
$ >emptyfile
```

Standardni izlaz za greške preusmerava se ukoliko korisnik želi da sačuva rezultat izvršenja komande u nekoj datoteci radi kasnije analize grešaka (na primer, debagovanje programa u fazi razvoja). Izlaz se uglavnom preusmerava u tekstualne datoteke pomoću znaka 2>.

```
$ ./testprogram 2> debugging.txt
```

Povezivanje komandi u pipeline

UNIX je razvijen kao operativni sistem čije komande izvršavaju jednostavne, jasno definisane zadatke, a čijim se kombinovanjem mogu izvršiti komplikovani poslovi. Jedna od funkcija komandnog interpretera koja omogućava povezivanje komandi je pipeline. Pipeline funkcioniše na sledeći način: standardni izlaz komande sa leve strane znaka pipe - cev (|) postaje standardni ulaz komande sa desne strane znaka. Znak pipe zahteva komande i sa leve i sa desne strane, a razmaci između znaka i komande su proizvoljni.

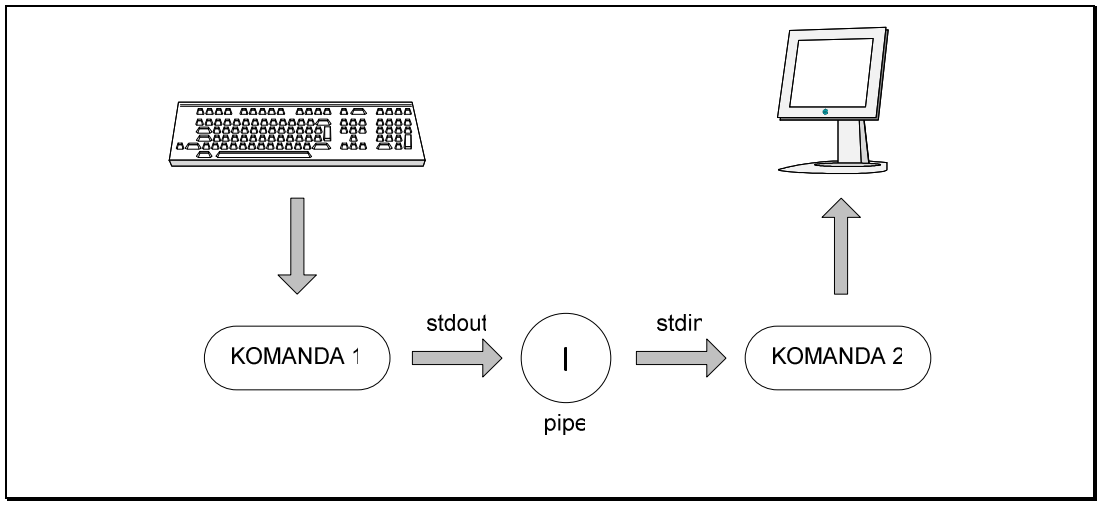
Na primer, pretpostavimo da korisnik želi da zna koliko datoteka ima u direktorijumu u /etc. Korisnik može izlistati sadržaj direktorijuma u takozvanom long-listing formatu komandom ls -l /etc i izvršiti redirekciju izlaza u privremenu datoteku /tmp/files_in_etc. Dalje, korisnik može prebrojati broj linija u privremenoj datoteci pomoću komande wc -l /tmp/files_in_etc (word count).

```
$ ls –l /etc > /tmp/files_in_etc
$ wc –l < /tmp/files_in_etc
145
```

Prva komanda koristi datoteku /tmp/ files_in_etc kao standardni izlaz, a druga kao standardni ulaz, što znači da se prebrojavanje datoteka u direktorijumu /etc može realizovati pomoću pipeline sprege komandi ls i wc:

```
$ ls –l /etc | wc –l
145
```

Princip funkcionisanja pipeline sprege prikazan je na slici 5.2.



Slika 5.2 *Povezivanje komandi u pipeline*

Opšta sintaksa pipeline sprege je sledeća:

```
$ command1 | command2 | .... | commandN
```

To znači da se u pipeline može povezati veći broj komandi (maksimalan broj zavisi od konkretnog UNIX sistema, a obično se kreće od 20 do 30). Sledeći primer ilustruje pipeline koji prebrojava sve korisnike koji su prijavljeni na sistem a ime im počinje slovom a i prikazuje ih sortirane po abecednom redu.

```
$ finger | grep a* | wc -l | sort
```

I pipeline i redirekcija kao funkcije komandnog interpretera imaju svoje prednosti. Preusmeravanje se koristi kad se koristi podatak iz datoteke koja će se i dalje koristiti. U suprotnom se koristi pipe, čijom se upotrebom sprečava nepotrebno gomilanje podataka na diskovima.

Zamena imena datoteka

Imena datoteka mogu se zameniti džoker karakterima (joker) *, ? i []. Argument komande koji sarži džoker karakter zamenjuje se odgovarajućom listom datoteka shodno pravilima zamene. Komandni interpreter izvršava ovu zamenu pre izvršavanja same komande, odnosno pre pokretanja programa. Sledeći primer ilustruje korišćenje džoker karaktera:

```
$ echo *
myfile1 kyuss.txt file3 anotherfile3 file4
```

Karakter * zamenjuje se imenima svih datoteka u tekućem direktorijumu.

Pravila zamene jednostavno se mogu objasniti na sledeći način:

- karakter * menja bilo koji niz znakova proizvoljne dužine

```
# ls -d /var/s*
/var/spool          /var/stat
```

- karakter ? menja bilo koji znak (tačno jedan znak)

```
# ls -d /?bin
/sbin
```

- opseg [poc-kraj] menja tačno jedan znak koji pripada tom opsegu. Opseg se ne sme zadati u opadajućem redu.

```
# ls -d /etc/[a-d][a-d]*
/etc/adduser.conf  /etc/bash.bashrc      /etc/bash_completion.d
/etc/adjtime       /etc/bash_completion  /etc/calendar
```

Rukovanje promenljivim i kontrola okruženja

Da bi komandni interpreter bio fleksibilniji i lakši za korišćenje, u shell je uveden koncept okruženja. Okruženje je skup promenljivih (kao što je, na primer, sistemska putanja) čije vrednosti korisnici mogu menjati i na taj način prilagoditi radno okruženje svojim potrebama. Dodatno, korisnici mogu definisati nove promenljive i brisati postojeće.

Shell programiranje

Shell se najčešće koristi na sledeći način: korisnik zada komandu, shell interpretira komandnu liniju, nakon čega izvršava unetu komandu i po potrebi izvršava neki program. Komandni interpreter takođe se može koristiti i kao programski jezik. Kombinovanjem komandi sa vrednostima promenjivih i strukturama za kontrolu toka, shell postaje moćan programski alat. Korišćenjem komandnog interpretera kao programskog alata mogu se automatizovati razni administrativni zadaci koji zahtevaju od korisnika da unese veliki broj komandi.

Dodatne mogućnosti Bourne-again shella

Korišćenje kontrolinih karaktera

Kontrolni karakteri koriste se za izvršavanje raznih specifičnih zadataka, kao što su prekidanje izvršenja procesa koji radi u prvom planu i modifikacija komandne linije. Kontrolni karakteri se zadaju pomoću tastera <Ctrl> koji se na ekranu prikazuje kao simbol ^ (carret). Korisnik zadaje kontrolne karaktere tako što istovremeno pritisne taster <Ctrl> i još jedan karakter. Kontrolni karakteri Bourne-again shella koji se najčešće koriste su:

<Ctrl-c> prekida izvršenje procesa koji radi u prvom planu;

<Ctrl-d> označava kraj datoteke. Koristi se za napuštanje programa koji podatke čitaju sa standardnog ulaza (tastatura). Sledeći primer ilustruje korišćenje kontrolnih karaktera u programu bc (basic calculator):

```
$ bc            # pokreće program bc
100/5           # inicira operaciju deljenja
20              # program bc prikazuje rezultat prethodne operacije
<Ctrl-d>        # napuštanje programa i povratak u shell
$
```

<Ctrl-u> briše celu komandnu liniju;

<Ctrl-w> briše zadnju reč u komandnoj liniji;

<Ctrl-s> privremeno zaustavlja izvršenje procesa u prvom planu. Može se koristiti prilikom pregledanja sadržine nekog velikog direktorijuma komandom ls ili ukoliko se neka datoteka prikazuje na ekranu programom cat;

<Ctrl-g> nastavlja se izvršenje procesa u prvom planu.

Alternativno ime komande (alias)

Alias je način dodele kraćeg imena pomoću kog se određena komanda, ili niz komandi, može pozvati iz komandnog interpretera. Na primer, može se dodeliti alias ll (long listing) koji izvršava komandu ls –l. Alias je aktivan samo u komandnom interpreteru za koji je napravljen. Za korn i bash alias se dodeljuje na sledeći način:

```
$ alias aliasname=value
```

Napomena: razmak postoji između komande alias i alternativnog imena, ali se ne stavlja ni sa jedne strane znaka jednakosti. Ukoliko se alternativno ime dodeljuje nizu komandi ili komandama koje imaju argumente, razmake ili specijalne karaktere, originalna komanda se mora staviti pod navodnike.

Sažeto rečeno, postoji nekoliko razloga za korišćenje alternativnih imena:

- komandi se može dodeliti kraće alternativno ime

```
$ alias h=history
$ alias c=clear
```

- jednom komandom se može zameniti sekvenca komandi

```
$ alias home="cd;ls"
```

- može se kreirati jednostavno ime za izvršavanje komandi sa određenim parametrima

```
$ alias ls="ls -l"
$ alias copy="cp -i"
```

Postojeći skup alternativnih imena može se prikazati na ekranu zadavanjem komande alias bez argumenata. Neki aliasi su unapred dodeljeni kao sastavni delovi korn ili bash komandnih interpretera.

```
$ alias
alias c='clear'
alias copy='cp -i'
alias h='history'
alias home='cd;ls'
alias l.='ls -d .[a-zA-Z]* --color=tty'
alias ll='ls -l --color=tty'
alias ls='ls -l'
alias vi='vim'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-
dot --show-tilde'
```

Dodeljeni aliasi mogu se ukloniti na način koji je prikazan u sledećem primeru. Nakon uklanjanja alias više ne postoji.

```
# alias myroot="ls -ld /root"
# myroot
drwxr-x---   7   root   root   1024   May 4  17:41   /root
# unalias myroot
# myroot
bash: myroot: command not found
```

Ponavljanje komandne linije (history)

Komandni interpreter bash upisuje svaku komandnu liniju u history datoteku. Ovo omogućava da se prethodne komande ponove, pri čemu se pre ponovnog izvršavanja mogu i izmeniti. Komande se takođe mogu ponavljati na osnovu rednog broja koji im je pridružen u history datoteci. Bash shell history datoteku smešta u home direktorijum korisnika (~/.bash_history), i u njoj podrazumevano čuva 1000 prethodno izvršenih

komandi. Broj komandi koje se mogu smestiti u ovu datoteku može se promeniti pomoću promenljive HISTSIZE - na primer, ako je HISTSIZE=500, to znači da se u datoteku ~/.bash_history mogu smestiti 500 prethodno izvršenih komandi.

Komanda history u bash shellu prikazuje prethodno izvršene komande:

```
$ date
$ whoami
$ finger
$ mail
$ history 5
  329  date
  330  whoami
  331  finger
  332  mail
  333  history 5
```

Najjednostavniji način za ponavljanje komandi je korišćenje gornje strelice (ili kombinacije tastera <Ctrl+p>) i donje strelice (ili kombinacije tastera <Ctrl+n>). Ovim tasterima korisnik bira jednu komandu iz datoteke ~/.bash_history, pri čemu se počinje od poslednje zadate komande. Nakon prikazivanja komande na ekranu korisnik može pritisnuti <Enter> i komanda će opet biti izvršena.

Dodatno, korisnik koji želi da unese novu komandu sa parametrima prethodne, može koristiti sledeće specijalne karaktere:

- !! izvršava ponovo prethodnu komandu,
- !* za izvršenje nove komande sa argumentima predhodne komande,
- !$ za izvršenje nove komande sa poslednjim argumentom prethodne komande.

Upotreba ovih specijalnih karaktera ilustrovana je sledećim primerom:

```
# date
Thu May 13 20:19:03 CEST 2004
# !!
date
Thu May 13 20:19:06 CEST 2004
# chown -R root /home/jsmith
# chgrp !*
chgrp -R root /home/jsmith
# ls -d /root
root
# dir !$
dir /root
oldfile     newusers     mydoc
```

Modifikacija komandne linije

Kao što je već rečeno, korisnik može u bash shellu ponoviti neku od prethodno izvršenih komandi. Ukoliko postoji potreba, korisnik je može modifikovati. Pri tome se pomeranje kursora u komandnoj liniji vrši pomoću kursorskih tastera (leva i desna strelica) ili kombinacijom tastera <Ctrl+b> i <Ctrl+f>.

Kompletiranje imena datoteka

Bourne-again shell sadrži mogućnost za kompletiranje imena datoteka. Korisnik može da unese samo nekoliko prvih karaktera imena datoteke, a zatim da pritisne karakter <Tab> kojim se naznačava shellu da završi ime datoteke. Na primer:

```
$ ls –l /etc/pas<Tab>
$ ls –l /etc/passwd
```

Ukoliko shell u tekućem direktorijumu pronađe više od jedne datoteke čije ime počinje tim karakterima, korisnik će morati da unese još nekoliko karaktera u imenu datoteke, a zatim da ponovo pritisne taster <Tab>.

Dodatno, ako korisnik dva puta pritisne <Tab>, shell će prikazati listu datoteka čije ime odgovara početku imena koje je korisnik uneo.

Shell promenljive i prilagođeni prompt

Shell promenljiva je mesto u operativnoj memoriji na kom se čuvaju informacije koje koriste procesi. Svaka promenljiva ima ime i vrednost. Promenljive se mogu klasifikovati na:

- lokalne promenljive (local variables), koje su dostupne samo u tekućem shellu,
- promenljive okruženja (environment variables), koje su dostupne u tekućem shellu i svim komandnim interpreterima koji su nastali kao podprocesi tekućeg shella.

Lokalne promenljive

Lokalna promenjiva je dostupna samo u tekućem shellu, odnosno u shellu u kom je definisana. Lokalne promenljive mogu se učiniti dostupnim u drugim shell instancama ukoliko se izvezu (export). Vrednost se dodeljuje promenljivoj pomoću komande VARIABLE=value, gde je VARIABLE ime promenljive, a value vrednost:

```
$ VARIABLE=value
```

Komandom unset, koja se retko koristi, promenjiva se uklanja iz tekuće shell instance:

```
$ unset VARIABLE
```

Vrednost promenljive može se prikazati pomoću komande echo. Na primer, echo $EDITOR će na ekranu prikazati emacs ukoliko je to vrednost promenljive. Znak $ kao specijalni karakter naglašava komandnom interpreteru da je EDITOR promenljiva, a ne niz karaktera. Ukoliko korisnik zada komandu echo EDITOR, shell će komandi echo proslediti EDITOR kao niz karaktera. Sledeći primer ilustruje prikazivanje vrednosti promenljive:

```
$ echo HOME
HOME
$ echo $HOME
/home/nm
```

Vrednosti svih promenljivih koje su vidljive iz tekućeg shell-a mogu se prikazati komandom set:

```
$ set
BASH=/bin/bash
BASH_VERSINFO=([0]="2" [1]="05a" [2]="0" [3]="1" [4]="release"
[5]="i686-pc-linux-gnu")
BASH_VERSION='2.05a.0(1)-release'
...
UID=859
USER=nm
_=history
langfile=/home/nm/.i18n
```

Promenljive okruženja

Promenljive okruženja su globalno dostupne, odnosno vidljive iz svih shell instanci. Lokalna promenljiva se može učiniti globalno dostupnom pomoću komande export. Globalna dostupnost podrazumeva vidljivost promenljive iz svih komandnih interpretera koji su podprocesi shella iz kog je komanda "eksportovana". Na primer, vrednost promenljive PS1 je MyPromt$ i ona je dostupna u tekućem shellu, što znači da će odziv (prompt) komandne linije biti MyPrompt$. Ako korisnik pokrene drugu instancu komandnog interpretera (na primer, ksh), shell prikazuje samo podrazumevani promt $. Ako se promenljiva PS1 izveze (export), ona će biti dostupna u svim shell instancama - svaki pokrenuti shell prikazivaće MyPrompt$ kao odziv. Promenljive se mogu izvesti na dva načina:

```
$ VARIABLE=value; export VARIABLE
```

ili

```
$ export VARIABLE=value
```

Promenljive okruženja mogu se prikazati pomoću komande export:

```
$ env
PWD=/home/nm
REMOTEHOST=172.16.40.158
HOSTNAME=tulip.internal.vets.edu.yu
...
HOME=/home/nm
PATH=/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:
/home/nm/bin
_=/usr/bin/env
$export
declare -x HISTSIZE="1000"
declare -x HOME="/home/nm"
declare -x HOSTNAME="tulip.internal.vets.edu.yu"
...
declare -x TERM="xterm"
declare -x USER="nm"
```

Prilagođavanje odziva (prompt) komandnog interpretera

U podrazumevanom stanju, odziv u bash shellu je znak $ za regularne korisnike, odnosno # za superusera. Svaki korisnik može promeniti svoj shell prompt pomoću lokalne promenljive PS1 (prompt string). Ove promene važe samo za tekući shell. Kada se korisnik ponovo prijavi na sistem dobiće podrazumevani prompt.

Sledeći primer ilustruje promenu odziva u niz karaktera praćenih znakom $ (znak $ nije obavezan, ali se preporučuje)

```
$ PS1="Good morning $ "
Good morning $
```

Dvostruki navodnici moraju se navesti zato što u odzivu postoji razmak i jedan specijalan karakter ($). Sledeći primer ilustruje korišćenje komande (uname -n) u odzivu. Ova komanda daje promt koji pokazuje ime host računara na kome je korisnik prijavljen. Znak backquote (‘) koristi se radi zamene niza pod navodnicima izlazom komande uname.

```
Good morning $ PS1="`uname –n` $ "
tulip.internal.vets.edu.yu $
```

Korisnik može vratiti originalni prompt sledećom komandom:

```
tulip.internal.vets.edu.yu $ PS1="$ "
$
```

Korisnik takođe može da podesi da odziv prikazuje tekući direktorijum:

```
$ PS1='$PWD $'
tmp $
```

Korisnik takođe može pomoću specijalnih karaktera formirati odziv koji prikazuje:

\u	korisničko ime,
\d	tekući datum,
\h	ime host računara,
\W	tekući direktorijum.

```
tmp $ PS1="\u@\h, \d, \W $ "
nmacek@tulip, Wed Dec 24, tmp $
```

Inicijalizacione datoteke komandnog interpretera

Inicijalizacione shell datoteke sadrže skup komandi koje se izvršavaju prilikom prijavljivanja korisnika na sistem radi podešavanja radnog okruženja. Neke komandne interpretere (bash, C shell) karakteriše postojanje datoteka koje se izvršavaju prilikom odjavljivanja korisnika sa sistema. Postoje dve vrste shell inicijalizacionih datoteka:

- globalne inicijalizacione datoteke, koje se nalaze u /etc direktorijumu (na primer /etc/profile). Ove datoteke postavljaju globalno okruženje za sve korisnike sistema i može ih modifikovati samo superuser;

- korisničke inicijalizacione datoteke, koje se nalaze u home direktorijumu korisnika (~/.bash_profile, ~/.bashrc), a koje postavljaju okruženje specifično za datog korisnika. Ove datoteke osim superusera može modifikovati i sam korisnik.

Prilikom prijavljivanja na sistem najpre se izvršavaju inicijalizacione datoteke iz /etc direktorijuma, čime se postavlja globalno okruženje, a zatim datoteke iz home direktorijuma korisnika koje prilagođavaju radno okruženje potrebama konkretnog korisnika. Na primer, korisnik kome se ne dopada odziv koji je administrator postavio u globalnoj konfiguraciji, odnosno u datoteci /etc/profile, može postaviti specifičan prompt u datoteci ~/.bash_profile.

Glavne funkcije koje se izvršavaju na osnovu datoteke /etc/profile su:

- inicijalizacija promenljivih okruženja, poput LOGNAME, PATH (sistemska putanja), TERM (definiše karakteristike terminala, odnosno ekrana i tastature radne stanice),
- prikazivanje sadržaja datoteke /etc/motd (Message Of The Day - poruka koju definiše administrator sistema),
- postavljanje inicijalne vrednosti umask promenljive, odnosno podrazumevanih dozvola za novokreirane datoteke i direktorijume,
- proveravanje pošte korisnika (mail),
- izvršavanje određenog skupa dodatnih komandi, poput cal (kojom se prikazuje kalendar za tekući mesec) ili banner "LOGNAME" (koja ispisuje ime korisnika velikim slovima formiranim od specijalnih karaktera).

Korisničke inicijalizacione datoteke

Nakon izvršavanja komandi koje su navedene u datoteci /etc/profile, sistem prilagođava radno okruženje korisniku na osnovu datoteka ~/.bash_profile i ~/.bashrc. Komande koje se izvršavaju prilikom prijavljivanja na sistem i definicije promenljivih smeštaju se u datoteku ~/.bash_profile. Alternativna imena komandi i odziv definišu se u datoteci ~/.bashrc. Kako se prilikom prijavljivanja na sistem čitaju obe datoteke korisnik njima može definisati:

- podrazumevani (default) štampač,
- podrazumevane dozvole za nove datoteke i direktorijume (umask),
- tip terminala,
- sistemsku putanju (path),
- odziv komandnog interpretera (prompt),
- alternativna imena komandi (alias),
- skup one-time komandi, odnosno komandi koje je potrebno izvršiti prilikom prijavljivanja korisnika na sistem (poput autoexec.bat u MS-DOS operativnom sistemu).

Prilikom odjavljivanja korisnika sa sistema čita se datoteka ~/.bash_logout i izvršavaju one komande koje je korisnik naveo u toj datoteci (na primer, brisanje sadržaja nekog privremenog direktorijuma, šifrovanje značajnih datoteka).

Poređenje poznatih komandnih interpretera

U svetu UNIX i Linux sistema, postoji veliki broj različitih komandnih interpretera koji pružaju različit skup mogućnosti i nude specifičan jezik za pisanje shell programa (shell script). Osnovni nedostatak prvih UNIX komandnih interpretera je nemogućnost pamćenja, ponavljanja i izmene prethodno zadatih komandi, što u slučaju greške znači da cela komandna linija mora ponovo da se otkuca. Sedamdesetih godina pojavili su se Bourne Shell, Korn Shell i C Shell, na osnovu kojih su kasnije formirane dve klase komandnih interpretera: klasa bazirana na Bourne shell-u i klasa bazirana na C shell-u. Osim komandnih interpretera koji su opisani u nastavku teksta, postoje još neki čija popularnost nije velika, kao što su Adventure shell (ash), Extensible shell (es), ERGO shell (esh) i Z shell (zsh).

Bourne shell (sh)

Stephen Bourne je razvio Bourne shell za AT&T UNIX okruženje. Bourne shell (sh) se smatra za originalni UNIX komandni interpreter sa potpunim skupom pratećih loših osobina kao što su nemogućnost pamćenja, ponavljanja i modifikacije prethodne komande linije. Moderniji komandni interpreteri koriste skup naredbi Bourne shell-a, kao i većinu njegovih dobrih osobina. Bourne shell (sh) postoji na svim UNIX/Linux sistemima, ali se svi noviji komandni interpreteri koriste kao podrazumevani, jer su osetno bolji. Bourne shell je poznat po tome što je uveo mnogo suštinskih ideja, kao što je, na primer, izlazni status izvršenih komandi, koji je praktično omogućio pisanje shell script programa.

C shell (csh)

C shell (csh) je razvijen s ciljem da pruži okruženje za pisanje skriptova i izvršavanje naredbi izvedenih iz sintakse popularnog jezika C. Kod osnovnog C shella ne postoji mogućnost modifikacije komandne linije, ali postoji mogućnost ponavljanja komandi, kao i mogućnost kreiranja aliasa. Većina Linux sistema, nudi poboljšanu varijantu C shella, koja se naziva Enhanced C shell (tcsh) koji omogućava modifikaciju komandi. Pored sličnosti sa C sintaksom, C shell ima ugrađenu aritmetiku i funkciju poređenja, dok interpreteri bazirani na Bourne shell-u u te svrhe moraju pozivati eksterne komande (expr).

Bourne-again shell (bash)

Bourne-again shell (bash) je najčešće korišćeni komandni interpreter pod Linux sistemima. Iz imena se vidi da je to poboljšana verzija Bourne shella, koja pruža mnoge dodatne mogućnosti kao što je ponavljanje i modifikovanje komandi i kompletiranje imena datoteka.

Korn shell (ksh)

Korn shell (ksh) je unapređeni Bourne shell za većinu UNIX sistema, osim za Linux gde je prihvaćen bash. Korn shell je među prvima uveo mogućnosti ponavljanja i modifikacije komandi, bazirane na eksternim editorima kao što su vi i emacs.

Osnovne komande za rad sa datotekama

Dobijanje pomoći. Lokatori komandi. Određivanje tipa datoteke. Informacije o sistemu.

U nastavku teksta opisane su osnovne komande za rad sa datotekama koje, kao i većina drugih komandi opisanih u ovom poglavlju, postoje na svim UNIX sistemima.

Dobijanje pomoći

Prilikom rada na UNIX sistemu korisnik može dobiti pomoć od sistema na nekoliko načina. Najjednostavniji način za dobijanje pomoći je navođenje opcije --help samoj komandi. Na primer:

```
$ mkdir --help
Usage: mkdir [OPTION] DIRECTORY...
Create the DIRECTORY(ies), if they do not already exist.

  -m, --mode=MODE   set permission mode (as in chmod), not
rwxrwxrwx - umask
  -p, --parents     no error if existing, make parent directories
as needed
  -v, --verbose     print a message for each created directory
      --help        display this help and exit
      --version     output version information and exit

Report bugs to <bug-fileutils@gnu.org>.
```

Većina komandi na ovaj način ispisuje na ekranu sintaksu i objašnjenja za odgovarajuće argumente i opcije, bez detaljnijeg opisa same komande. Ukoliko je objašnjenje dugačko, i ne može stati na jedan ekran, prethodnu komandu treba spregnuti u pipeline sa komandom less, čime se obezbeđuje pauza posle svake prikazane stranice pomoći (command --help | less).

Jedan od najkompletnijih izvora pomoći (ponekad i jako komplikovan i nejasan) su stranice uputstva za korišćenje komande (manual page, odnosno man page). Određena stranica uputstva prikazuje se pomoću komande man, čija je opšta sintaksa:

```
$ man command
```

U sledećem primeru prikazana je stranica uputstva za komandu lsmod (lsmod prikazuje spisak modula koji su učitani u aktivno Linux jezgro):

```
$ man groups

Reformatting groups(1), please wait...

GROUPS(1)                       FSF                          GROUPS(1)

NAME
       groups - print the groups a user is in

SYNOPSIS
       groups [OPTION]... [USERNAME]...

DESCRIPTION
       --help display this help and exit

       --version
              output version information and exit

       Same as id -Gn.  If no USERNAME, use current process.

REPORTING BUGS
       Report bugs to <sh-utils-bugs@gnu.org>.

SEE ALSO
       The  full documentation for groups is maintained as a Tex-
       info manual.  If the info and groups programs are properly
       installed at your site, the command

              info groups

       should give you access to the complete manual.

GNU sh-utils 2.0.11         July 2001                    GROUPS(1)
```

Svaka stranica uputstva sačinjena je od nekoliko elemenata, kao što su kratak opis komande, sintaksa, detaljan opis, argumenti i opcije, ime autora, neispravnosti u programu i licenca. Stranice su troff formatirane, a za navigaciju po stranici se mogu koristiti sledeći tasteri:

<Space>	sledeća stranica,
b	prethodna stranica,
<Enter> (ili <UP>)	sledeći red,
<DOWN>	sledeći ili prethodni red,
q	napuštanje programa man i povratak u shell.

Za interne shell komande postoji još jedan vid dokumentacije - infopage. Slično stranicama uputstva, info stranice se dobijaju pomoću komande info, čija je opšta sintaksa:

```
$ info infopage
```

Sledeći primer ilustruje prikazivanje info stranice za komandu groups:

```
$ info groups
```

```
File: sh-utils.info,  Node: groups invocation,  Next: users
invocation,  Prev: whoami invocation,  Up: User information

`groups': Print group names a user is in
========================================

   `groups' prints the names of the  primary and any supplementary
groups for each given USERNAME, or the current process if no names
are given.  If names  are given, the name of  each user is printed
before the list of that user's groups.  Synopsis:

     groups [USERNAME]...

   The  group lists  are equivalent  to the  output of the command
`id -Gn'.

   The  only options  are `--help'  and `--version'.  *Note Common
options::.

--zz-Info: (sh-utils.info.gz)groups invocation, 18 lines --All-----
Welcome to Info version 4.1. Type C-h for help, m for menu item.
```

Dodatno, korisnik koji nije siguran u ime komande koju želi da pokrene, može iskoristiti komandu apropos. Komanda apropos na ekranu prikazuje ime i opis svih komandi koje u opisu imaju zadati string.

```
$ apropos partition
GNU Parted [parted]  (8)  - a partition manipulation program
fdisk                (8)  - Partition table manipulator for Linux
iostat               (1)  - Report Central Processing Unit (CPU)
statistics and input/output statistics for devices and partitions
sfdisk               (8)  - Partition table manipulator for Linux
```

Napomena: detaljna dokumentacija o Linux komandama i programskim paketima može se naći na Internetu.

Lokatori komandi

Lokacija određenog programa u aktivnom UNIX stablu može se odrediti komandama which i whereis. Komanda whereis prikazuje lokaciju izvršnih datoteka, izvornog koda i prateće dokumentacije programa. Sintaksa komande whereis je:

```
$ whereis [-bms] command
```

Ukoliko se ne navede ni jedan parametar, komanda whereis prikazuje lokacije svih elemenata programa. Parametri određuju element programa čiju lokaciju treba odrediti (-b izvršne datoteke, -m uputstva, -s izvorni kôd). Primer ilustruje upotrebu komande whereis za pronalaženje lokacije programa insmod (koji se koristi za dodavanje modula u aktivno jezgro):

```
$ whereis insmod
insmod: /sbin/insmod /usr/share/man/man8/insmod.8.gz
```

Za razliku od komande whereis, komanda which prikazuje samo lokaciju izvršnih datoteka. Komanda koristi sličan algoritam kao i Bourne-again shell - traži izvršnu datoteku u direktorijumima navedenim u sistemskoj putanji i ukoliko je nađe, prikazuje putanju i ime prve pronađene komande na standardnom izlazu. Sintaksa komande which je:

```
$ which [-a] command
```

Ukoliko se navede opcija –a, komanda which na standardnom izlazu prikazuje ime i putanju svih nađenih komandi, a ne samo prve.

Napomena: većina programa koji se nalaze u direktorijumima /sbin i /usr/sbin su programi namenjeni superuseru. Ovi direktorijumi se ne nalaze u sistemskoj putanji običnih korisnika, tako da oni komandom which ne mogu locirati programe koji se u tim direktorijumima nalaze.

```
# which insmod
/sbin/insmod
# which fdisk
/sbin/fdisk
# su jsmith
$ which insmod
$ which fdisk
$ which cp
/bin/cp
$ which less
/usr/bin/less
```

Prikazivanje informacija o sistemu

Komanda unname -a na standardnom izlazu štampa informacije o sistemu, poput imena operativnog sistema, arhitekture hardvera i imena host računara. Bez parametra -a, prikazuje se samo ime operativnog sistema.

```
$ uname
Linux
$ uname -a
Linux tulip 2.4.19-686 #1 SMP Sat Apr 24 10:56:05 CEST 2004 i686
unknown
```

Dodatne informacije o sistemu, kao što su količina slobodne memorije, skup aktiviranih sistema datoteka i skup modula učitanih u kernel dostupne su svim korisnicima pomoću odgovarajućih komadni (u ovom slučaju free, mount i /sbin/lsmod).

Određivanje tipa datoteke

Za razliku od programa koji rade u grafičkom okruženju (kao što su gimp i Open Office), programi koji rade u UNIX komandnoj liniji ne prepoznaju datoteke na osnovu ekstenzija. Posmatrano sa stanovišta komandne linije može se reći da UNIX nema pojam ekstenzije datoteke. Zato na UNIX sistemima postoji poseban program file pomoću koga korisnici mogu odrediti tip datoteke. Određivanjem tipa datoteke korisnik određuje i program kojim

će izvršiti neku akciju nad datotekom (kao što je prevođenje C programa, izvršavanje shell programa ili prikazivanje slike ili video zapisa).

Tipovi datoteka

Na UNIX sistemima postoji nekoliko osnovnih tipova datoteka:

- tekstualne datoteke - ASCII (neformatiran tekst), English text (tekst sa interpunkcijskim karakterima) i izvršni shell programi. Ovaj tip datoteka se može pročitati korišćenjem programa za pregled sadržaja tekstualnih datoteka (cat, less) ili modifikovati pomoću editora teksta (kao što su vi i joe);
- izvršne (binarne) datoteke;
- datoteke u koje su smešteni podaci (na primer, Open Office Writer dokument). Ove datoteke mogu se pročitati korišćenjem programa iz koga su kreirane i programa koji je kompatibilan sa tim formatom datoteka (na primer, Open Office može da otvori dokumente kreirane Microsoft Office paketom).

Prilikom određivanja tipa datoteke komanda file izvršava sledeće testove:

- test specijalnih datoteka (filesystem test),
- test magičnih brojeva (magic number test),
- jezički test (language test).

Testom specijalnih datoteka se na osnovu sistemskog poziva stat() određuje da li je datoteka regularna ili specijalna (odnosno node, simbolički link, imenovani pipe). Ukoliko datoteka prođe ovaj test na ekranu se prikazuje tip datoteke, a dalje izvršenje komande se obustavlja.

Testom magičnih brojeva određuje se programski paket kojim je datoteka kreirana. Svaki program prilikom kreiranja datoteka upisuje neke kontrolne informacije (overhead) koje ovaj test posmatra kao identifikator tipa datoteke, odnosno magični broj. Bilo koja datoteka sa nepromenjenim identifikatorom, koji se nalazi na fiksnom ofsetu od početka datoteke, može biti opisana na ovaj način. Magični brojevi se smeštaju u datoteke /usr/share/magic.mgc i /usr/share/magic i prilikom izvršavanja ovog testa redom traže na fiksnom ofsetu u datoteci. Ukoliko se određeni magični broj poklopi sa magičnim brojem datoteke, program file prekida dalje izvršenje i na ekranu ispisuje informaciju o programu kojim je datoteka kreirana.

Ukoliko datoteka ne prođe ni test magičnih brojeva, program file će pokušati da odredi da li je datoteka tekstualna i u tom smislu će najpre izvršiti test seta karaktera (character set test). U datoteci postoje tri tipa karaktera:

- normalni karakteri (karakteri koji mogu biti prikazani na ekranu),
- kontrolni karakteri, kao što su space i tab (karakteri koji ne mogu biti prikazani na ekranu, ali se pojavljuju u običnim tekstualnim datotekama),
- binarni karakteri (karakteri koji se ne mogu prikazati direktno na ekranu, i ne pojavljuju se u običnim tekstualnim datotekama).

Karakter set (character set) je jednostavna deklaracija normalnih, kontrolnih i binarnih karaktera. Na osnovu karaktera koji se u datoteci pojavljuju komanda file određuje karakter set kome datoteka pripada. Ukoliko datoteka prođe ovaj test komanda file na ekranu prikazuje ime karakter seta datoteke (ASCII, ISO-8859-x, UTF-8, UTF-16, EBCDIC). Nakon toga se izvršava jezički test (language test) kojim će komanda file pokušati da odredi programski jezik u kome je datoteka napisana - u datoteci se traže nizovi karaktera koji su karakteristični za određene programske jezike. Na primer, ukoliko se u datoteci pojavi reč keyword.br, to znači da se ispitivanje vrši nad troff(1) ulaznom datotekom, dok reč struct nagoveštava da se radi o C programu. U slučaju da program file jezičkim testom ne može da odredi programski jezik, datoteka se smatra tekstualnom.

Upotreba komande file

Sintaksa komande file je:

```
$ file [-z] [-f namefile] [-m magicfiles] filename
```

gde je argument filename ime datoteke čiji se tip određuje. Važnije opcije su:

-f namefile	imena datoteka koje treba ispitati čitaju se iz datoteke namefile (jedna datoteka po liniji);
-m magicfiles	specificira alternativnu listu datoteka koje sadrže magične brojeve. Usvojeno je da se magični brojevi čitaju iz datoteke /usr/share/magic, ali se ovo podešavanje može izmeniti pomoću promenljive MAGIC;
-z	pokušava da pogleda unutar kompresovanih fajlova.

Sledeći primeri ilustruju upotrebu komande file:

```
$ file /etc/hosts
/etc/hosts: ASCII English text
$ file /bin/cp
/bin/cp: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), stripped
$ file /dev/hda
/dev/hda: block special (3/0)
$ file /dev/lp0
/dev/lp0: character special (6/0)
$ file /sbin/rmt
/sbin/rmt: symbolic link to /usr/sbin/rmt
$ file /etc
/etc: directory
```

Komanda strings

Komanda strings može se upotrebiti za prikazivanje vidljivih karaktera u izvršnoj ili binarnoj datoteci. Razni aplikativni programi upisuju ime programa u prvih par linija datoteke, tako da se komanda strings može upotrebiti radi određivanja programa kojim je datoteka napravljena, a samim tim i (alternativnog) programa koji je pogodan za otvaranje

datoteke. Komanda strings je ovde upotrebljena samo radi demonstracije prikaza štampajućih karaktera izvršne datoteke.

```
# strings /bin/cp
/lib/ld-linux.so.2
libc.so.6
strcpy
...
Try `%s --help' for more information.
Usage: %s [OPTION]... SOURCE DEST
  or:  %s [OPTION]... SOURCE... DIRECTORY
  or:  %s [OPTION]... --target-directory=DIRECTORY SOURCE...
Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.
  -a, --archive                same as -dpR
      --backup[=CONTROL]       make a backup of each existing
destination file
...
Written by %s.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
memory exhausted
```

Kopiranje, pomeranje i brisanje datoteka

Komande cp, mv i rm. Relacioni odnosi originala i kopije. Potrebni i dovoljni uslovi za kopiranje, pomeranje i brisanje datoteka

U nastavku teksta opisane su komande cp, mv i rm koje respektivno služe za kopiranje, pomeranje i brisanje datoteka i direktorijuma. Opisani su značajni flegovi, relacioni odnosi između originalne datoteke i kopije u slučaju kopiranja i pomeranja (vlasnički odnosi, pristupna prava, vremena), kao i potrebni i dovoljni uslovi za izvršenje ovih komandi.

Kopiranje datoteka i direktorijuma

Komanda cp služi za kopiranje datoteka i direktorijuma. U opštem slučaju sintaksa komande cp je:

```
$ cp SOURCE DESTINATION
```

što znači da komanda cp kopira izvorišni objekat sistema datoteka (SOURCE) u odgovarajuće odredište (DESTINATION).

Napomena: parametri SOURCE i DESTINATION moraju se navesti. Ukoliko korisnik želi da iskopira datoteku u tekući direktorijum, kao odredišni argument može navesti tačku.

U zavisnosti od parametara SOURCE i DESTINATION postoje tri osnovna oblika korišćenja komande cp:

- kopiranje jedne datoteke,
- kopiranje grupe datoteka iz istog direktorijuma,
- rekurzivno kopiranje direktorijumskog stabla.

Kopiranje jedne datoteke

Sintaksa komande cp u režimu kopiranja jedne datoteke je:

```
$ cp source_file destination
```

Argument source_file je ime izvorišne datoteke, dok argument destination može biti odredišni direktorijum ili ime odredišne datoteke. Argumenti sourcefile i destination (ukoliko je destination ime datoteke) opciono mogu sadržati ili apsolutne ili relativne putanje datoteka.

U nastavku teksta dati su primeri korišćenja komande cp u ovom režimu rada:

```
$ cp a.a b.b
$ cp /home/a.a /tmp/b.b
$ cp a.a /tmp/a.a
```

Osobine ovog režima rada komande cp su sledeće:

- kopira se samo jedna datoteka u drugu (moguće je prepisati postojeću datoteku),
- original i kopija mogu biti na istom ili različitim direktorijumima,
- original i kopija mogu imati ista ili različita imena,
- ne može se kreirati istoimena kopija datoteke na istom direktorijumu (datoteka se ne može kopirati u samu sebe).

Kopiranje grupe datoteka iz istog direktorijuma

Sintaksa komande cp u režimu kopiranja grupe datoteka iz istog direktorijuma je:

```
$ cp source_files dest_dir
```

Grupa izvorišnih datoteka navodi se kao argument source_files koji se formira pomoću džoker karaktera (*, ?, [a-z], [A-Z], [0-9]). Džokeri mogu da zamene jedan ili više karaktera ili opseg (pogledati funkcije komandnog interpretera - zamena imena datoteka). Odredišni objekat u ovom režimu rada mora biti direktorijum (apsolutna ili relativna putanja) i specificira se argumentom dest_dir.

U nastavku teksta dati su primeri korišćenja komande cp u ovom režimu rada:

```
$ cp a* /tmp
$ cp a?.old /tmp
$ cp a3[0-9].old  /usr/tmp
$ cp /etc/[a-d][1-5]* .
```

Osobine ovog režima rada komande cp su sledeće:

- kopira se grupa datoteka iz iste grane;
- originali i kopije moraju imati ista imena. Nije moguće izvesti kopiranje grupe datoteka sa promenom imena kopija (na primer cp a* /tmp/b*);
- originali i kopije moraju biti na različitim direktorijumima.

Rekurzivno kopiranje direktorijumskog stabla

Sintaksa komande cp u režimu rekurzivnog kopiranja direktorijumskog stabla je:

```
$ cp –r(R) source_dir dest_dir
$ cp –r source_files dest_dir
```

U nastavku teksta dati su primeri korišćenja komande cp u ovom režimu rada:

```
$ cp –r /etc /tmp/oldconfig
$ cp –r /etc/* /tmp/oldconfig
$ cp –r a* /tmp/mybackup
```

Primeri redom ilustruju:

- kopiranje direktorijuma /etc sa svim poddirektorijumima i datotekama u direktorijum /tmp/oldconfig/etc (datoteka /etc/passwd kopira se u /tmp/oldconfig/etc/passwd),
- kopiranje kompletnog sadržaja direktorijuma /etc u direktorijum /tmp/oldconfig (datoteka /etc/passwd kopira se u /tmp/oldconfig/passwd),
- kopiranje datoteka čije ime počinje sa a iz tekućeg direktorijuma i svih poddirektorijuma u direktorijum /tmp/mybackup.

Osobine ovog režima rada komande cp su sledeće:

- kopira se celo stablo ili grupa datoteka iz jednog direktorijuma i svih poddirektorijuma u odredišni direktorijum;
- originali i kopije moraju imati ista imena. Nije moguće izvesti kopiranje grupe datoteka ili direktorijuma sa promenom imena kopija (na primer cp -r /etc/a* /tmp/b*);
- originali i kopije moraju biti na različitim direktorijumima.

Relacioni odnosi originala i kopije

Prilikom kopiranja operativni sistem kreira datoteku istog sadržaja. Ukoliko se datoteka kopira u isti direktorijum, ime kopije mora biti različito od imena originala. Ukoliko se datoteka kopira u drugi direktorijum, ime kopije može biti isto ili različito od imena originala.

Što se tiče vlasničkih odnosa i pristupnih prava, važe sledeća pravila:

- vlasnik kopije je korisnik koji je pokrenuo komandu cp,
- datoteka se dodeljuje primarnoj grupi korisnika koji je pokrenuo komandu cp,

- pristupna prava kopije su najčešće sužena u odnosu na pristupna prava originala, a dobijaju se logičkim množenjem bitova pristupnih prava originala i vrednosti promenljive umask. Na primer: ako su pristupna prava originalne datoteke 666, a vrednost promenljive umask 002, pristupna prava kopije biće 664.

Što se tiče vremena, sva tri vremena kopije (vreme kreiranja, poslednjeg pristupa i poslednje modifikacije) jednaka su vremenu pokretanja komande cp. Kako se originalu mora pristupiti da bi se napravila kopija, vreme poslednjeg pristupa originalne datoteke se takođe menja i jednako je vremenu pokretanja komande cp.

Broj hard linkova kopije i originala je jednak i ne menja se, osim ukoliko se umesto kopiranja vrši kreiranje hard linka. U tom slučaju se broj linkova originala i kopije (hard linka) povećava za jedan.

Potrebni i dovoljni uslovi za kopiranje datoteke

Da bi korisnik mogao da iskopira datoteku <file> iz direktorijuma <dir1> u direktorijum <dir2> potrebno je da ima sledeća dva prava:

- pravo r nad datotekom <file> (čime se omogućava čitanje sadržaja originalne datoteke).
- pravo w nad direktorijumom <dir2> (čime se omogućava izmena sadržaja odredišnog direktorijuma, odnosno kreiranje nove datoteke).

Ovaj skup prava je minimalan - neke varijante UNIX sistema zahtevaju dodatna prava. Dovoljni uslovi za kopiranje fajla na svim UNIX sistemima uključuju potrebne uslove i pravo x nad direktorijumima <dir1> i <dir2>.

Opcije komande cp

Komanda cp ima veliki broj opcija, od kojih su najznačajnije opisane u nastavku teksta:

-i interaktivni režim kopiranja (u slučaju da datoteka istog imena već postoji pitaće korisnika da li želi da je prepiše kopijom). Ovaj režim je podrazumevan za superusera;

-f forsirani režim kopiranja (prilikom kopiranja ne postoji interakcija između programa cp i korisnika). Ovaj režim je podrazumevan za regularne korisnike;

-l umesto kopije kreira se hard link;

-s umesto kopije kreira se simbolički link;

-r rekurzivno kopiranje direktorijumskog stabla (koristi se isključivo za kopiranje regularnih datoteka);

-R rekurzivno kopiranje direktorijumskog stabla (koristi se za kopiranje regularnih i specijalnih datoteka kao što je imenovani pipe);

-d prilikom kopiranja simboličkih linkova vrši se dereferenciranje, odnosno kopira se originalna datoteka, a ne link. Ukoliko se navede opcija -d, dereferenciranje se ne vrši - kopira se link, a ne originalna datoteka.;

-p ukoliko se datoteka kopira bez preserve opcije (-p), prava pristupa i vreme poslednje modifikacije datoteke se menjaju. Ukoliko regularan korisnik kopira datoteku sa opcijom -p, prava pristupa i vreme poslednje modifikacije datoteke verno se prenose. Ukoliko superuser root kopira datoteku sa opcijom -p, svi parametri se verno prenose (vlasnik, grupa, prava pristupa i vreme poslednje modifikacije);

-u datoteka se kopira samo u slučaju da je novijeg datuma od postojeće kopije ili ukoliko kopija ne postoji;

-b kreira se rezervna kopija (backup) postojeće kopije pod drugim imenom (ukoliko se drugačije ne naznači, dodaje se sufix ~).

Sufiks rezervne kopije datoteke može se promeniti pomoću opcije --suffix=SUFFIX ili pomoću promenljive okruženja SIMPLE_BACKUP_SUFFIX. Dodatno, komanda cp podržava kontrolu verzije datoteke (version control), koja se može uključiti ili pomoću opcije --backup=numbered ili dodelom vrednosti numbered promenljivoj okruženja VERSION_CONTROL.

Primeri korišćenja komande cp

Sledećih nekoliko primera ilustruju relacione odnose između originala i kopije, potrebne i dovoljne uslove za kopiranje datoteka i uticaj flegova na kopiranje.

Primer 1. Primer ilustruje uticaj promenljive umask na pristupna prava kopije. U primeru je korišćena vrednost promenljive umask 022 koja ukida pravo čitanja kategorijama group i others, kao i restriktivnija vrednost 027, koja ukida pravo čitanja kategoriji group i sva prava kategoriji others. Najpre je potrebno na direktorijumu /tmp kreirati sledeće datoteke:

```
-rwxrwxrwx   1   root   root   31   11:00   /tmp/f1
-rw-rw-rw-   1   root   root   31   11:00   /tmp/f2
-rw-r-----   1   root   root   31   11:00   /tmp/f3
```

Korisnik dalje radi sledeće:

```
$ whoami
bora
$ cd ~
$ pwd
/home/bora
$ umask 022
$ cp /tmp/f1 ./f1
$ cp /tmp/f2 ./f2
$ cp /tmp/f3 ./f3
$ ls -l /tmp/f1 f1
-rwxrwxrwx   1   root   root   31   11:00   /tmp/f1
-rwxr-xr-x   1   bora   bora   31   11:05   f1
$ ls -l /tmp/f2 f2
-rw-rw-rw-   1   root   root   31   11:00   /tmp/f2
-rw-r--r--   1   bora   bora   31   11:05   f2
```

```
$ ls -l /tmp/f3 f3
-rw-r-----   1   root   root   31   11:00   /tmp/f3
-rw-r-----   1   bora   bora   31   11:05   f3
$ rm f1 f2 f3
```

Slične aktivnosti se obavljaju ukoliko se analizira uticaj vrednosti 027 promenljive umask:

```
$ umask 027
$ cp /tmp/f1 ./f1
$ cp /tmp/f2 ./f2
$ cp /tmp/f3 ./f3
$ ls -l /tmp/f1 f1
-rwxrwxrwx   1   root   root   31   11:00   /tmp/f1
-rwxr-x---   1   bora   bora   31   11:10   f1
$ ls -l /tmp/f2 f2
-rw-rw-rw-   1   root   root   31   11:00   /tmp/f2
-rw-r-----   1   bora   bora   31   11:10   f2
$ ls -l /tmp/f3 f3
-rw-r-----   1   root   root   31   11:00   /tmp/f3
-rw-r-----   1   bora   bora   31   11:10   f3
$ rm f1 f2 f3
```

Primer 2. Sledeći primer ilustruje prava pristupa koja su neophodna da bi se izvršilo kopiranje datoteke. Korisnik najpre radi sledeće:

```
$ cd
$ mkdir dst
$ mkdir /tmp/src
$ >/tmp/src/f1
$ ls -ld dst /tmp/src
drwxrwxr-x   2   bora   bora   4096   12:20   dst
drwxrwxr-x   2   bora   bora   4096   12:20   /tmp/src
$ ls -l /tmp/src/f1
-rw-rw-r--   1   bora   bora      0   12:20   /tmp/src/f1
$ whoami
bora
```

Pošto su adekvatna prava pristupa dodeljena korisniku, kopiranje datoteke se može obaviti:

```
$ cp /tmp/src/f1 ~/dst/f1
```

Zatim se analizira potreban skup prava na konkretnom UNIX sistemu. Korisniku se redom oduzimaju prava iz skupa dovoljnih uslova. Nakon svakog ukinutog prava korisnik može isprobati da li je to pravo na konkretnom UNIX sistemu potrebno da bi se izvršilo kopiranje. U nastavku teksta prikazana je reakcija Debian Linux 3.0r2 sistema na ukidanje ovih prava:

- ukinuto pravo r nad datotekom f1

```
$ chmod -r /tmp/src/f1
$ cp /tmp/src/f1 ~/dst/f1
cp: cannot open `/tmp/src/f1' for reading: Permission denied
$ chmod +r /tmp/src/f1
```

- ukinuto pravo w nad direktorijumom ~/dst

```
$ chmod -w dst
$ cp /tmp/src/f1 ~/dst/f1
cp: cannot create regular file `/home/bora/dst/f1': Permission
denied
$ chmod +w dst
```

- ukinuto pravo x nad direktorijumom /tmp/src

```
$ chmod -x /tmp/src
$ cp /tmp/src/f1 ~/dst/f1
cp: cannot stat `/tmp/src/f1': Permission denied
$ chmod +x /tmp/src
```

- ukinuto pravo x nad direktorijumom ~/dst

```
$ chmod -x ~/dst
$ cp /tmp/src/f1 ~/dst/f1
cp: accessing `/home/bora/dst/f1': Permission denied
$ chmod +x ~/dst
```

Primer 3. Sledeći primer ilustruje uticaj opcije -p (preserve) komande cp. Najpre je potrebno da na direktorijumu /tmp superuser kreira datoteku f1 sa pravima pristupa 666:

```
# cd /tmp
# >f1
# chmod 666 f1
# ls -l f1
-rw-rw-rw-   1   root   root   0   12:30   f1
```

Nakon toga regularan korisnik kopira datoteku komandom cp, sa i bez -p opcije.

```
$ whoami
bora
$ cd
$ umask
022
$ cp /tmp/f1 f1
$ cp -p /tmp/f1 f1-u
$ ls -l f1 f1-u
-rw-r--r--   1   root   root   0   12:35   f1
-rw-rw-rw-   1   root   root   0   12:30   f1-u
$ rm f1 f1-u
```

Zaključak: ukoliko se datoteka kopira bez opcije -p (preserve), prava pristupa i vreme poslednje modifikacije datoteke se menjaju. Ukoliko se datoteka kopira sa opcijom -p, a kopiranje izvršava regularan korisnik, prava pristupa i vreme poslednje modifikacije datoteke se prenose (u ovom slučaju umask nema uticaja na kopiranje).

Napomena: kada superuser root kopira datoteke sa -p opcijom, svi parametri se verno prenose (vlasnik, grupa, prava pristupa i vreme poslednje modifikacije).

Primer 4. Sledeći primer ilustruje uticaj opcije -u (update) komande cp. Za potrebe ovog primera može se koristiti bilo koja veća datoteka radi demonstracije vremena potrebnog za izvršavanje komande cp (na primer datoteka /etc/termcap).

Potrebno je izvršiti sledeće dve komande:

```
$ cd
$ time cp /etc/termcap my_termcap
real    0m0.180s
user    0m0.010s
sys     0m0.030s
$ time cp -u /etc/termcap my_termcap
real    0m0.009s
user    0m0.010s
sys     0m0.000s
$ rm my_termcap
```

Zaključak: nakon zadavanja komande cp sa opcijom -u, nije izvršeno kopiranje jer odredišna datoteka postoji, a izvorišna datoteka nije novija od nje.

Primer 5. Primer ilustruje interaktivni i forsirani režim kopiranja. U interaktivnom režimu sistem postavlja pitanje korisniku da li želi da prepiše datoteku (ukoliko odredišna datoteka postoji). U forsiranom režimu, sistem prepisuje postojeću datoteku bez prethodnog upozorenja.

```
$ cd
$ >/tmp/f1
$ cp /tmp/f1 f1
$ cp -i /tmp/f1 f1
cp: overwrite `f1'? y
$ cp -f /tmp/f1 f1
```

Primer 6. Primer ilustruje uticaj opcije -b (backup). Najpre se na direktorijumu /tmp kreira datoteka f1 sa nekim sadržajem:

```
$ cp /etc/passwd /tmp/f1
```

Korisnik zatim izvršava sledeće komande:

```
$ cp /tmp/f1 f1
$ ls -l f1
-rw-r--r--   1   root   root   512   12:45   f1
```

Nakon toga se menja sadržaj originalne datoteke (pomoću vi editora):

```
$ vi /tmp/f1
$ ls -l /tmp/f1
-rw-r--r--   1   root   root   850   12:50   /tmp/f1
```

Datoteka se zatim ponovo kopira sa direktorijuma /tmp na home direktorijum, ali sa opcijom -b (backup):

```
$ cp -b /tmp/f1 f1
$ ls -l f1*
-rw-r--r--   1   root   root   850   12:52   f1
```

```
-rw-r--r--   1   root   root   512   12:45   f1~
$ rm f1 f1~
```

Zaključak: datoteka f1~ je rezervna kopija. Ukoliko se drugačije ne naznači, kao sufiks se dodaje znak (~). Korisnici pomoću opcije -S mogu odabrati sufiks rezervne kopije.

```
$ cp /tmp/f1 f1
$ cp -b -S .old /tmp/f1 f1
$ ls -l f1*
-rw-r--r--   1   root   root   850   12:55   f1
-rw-r--r--   1   root   root   850   12:52   f1.old
```

Primer 7. Primer ilustruje rekurzivno kopiranje direktorijumskog stabla. Najpre se na direktorijumu /tmp kreiraju direktorijumi /tmp/dir1 i /tmp/dir1/dir11 i datoteke /tmp/dir1/f1 i /tmp/dir1/dir11/f11:

```
$ cd /tmp
$ mkdir dir1
$ mkdir dir1/dir11
$ >/dir1/f1
$ >/dir1/dir11/f11
```

Na ovaj način je na direktorijumu tmp kreirano podstablo sa dve grane i po jednom datotekom na svakoj grani. Korisnik dalje izvršava sledeće komande:

- kopira celo stablo /tmp/dir1 u home direktorijum

```
$ cd
$ cp -R /tmp/dir1 .
$ ls -R dir1
.:
dir1
./dir1:
dir11  f1
./dir1/dir11:
f11
```

- kopira celo stablo /tmp/dir1 u direktorijum ~/dir2

```
$ cd
$ mkdir dir2
$ cp -R /tmp/dir1 dir2
$ ls -R dir2
dir2:
dir1
dir2/dir1:
dir11  f1
dir2/dir1/dir11:
f11
```

- kopira sadržaj direktorijuma /tmp/dir1 u direktorijum ~/dir3

```
$ cd
$ mkdir dir3
$ cp -R /tmp/dir1/* dir3
```

```
$ ls -R dir3
dir3:
dir11  f1
dir3/dir11:
f11
$ rm -fr dir1 dir2 dir3
```

Pomeranje datoteka i direktorijuma

Komanda mv služi za pomeranje i promenu imena datoteka i direktorijuma. U opštem slučaju, sintaksa komande mv je:

```
$ mv SOURCE DESTINATION
```

što znači da komanda mv pomera izvorišni objekat sistema datoteka (SOURCE) u odgovarajuće odredište (DESTINATION). Ukoliko je izvorišni objekat datoteka komandom mv joj se može promeniti ime.

Napomena: parametri SOURCE i DESTINATION moraju se navesti. Ukoliko korisnik želi da pomeri datoteku u tekući direktorijum, kao odredišni argument može navesti tačku.

U zavisnosti od parametara SOURCE i DESTINATION postoje dva osnovna oblika korišćenja komande mv:

- promena imena i/ili pomeranje jedne datoteke,
- pomeranje grupe datoteka iz istog direktorijuma.

Promena imena i/ili pomeranje jedne datoteke

Sintaksa komande mv u režimu kopiranja jedne datoteke je:

```
$ mv source_file destination
```

Argument source_file je ime izvorišne datoteke, dok argument destination može biti odredišni direktorijum i/ili novo ime datoteke.

U nastavku teksta dati su primeri korišćenja komande mv u ovom režimu rada koji redom ilustruju:

- promenu imena datoteke,
- pomeranje datoteke iz jednog direktorijuma u drugi,
- promenu imena i pomeranje datoteke iz jednog direktorijuma u drugi.

```
$ mv a.a b.b
$ mv /home/a.a /tmp/a.a
$ mv /home/a.a /tmp/b.b
```

Pomeranje grupe datoteka iz istog direktorijuma

Sintaksa komande mv u režimu pomeranja grupe datoteka iz istog direktorijuma je:

```
$ mv source_files dest_dir
```

Grupa izvorišnih datoteka navodi se kao argument source_files koji se formira pomoću džoker karaktera (*, ?, [a-z], [A-Z], [0-9]). Odredišni objekat u ovom režimu rada mora biti direktorijum (apsolutna ili relativna putanja) i specificira se argumentom dest_dir.

U nastavku teksta dati su primeri korišćenja komande cp u ovom režimu rada:

```
$ mv a* /tmp
$ mv a?[0-9].old  /usr/tmp
$ mv /etc/[a-d]* ./backup
```

U ovom režimu rada komanda mv pomera grupu datoteka iz iste grane isključivo u drugi direktorijum. Pri tom originali i kopije moraju imati ista imena. Nije moguće istovremeno izvesti pomeranje grupe datoteka sa promenom imena (na primer mv a* /tmp/b*).

Potrebni i dovoljni uslovi za promenu imena datoteke

Da bi korisnik mogao da promeni ime datoteke <file> koja se nalazi u direktorijumu <dir1>, potrebno je da ima pravo w nad direktorijumom <dir1> (čime se omogućava izmena sadržaja izvorišnog direktorijuma, odnosno promena imena datoteke). Dovoljni uslovi za promenu imena datoteka na svim UNIX sistemima osim ovog prava uključuju i pravo x nad direktorijumom <dir1>.

Potrebni i dovoljni uslovi za pomeranje datoteke

Da bi korisnik mogao da pomeri datoteku <file> iz direktorijuma <dir1> u direktorijum <dir2>, potrebno je da ima sledeća prava:

- pravo r nad datotekom <file> (čime se omogućava čitanje sadržaja originalne datoteke),
- pravo w nad direktorijumom <dir1> (čime se omogućava izmena sadržaja izvorišnog direktorijuma, odnosno brisanje datoteke),
- pravo w nad direktorijumom <dir2> (čime se omogućava izmena sadržaja odredišnog direktorijuma, odnosno kreiranje nove datoteke).

Dovoljni uslovi za pomeranje datoteka na svim UNIX sistemima uključuju potrebne uslove i pravo x nad direktorijumima <dir1> i <dir2>.

Opcije komande mv

Najznačajnije opcije komande mv opisane su u nastavku teksta:

-i interaktivni režim pomeranja (u slučaju da datoteka istog imena već postoji pitaće korisnika da li želi da je prepiše novom datotekom). Ovaj režim je podrazumevan za superusera:

-f forsirani režim pomeranja (prilikom pomeranja ne postoji interakcija između programa mv i korisnika). Ovaj režim je podrazumevan za regularne korisnike;

-u datoteka se pomera samo u slučaju da je novijeg datuma od postojeće datoteke ili ukoliko datoteka ne postoji;

-b kreira se rezervna kopija (backup) postojeće kopije pod drugim imenom;

Brisanje datoteka

Komanda rm služi za brisanje datoteka i direktorijuma. U opštem slučaju, sintaksa komande rm je:

```
$ rm OBJECT
```

što znači da komanda rm briše objekat sistema datoteka (OBJECT). Parametar OBJECT se mora navesti, a u zavisnosti od njega postoje tri osnovna oblika korišćenja komande rm:

- brisanje jedne datoteke,
- brisanje grupe datoteka iz istog direktorijuma,
- rekurzivno brisanje direktorijumskog stabla.

Brisanje jedne datoteke

Sintaksa komande rm u režimu brisanja jedne datoteke je:

```
$ cp filename
```

gde je argument filename ime datoteke koja će biti obrisana, a opciono može sadržati i apsolutnu ili relativnu putanju datoteke.

U nastavku teksta dati su primeri korišćenja komande rm u ovom režimu rada:

```
$ rm a.a
$ rm /etc/passwd.old
```

Brisanje grupe datoteka iz istog direktorijuma

Sintaksa komande rm u režimu brisanja grupe datoteka iz istog direktorijuma je:

```
$ rm group_of_files
```

Grupa izvorišnih datoteka navodi se kao argument group_of_files koji se formira pomoću džoker karaktera (*, ?, [a-z], [A-Z], [0-9]).

U nastavku teksta dati su primeri korišćenja komande rm u ovom režimu rada:

```
$ rm a*
$ rm /etc/*.conf.old
```

Napomena: u ovom režimu rada komandom rm mogu se obrisati isključivo datoteke koje se nalaze u istom direktorijumu.

Rekurzivno brisanje direktorijumskog stabla

Sintaksa komande rm u režimu rekurzivnog brisanja direktorijumskog stabla je:

```
$ rm –R directory
```

U nastavku teksta dat je primer korišćenja komande rm u ovom režimu rada:

```
$ rm –r /etc/backup
```

Potrebni i dovoljni uslovi za brisanje datoteke

Da bi korisnik mogao da obriše datoteku <file> iz direktorijuma <dir1>, potrebno je da ima pravo w nad direktorijumom <dir1> (čime se omogućava izmena sadržaja direktorijuma, odnosno brisanje datoteke). Dovoljni uslovi za brisanje datoteka na svim UNIX sistemima osim ovog prava uključuju i pravo x nad direktorijumom <dir1>.

Opcije komande rm

Najznačajnije opcije komande mv opisane se u nastavku teksta:

-i interaktivni režim brisanja (sistem će pitati korisnika da li želi da obriše datoteku). Ovaj režim je podrazumevan za superusera;

-f forsirani režim brisanja (prilikom brisanja ne postoji interakcija između programa mv i korisnika). Ovaj režim je podrazumevan za regularne korisnike;

-R rekurzivno brisanje direktorijumskog stabla.

Linkovi

Pojam linka. Osobine hard i simboličkih linkova. Kreiranje linkova komandom ln.Upotreba opcije no-dereference komande cp.

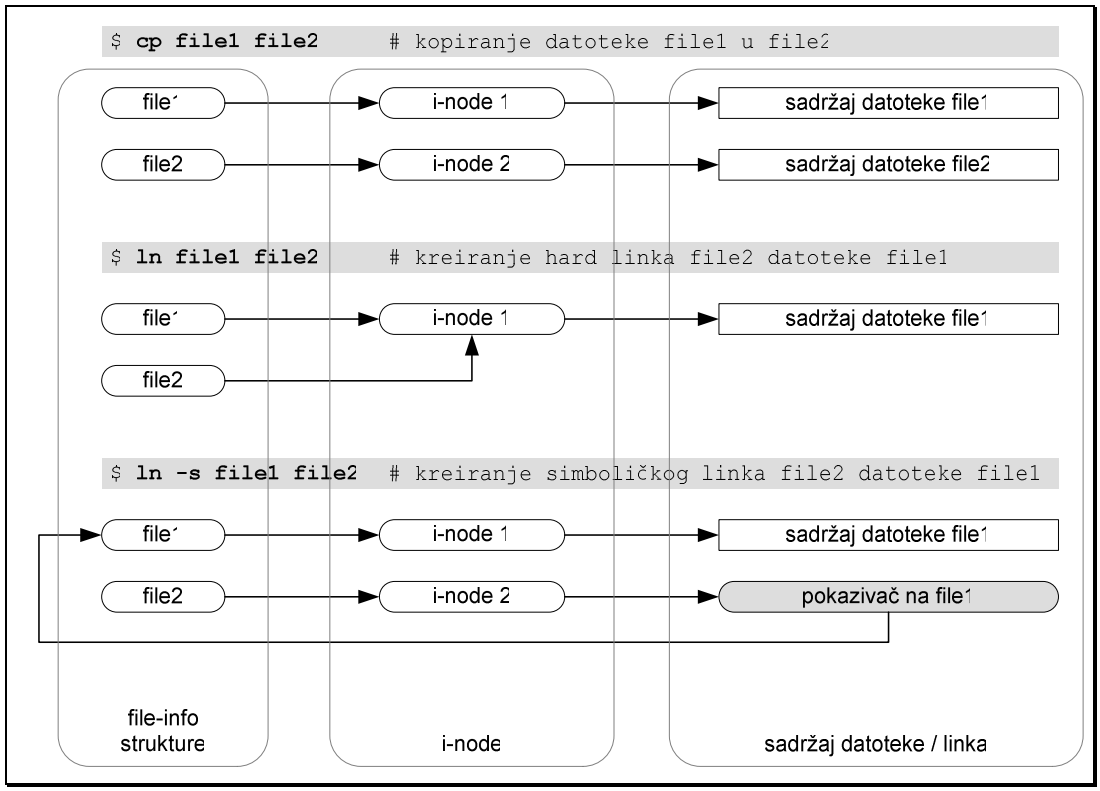
Datoteke se pod UNIX sistemom mogu povezati (linkovati). Svaka referenca na linkovanu datoteku (link) uvek se odnosi na originalnu datoteku (osim ukoliko se link navodi kao argument komandi rm). Razlozi za korišćenje linkova su:

- lakše pretraživanje i korišćenje datoteka (svakom korisniku je lakše da otkuca less ph nego less /tmp/jsmith/adresses/business/phonebook.2004, pod uslovom da u tekućem direktorijumu postoji link na odgovarajuću datoteku),
- ušteda prostora na disku (većina Linux distribucija koristi linkove na određene datoteke i direktorijume radi kompatibilnosti sa starijim varijantama UNIX sistema).

Na UNIX sistemima se mogu kreirati dve vrste linkova na datoteke:

- hard link, i
- simbolički link (symbolic link).

Slika 5.3 ilustruje hard i simboličke linkove koji su opisani u daljem tekstu.



Slika 5.3 *Hard i simbolički linkovi*

Hard linkovi

Kada korisnik pozove datoteku po imenu (na primer: cat tekst1), UNIX prevodi simboličko ime datoteke koje je naveo korisnik u interno ime, koje koristi operativni sistem. Zbog posebne interne reprezentacije, korisnici mogu datotekama dodeliti veći broj imena. Hard link je jedno od tih imena, odnosno alternativno ime datoteke. Na primer, korisnik koji u svom home direktorijumu ima datoteku file1 može kreirati hard link file2, odnosno referencu na tu datoteku

```
$ ln file1 file2
$ ls file*
file1 file2
```

Ime datoteke file2 upućuje na istu datoteku kao i file1. Ukoliko korisnik promeni sadržaj datoteke file1, istovremeno se menja i sadržaj "datoteke" file2, jer file2 nije kopija datoteke file1, već drugo ime za istu datoteku. Jednostavno rečeno, file1 i file2 su ista datoteka sa različitim imenom. Ukoliko korisnik obriše datoteku file1, file2 se ne briše.

UNIX će ukloniti datoteku tek kada se obrišu svi hard linkovi. Ukoliko datoteka ima nekoliko hard linkova, komanda rm će ukloniti link, a ne samu datoteku. Datoteka će biti obrisana tek kada se uklone svi hard linkovi koji upućuju na nju.

Hard linkovi se koriste ukoliko grupa ljudi koja, na primer, radi na nekom projektu želi da deli datoteke. Svi korisnici u svom home direktorijumu mogu imati link na datoteku koja se nalazi u nekom javnom direktorijumu. Ukoliko neki korisnik putem linka promeni sadržaj datoteke, promena se automatski reflektuje na sve linkove.

Kreiranje hard linkova

Hard linkovi se mogu kreirati na dva načina: pomoću komande ln i pomoću komande cp. Sintaksa ovih komandi je:

```
$ ln original linkname
$ cp -l original linkname
```

Argument original je ime originalne datoteke, a argument linkname ime hardlinka.

Osobine hard linkova

Hard linkovi su alternativna imena datoteka, i kao takve karakterišu ih sledeće osobine:

- link i original imaju isti i-node, tako da se moraju nalaziti na fizički istom sistemu datoteka (hard link se ne sme nalaziti na drugoj particiji ili na drugom disku). Ne mogu se linkovati datoteke sa mrežnog sistema datoteka (NFS);
- ne može se linkovati direktorijum niti nepostojeća datoteka;
- vlasnik, grupa i prava pristupa su isti za link i za original;
- slobodan prostor na disku neznatno se umanjuje (jedna dir-info struktura više za alternativno ime datoteke);
- broj linkova originalne datoteke uvećava se za jedan nakon linkovanja;
- datoteka sa hard linkovima se ne može obrisati sa diska sve dok se ne uklone svi hard linkovi koji upućuju na tu datoteku.

Primer kreiranja i upotrebe hard linkova

Za potrebe ovog primera upotrebljena je velika datoteka /etc/termcap. Najpre se određuje broj linkova i atributi originalne datoteke.

```
$ ls -l /etc/termcap
-rw-r--r--   1   root   root   729360   Mar 14 2001   /etc/termcap
```

Nakon toga se kreira jedan hard link datoteke na tmp direktorijumu i određuje broj linkova originalne datoteke i atributi linka.

```
$ cp -l /etc/termcap /tmp/tcap1
$ ls -l /etc/termcap   # broj linkova originala: 2
-rw-r--r--   2   root   root   729360   Mar 14 2001   /etc/termcap
```

```
$ ls –l /tmp/tcap1
-rw-r--r--   2   root   root   729360   Mar 14 2001   /tmp/tcap1
```

Zatim se komandom cmp može utvrditi da su /etc/termcap i /tmp/tcap1 iste datoteke. Dodatno, korisnik može uporediti početak i kraj ovih datoteka komandama head i tail.

```
$ head /etc/termcap
$ head /tmp/tcap1
```

Sledeći korak je brisanje hard linka, nakon čega se broj linkova originalne datoteke smanjuje za 1.

```
$ rm /tmp/tcap1
$ ls –l /etc/termcap
-rw-r--r--   1   root   root   729360   Mar 14 2001   /etc/termcap
```

Primer koji sledi ilustruje nemogućnost linkovanja direktorijuma ili nepostojeće datoteke:

```
$ ln /etc ~/dir_etc
ln: `/etc': hard link not allowed for directory
$ ln unexisting_file ~/junk
ln: accessing `unexisting_file': No such file or directory
```

Simbolički linkovi

Simbolički link (symbolic link, symlink) je prečica ka objektu u sistemu datoteka. Princip korišćenja simboličkih linkova je identičan principu korišćenja hard linkova: svaka referenca na link odnosi se na originalnu datoteku, osim ukoliko se link poziva iz komande rm.

Sa tačke gledišta korisnika, simbolički link pruža veću fleksibilnost jer dozvoljava linkovanje na direktorijume, nepostojeće datoteke i datoteke koje se nalaze na drugom sistemu datoteka (lokalnom ili mrežnom, domaćem ili stranom). Sa tačke gledišta Linux administratora, simbolički link je jednostavna referenca, odnosno zaseban objekat sistema datoteka koji koristi jedan i-node i jedan blok podataka u kom je zapisana lokacija originalnog objekta. Kao takav, simbolički link zauzima određen prostor na disku i jedno mesto u i-node tabeli.

Simbolički linkovi se koriste u situacijama kada korisnici pristupaju datotekama u nekom direktorijumu čija je putanja dugačka za pisanje. Na primer, neka u home direktorijumu korisnika jsmith postoji direktorijum /project1/public, takav da je svim korisnicima sistema dozvoljen pristup tom direktorijumu. Korisnik user2 može se prijaviti na sistem i pristupiti datotekama u tom direktorijumu:

```
$ less /home/jsmith/project1/public/letter1.txt
```

Korisnik može iskopirati potrebne datoteke u svoj home direktorijum, ali će to morati da uradi ponovo svaki put kad neko izvrši modifikaciju sadržaja datoteka. Dodatno, ovim se troši prostor na disku, što nije poželjno u slučajevima kada sistem administrator postavi disk kvote.

Jednostavnije rešenje je da korisnik user2 kreira simbolički link u svom home direktorijumu koji će ukazivati na direktorijum /home/jsmith/project1/public. Simbolički link se jednostavno može kreirati komandom ln -s:

```
$ cd
$ ln –s /home/jsmith/project1/public jspub
```

Simbolički link jspub ukazuje na direktorijum /home/jsmith/project1/public. Koristeći ovaj link, korisnik user2 može jednostavno pristupati direktorijumu public:

```
$ ls jspub
$ less jspub/letter1.txt
```

Kreiranje simboličkih linkova

Simbolički linkovi se mogu kreirati na dva načina: pomoću komande ln i pomoću komande cp. Sintaksa ovih komandi je:

```
$ ln –s original linkname
$ cp –s original linkname
```

Argument original je ime originalne datoteke, a argument linkname ime simboličkog linka.

Osobine simboličkih linkova

Simbolički linkovi su objekti sistema datoteka koji ukazuju na druge objekte (prečice). Simboličke linkove karakterišu sledeće osobine:

- svaki simbolički link koristi poseban i-node i jedan blok podataka u sistemu datoteka. Simbolički linkovi se mogu kreirati nalaziti na fizički istom ili različitom sistemu datoteka, odnosno na istoj ili drugoj particiji (disku). Takođe, mogu se linkovati datoteke sa mrežnog sistema datoteka (NFS);
- može se linkovati direktorijum, kao i nepostojeća datoteka;
- u odnosu na original, link može imati različitog vlasnika, grupu i prava pristupa. Na korisnika koji datoteci ili direktorijumu pristupa putem simboličkog linka primenjuje se unija restrikcija (presek dozvola) linka i datoteke. Na primer, neka je korisnik user2 vlasnik linka link1 koji ukazuje na datoteku file1, i nek pripada grupi kojoj je ta datoteka formalno dodeljena. Ukoliko su pristupna prava za link i datoteku 777 i 640 respektivno, korisnik će imati samo pravo čitanja te datoteke;
- slobodan prostor na disku se umanjuje (za jedan blok podataka). Takođe, simbolički link troši jedan i-node iz i-node tabele;
- broj linkova originalne datoteke se ne uvećava za jedan nakon linkovanja, već ostaje isti kao pre linkovanja;
- s obzirom da simbolički link može ukazivati na nepostojeći objekat, originalna datoteka se može obrisati sa diska bez obzira na broj simboličkih linkova koji upućuju na nju.

Primer kreiranja i upotrebe simboličkih linkova

Za potrebe ovog primera upotrebljena je velika datoteka /etc/termcap. Najpre se određuju atributi originalne datoteke (vlasnik, grupa, prava pristupa).

```
$ ls -l /etc/termcap
-rw-r--r--   1   root   root   729360   Mar 14 2001   /etc/termcap
```

Nakon toga se kreira jedan simbolički link datoteke na tmp direktorijumu i određuju njegovi atributi.

```
$ whoami
bora
$ ln -s /etc/termcap /tmp/tcap2
$ ls -l /tmp/tcap2
lrwxrwxrwx   1   bora   bora   12   Sep 5  14:35   /etc/termcap ->
/etc/termcap
```

Zatim se komandom cmp može utvrditi da su /etc/termcap i /tmp/tcap2 iste datoteke. Dodatno, korisnik može uporediti početak i kraj ovih datoteka komandama head i tail.

```
$ head /etc/termcap
$ head /tmp/tcap2
$ rm /tmp/tcap2
```

Sledeći primer ilustruje linkovanje direktorijuma i nepostojeće datoteke:

```
$ ln -s /etc dir_etc
$ ls -l dir_etc
lrwxrwxrwx   1   root   root    4   Sep 5  14:40   dir_etc -> /etc
$ ls -l unexist
ls: unexis: No such file or directory
$ ln -s unexist junk
$ ls -l junk
lrwxrwxrwx   1   root   root   15   Sep 5  14:40   junk -> unexist
```

Upotreba opcije -d komande cp (no-dereference)

Prilikom kopiranja simboličkih linkova vrši se ukidanje reference (dereference), odnosno kopira se originalna datoteka na koju link ukazuje. Osim komande cp i veliki broj drugih komandi za rad sa datotekama primenjuje dereferenciranje. Ukoliko postoji potreba da se iskopira link, a ne originalna datoteka, komanda cp se izvršava sa opcijom -d. U tom slučaju se ne vrši dereferenciranje. To je moguće izvesti i prilikom drugih operacija (kao što je kreiranje arhive), a opcije kojimam se dereferenciranje isključuje zavise od konkretne komande. Sledeći primer ilustruje upotrebu opcije -d komande cp:

```
$ ls -l /etc/termcap
-rw-r--r--   1   root   root   729360   Mar 14 2001   /etc/termcap
$ ln -s /etc/termcap tc1
$ cp tc1 tc2
$ cp -d tc1 tc3
$ ls -l tc2 tc3
-rw-r--r--   1   bora   bora   729360   Sep 5 14:37   tc2
```

```
lrwxrwxrwx   1   bora   bora       12   Sep 5 14:37   tc3 ->
/etc/termcap
```

Rad sa direktorijumima

Navigacija po direktorijumskom stablu. Prikazivanje sadržaja direktorijuma. Kreiranje i brisanje direktorijuma. Kopiranje, pomeranje i brisanje direktorijumskog stabla.

Sa tačke gledišta običnog korisnika, direktorijum je fascikla u kojoj se nalaze datoteke. Sa tačke gledišta sistem administratora, direktorijum je specijalna datoteka koja u sebi sadrži imena drugih objekata sistema datoteka (uključujući i poddirektorijume) i pokazivače na i-node strukture kojima su ti objekti u potpunosti opisani.

Svaki sistem datoteka ima svoj koreni direktorijum (root) u kome se nalaze drugi direktorijumi i datoteke. Svi direktorijumi jednog sistema datoteka čine njegovo direktorijumsko stablo. Sistemi datoteka su montirani na odgovarajuće mount-point direktorijume. Stabla svih aktiviranih sistema datoteka čine aktivno UNIX stablo.

Kretanje po direktorijumskom stablu

Korisnik se u svakom trenutku nalazi na jednoj lokaciji u aktivnom UNIX stablu. Direktorijum u kom se korisnik trenutno nalazi naziva se tekući ili radni direktorijum (current directory, working directory). Svaki korisnik može videti tekući direktorijum u odzivu komandnog interpretera, ukoliko je promenljivoj okruženja PS1 dodeljena vrednost koja uključuje promenljivu $PWD:

```
$ PS1='Dir: $PWD $'
Dir: mydoc $
```

Ovako postavljen, odziv će prikazati samo zadnji poddirektorijum, odnosno relativnu putanju ka tekućem direktorijumu iz roditeljskog direktorijuma. Apsolutna putanja tekućeg direktorijuma može se prikazati pomoću komande pwd (print working directory):

```
Dir: mydoc $ pwd
/home/jsmith/mydoc
```

Tekući direktorijum se menja komandom cd (change directory). Sintaksa komande je krajnje jednostavna:

```
$ cd dir
```

Argument dir je apsolutna ili relativna putanja direktorijuma na koji korisnik želi da se pozicionira. Ukoliko takav direktorijum postoji i korisnik ima dovoljna prava da se na njega pozicionira, nakon pokretanja komande dir će postati tekući direktorijum.

Korisnik koji zada komandu bez argumenta dir biće premešten u direktorijum specificiran promenljivom $HOME (u podrazumevanom stanju, to je home direktorijum korisnika).

Apsolutna putanja direktorijuma koji je naveden računa se kao relativna putanja u odnosu na tekući direktorijum. Na primer, ako se korisnik nalazi u direktorijumu /tmp/backup i zada komandu cd ../data, biće prebačen u direktorijum /tmp/data. Ukoliko direktorijum ne postoji korisnik će na ekranu dobiti sledeću poruku:

```
$ cd ../data
bash: cd: ../data: No such file or directory
```

Promenljiva CDPATH specificira dodatne direktorijume koje se koriste za računanje apsolutne putanje. Na primer, neka je u promenljivoj CDPATH naveden direktorijum /mnt/net1. Ako se korisnik nalazi u direktorijumu /tmp i zada komandu cd data, biće prebačen u direktorijum /tmp/data. Ukoliko direktorijum /tmp/data ne postoji, komanda cd će pokušati da prebaci korisnika u direktorijum /mnt/net/data. Ukoliko direktorijum ne postoji korisnik će na ekranu dobiti poruku da direktorijum ne postoji.

Sledeći primeri korišćenja komande cd redom ilustruju:

- prelazak na direktorijum specificiran promenljivom HOME (najčešće home direktorijum korisnika),
- prelazak na home direktorijum korisnika,
- prelazak na roditeljski direktorijum,
- prelazak na direktorijum backup (direktorijum se traži u tekućem direktorijumu i direktorijumima specificiranim u promenljivoj CDPATH),
- prelazak na root direktorijum aktivnog UNIX stabla,
- prelazak na direktorijum backup koji se nalazi u root direktorijumu aktivnog UNIX stabla,
- prelazak na direktorijum backup koji se nalazi u roditeljskom direktorijumu tekućeg direktorijuma,

```
$ cd
$ cd ~
$ cd ..
$ cd backup
$ cd /
$ cd /backup
$ cd ../temp
```

Prikazivanje sadržaja direktorijuma

Korisnik može na ekranu pomoću komande ls (list) prikazati sadržaj bilo kog direktorijuma aktivnog UNIX stabla. Sintaksa komande ls je:

```
$ ls [options] [dir][filespec]
```

Komanda ls prikazaće na ekranu spisak objekata direktorijuma dir definisanih argumentom filespec. Argument filespec formira se pomoću džoker karaktera i nije obavezan. Ukoliko se ne navede, podrazumevaju se svi objekti u direktorijumu. Argument dir takođe nije obavezan, i ako se ne navede prikazuje se sadržaj tekućeg direktorijuma.

Ukoliko se ne navedu dodatne opcije ls prikazuje samo imena objekata sortiranih u abecednom redu. Imena skrivenih objekata (objekti čije ime počinje tačkom) se ne prikazuju, a imena rezervnih kopija datoteka (datoteke čije se ime završava znakom ~) se prikazuju.

Od značajnijih opcija komande ls navode se sledeće:

-a	prikazuju se i imena skrivenih objekata
-B	imena rezervnih kopija datoteka se ne prikazuju
-d	prikazuje se kontekst direktorijuma umesto sadržaja
-i	prikazuje se i-node broj datoteke
-R	rekurzivno se prikazuje sadržaj svih poddirektorijuma
-L	dereferenciranje (umesto simboličkih linkova se prikazuju imena datoteka na koje linkovi upućuju)
-h	veličine datoteka se prikazuju u čitljivom formatu (1K, 234M, 2G)
-k	veličine datoteka se prikazuju u kilobajtima
-l	osim imena objekata, prikazuju se i informacije upisane u i-node (prava pristupa, vlasnik, grupa, datum, vreme)
-1	prikazuje samo imena objekata (jedno ime u jednoj liniji)
-r	imena objekata se prikazuju sortirana u opadajućem redosledu
--sort=WORD	definisanje kriterijuma za sortiranje imena objekata (na primer: ekstenzija -X, bez sortiranja -U, veličina -S).

Sledeći primeri ilustruju upotrebu komande ls:

- prikazivanje inverzno sortiranih imena objekata tekućeg direktorijuma

```
$ ls -r
l1  f2  f1  d1
```

- prikazivanje imena svih objekata tekućeg direktorijuma

```
$ ls -a
.   .alias         .bash_profile  .cshrc  d1  f2
..  .bash_history  .bashrc        .gnupg  f1  l1
```

- prikazivanje konteksta svih objekata tekućeg direktorijuma

```
$ ls -l
total 16
drwxr-xr-x   2   nmacek   nmacek   4096   Sep 21   d1
-rw-r--r--   1   nmacek   nmacek    315   Sep 21   f1
-rw-r--r--   1   nmacek   nmacek   4125   Sep 21   f2
lrwxrwxrwx   1   nmacek   nmacek      2   Sep 21   l1 -> f1
```

- prikazivanje imena objekata tekućeg direktorijuma i svih poddirektorijuma

```
$ ls -R
.:
d1  f1  f2  l1

./d1:
f11
```

- prikazivanje konteksta direktorijuma /etc

```
$ ls -dl /etc
drwxr-xr-x   53 root     root         3072 May 21 09:58 /etc
```

- prikazivanje i-node brojeva svih objekata tekućeg direktorijuma

```
$ ls -i1
  31782 d1
  95163 f1
  95164 f2
  95165 l1
```

Kreiranje direktorijuma

Direktorijumi se kreiraju komandom mkdir (make directory), čija je sintaksa:

```
$ mkdir [options] dir
```

Komanda mkdir kreira direktorijum dir ukoliko on već ne postoji i dodeljuje mu inicijalna prava pristupa na osnovu vrednosti promenljive umask. Ukoliko se u argumentu dir ne navede putanja, direktorijum će biti kreiran u tekućem direktorijumu. Od značajnijih opcija komande mkdir navode se sledeće:

-mMODE direktorijumu se nakon kreiranja dodeljuju prava pristupa specificirana oktalnim režimom MODE

-p mkdir će kreirati i roditeljski direktorijum, ukoliko on već ne postoji

Brisanje direktorijuma

Direktorijum se može obrisati komandama rm i rmdir (remove directory). Komandom rmdir može se obrisati isključivo prazan direktorijum. Sintaksa komande rmdir je:

```
$ rmdir [-p] dir
```

Ukoliko se u argumentu dir ne navede putanja, briše se direktorijum dir iz tekućeg direktorijuma.Ukoliko se navede opcija –p, komanda rmdir uklanja kompletno stablo roditeljskih direktorijuma, pod uslovom da su prazni. Na primer, komanda

```
$ rmdir -p /backup/tape1/hd
```

radi isto što i:

```
$ rmdir /backup/tape1/hd
$ rmdir /backup/tape1
```

```
$ rmdir /backup
```

Komanda rm je moćnija – za razliku od komande rmdir, rekurzivnim brisanjem komandom rm mogu se ukloniti cela direktorijumska stabla sa datotekama. Na primer, ukoliko bi se u direktorijumu iz prethodnog primera nalazila jedna datoteka (/backup/tape1/hd/hd1.tar.gz) korisnik ne bi mogao da ukloni stablo komandom rmdir, već samo komandom rm:

```
$ rm -r /backup
```

Kopiranje direktorijuma

Direktorijumi se kopiraju komandom cp u režimu rekurzivnog kopiranja direktorijumskog stabla, kao što je ranije opisano. Sintaksa komande cp za kopiranje direktorijumskog stabla je:

```
$ cp -r dir1 dir2
```

Argumenti dir1 i dir2 mogu se navesti kao apsolutne ili relativne putanje, i u opštem slučaju, kopija se formira u obliku dir2\dir1. Direktorijum se ne može iskopirati pod drugim imenom.

Pretraživanje direktorijuma

Jedna od povoljnosti koju UNIX pruža korisnicima je praktično neograničena dužina putanje objekta. Svaki direktorijum može imati poddirektorijum, a u svakom direktorijumu se mogu naći i datoteke. Struktura drveta formirana na ovaj način nije ograničena u smislu dubine (na primer, ne postoji ograničenje na deset nivoa poddirektorijuma), tako da korisnici podatke mogu organizovati onako kako im odgovara. Međutim, ukoliko korisnik želi da radi nešto sa datotekom koja se ne nalazi u tekućem direktorijumu, on mora navesti uz ime datoteke i putanju, što može biti nezgodno ukoliko je putanja dugačka. Jedno rešenje je da korisnik napravi simboličke linkove na značajne datoteke i direktorijume. Međutim, praksa pokazuje da većina korisnika ne kreira linkove, već jednostavno iskopira datoteke na "logično" mesto u aktivnom stablu. Praksa takođe pokazuje da veliki broj korisnika već sledeći put ne može da se seti lokacije nekih datoteka.

UNIX rešava problem izgubljenih datoteka pomoću komande find. Komanda find traži datoteke čiji atributi zadovoljavaju kriterijume pretrage u direktorijumu koji je naveden kao početna tačka pretrage i svim poddirektorijumima, rekurzivno. Ukoliko korisnik drugačije ne naznači, pretraga prelazi granice sistema datoteka. Komanda find na ekranu može prikazati apsolutne putanje i imena pronađenih datoteka ili izvršiti neke operacije nad njima.

Sledeći primer ilustruje najjednostavniji slučaj pretrage – u celom aktivnom stablu traži se datoteka poznatog imena (urgent.txt):

```
$ find / -name urgent.txt -print
/tmp/jsmith/temp/urgent.txt
```

Komplikovaniji slučaj je pretraga po drugim atributima – na primer, traže se sve datoteke koje pripadaju korisniku jsmith veličine najmanje 50 blokova:

```
$ find /tmp –user jsmith –size +50 – print
```

Napomena: ono što je neobično za find komandu je to da ona nema svoj prirodni izlaz. Komanda find će pronaći sve datoteke koje ispunjavaju određene uslove, ali ukoliko korisnik ne naznači komandi šta da uradi sa datotekama koje pronađe, komanda neće izvršiti nikakvu akciju. Na nekim Linux sistemima, podrazumevana akcija je prikazivanje imena datoteka na ekranu.

Argumenti komande find mogu se klasifikovati u tri kategorije:

- kriterijumi pretrage,
- akcioni izrazi,
- kvalifikatori pretrage.

Komanda find sadrži jedan ili više kriterijuma pretrage, jedan akcioni izraz, i opciono kvalifikator pretrage. Drugim rečima, find pronalazi datoteke na osnovu zadatih kriterijuma i nad njima izvršava neku akciju, makar to bilo i najjednostavnije prikazivanje na ekranu.

Kriterijumi pretrage

Prvi zadatak komande find je da odredi lokaciju svih datoteka čiji atributi odgovaraju kriterijumu koji je korisnik naveo. Datoteke se mogu tražiti na osnovu imena, veličine, vlasnika, vremena ili i-node broja. U najjednostavnijem slučaju, ukoliko je poznato ime ili deo imena datoteke, koristi se opcija -name, a komanda find se zadaje na sledeći način:

```
$ find /tmp –name letter1.doc –print
$ find /home/jsmith –name "*.old" –print
```

Ove komande će na ekranu prikazati putanju i ime svih datoteka čije je ime letter1.txt, odnosno svih datoteka čija je ekstenzija .old.

Napomena: u drugom slučaju ime datoteke sadrži džoker karaktere, tako da se mora staviti pod navodnike. Navodnici sprečavaju shell da izvrši zamenu imena datoteka pre izvršenja komande. Ukoliko se ime koje sadrži džoker karaktere ne stavi pod navodnike, shell će ga zameniti imenima svih datoteka koje odgovarju tom izrazu, a nakon toga izvršiti komandu find. Na primer, ukoliko se u tekućem direktorijumu nalaze datoteke a.old, b.old i c.old izvršiće se sledeća komanda koja ne obavlja željenu aktivnost:

```
$ find /home/jsmith –name a.old b.old c.old –print
```

Sledeći zanimljiv kriterijum pretrage je veličina datoteke. Argument pretrage -size dozvoljava parametre u obliku -n, n ili +n, gde je n ceo broj. U tom slučaju se traže datoteke čija je veličina manja od n, tačno n ili veća od n sistemskih blokova. Sledeća komanda prikazuje imena svih datoteka koje se nalaze u direktorijumu /tmp, a koje su veće od 100 sistemskih blokova.

```
$ find /tmp –size +100 –print
```

Kriterijum veličine datoteke može sadržati i sufiks c, što znači da je veličina datoteke izražena u bajtovima. Sledeća komanda će na ekranu prikazati imena svih datoteka čija je veličina manja od 512 bajtova:

```
$ find /tmp –size –512c –print
```

Ostali kriterijumi pretrage su:

-username uname	traže se datoteke čiji je vlasnik korisnik uname;
-groupname gname	traže se datoteke koje su formalno dodeljene grupi gname;
-atime n	traže se datoteke kojima niko nije pristupio tačno n dana (n mora biti ceo broj, a dozvoljeni su i oblici -n i +n);
-mtime n	traže se datoteke koje niko nije modifikovao tačno n dana (n mora biti ceo broj, a dozvoljeni su i oblici -n i +n);
-perm mode	traže se datoteke čija su prava pristupa u oktalnom obliku jednaka parametru mode. Ako se navede -perm -onum, traže se one datoteke čija su pristupna prava nadskup prava definisanih parametrom onum. Na primer, ako se navede -perm 666, traže se sve datoteke čija su prava r i w za sve tri vlasničke kategorije. Ako se navede -perm -111, traže se sve datoteke čija prava uključuju pravo izvršavanja za kategorije vlasnika i grupe, bez obzira na ostala prava;
-links n	traže se sve datoteke sa n hard linkova (n mora biti ceo broj, a dozvoljeni su i oblici -n i +n);
-type x	traže se sve datoteke koje su tipa x, pri čemu x može biti b (blok uređaj), c (karakter uređaj), d (direktorijum), p (imenovani pipe);
-inode n	traže se sve datoteke čiji je i-node n;
-newer fname	traže se sve datoteke koje su modifikovane pre datoteke fname;
-local	traže se sve datoteke koje se nalaze na lokalnim diskovima.

Kada komanda find pronađe datoteke, izvršava operacije specificirane akcionim izrazima.

Akcioni izrazi

Ukoliko se kao akcioni izraz navede -print, komanda find će na ekranu prikazati apsolutne putanje i imena datoteka. To se u praksi često koristi pre nego što se nad datotekama izvrši konkretna akcija, odnosno neka druga UNIX komanda.

Ukoliko se kao akcioni izraz navede

```
–exec cmd {} \;
```

komanda find će pokrenuti komandu cmd čiji će parametri biti putanje i imena pronađenih datoteka. Najmoćnija stvar kod komande find je jedinstveni metod zamene imena datoteka. Na primer, pretpostavimo da:

```
$ find /usr/home –name list.txt –print
```

daje rezultat:

```
/usr/home/dave/list.txt
/usr/home/marsha/list.txt
/usr/home/mike/list.txt
```

Komanda:

```
$ find /usr/home –name list.txt –exec rm {} \;
```

izvršava isto što i:

```
$ rm /usr/home/dave/list.txt
$ rm /usr/home/mike/list.txt
$ rm /usr/home/marsha/list.txt
```

Dodatno, komanda find nudi mogućnost da nakon svake pronađene datoteke korisnik pita da li nad tom datotekom želi da izvrši akciju ili ne. U tom slučaju se umesto akcionog izraza exec koristi akcioni izraz ok.

```
$ find /usr/home –name list.txt –exec rm {} \;
```

Još jedna zanimljiva mogućnost koja se može realizovati akcionim izrazom je kreiranje rezervne kopije pronađenih datoteka:

```
$ find /usr/home –name "*.doc" –cpio –o > /dev/rmt0
```

što je identično pipeline sprezi:

```
$ find /usr/home –name "*.doc" –print | cpio –o > /dev/rmt0
```

Kvalifikatori pretrage

Kvalifikatorima pretrage se modifikuje podrazumevana pretraga. Najčešće korišćeni kvalifikator je -mount, kojim se sprečava da pretraga pređe granice sistema datoteka. Na primer, komanda find u sledećem obliku će pronaći sve datoteke sa ekstenzijom .old u root sistemu datoteka:

```
$ find / –mount –name "*.old" –print
```

Složeni kriterijumi pretrage

Kriterijumi pretrage mogu se kombinovati - više izraza navedenih u komandi find pomoću I operatora formira složeni kriterijum. Na primer, sledeća komanda će iz direktorijuma /tmp obrisati sve datoteke sa ekstenzijom tmp kojima niko nije pristupao najmanje 7 dana.

```
$ find /tmp –name "*.tmp" –atime +7 –exec rm {} \;
```

Pretpostavimo dalje da korisnik želi da pronađe datoteke sa ekstenzijom .tmp ili .temp. Jedna mogućnost je zadavanje dve find komande. Druga mogućnost je korišćenje uslova –name "*mp", ali u tom slučaju kriterijum pretrage nije verodostojan (kriterijum zadovoljavaju i datoteke blimp, romp, stomp i slične). Ovaj problem se najjednostavnije može rešiti kombinovanjem kriterijuma pretrage logičkim ILI izrazom:

```
$ find /tmp \( –name "*.tmp" –o –name "*.temp" \)
```

Takođe, korisnik može negirati uslov pretrage. Na primer, korisnik može obrisati iz svog home direktorijuma sve datoteke kojima on nije vlasnik:

```
$ find /home/jsmith ! –user jsmith –exec rm {} \;
```

Rad sa tekstualnim datotekama

Pregledanje sadržaja tekstualne datoteke. Brojanje karaktera, reči i linija. Upoređivanje i uređivanje sadržaja datoteka. Traženje teksta u datoteci. Ekranski editori teksta vi, joe i jed.

Kako su sve konfiguracione datoteke UNIX i Linux sistema tekstualne (na primer, /etc/passwd), administrator UNIX, odnosno Linux sistema treba da poznaje osnovne alate za rad sa tekstualnim datotekama. U nastavku teksta opisane su osnovne komande za rad sa tekstualnim datotekama, uključujući poznatije ekranske editore vi, joe i jed.

Pregledanje sadržaja tekstualne datoteke

Sadržaj tekstualne datoteke najlakše se može pregledati pomoću programa cat, more i less. U zavisnosti od veličine datoteke korisnik će odrediti koji će program koristiti - ukoliko je datoteka kraća i može se prikazati na jednom ekranu, može se koristiti cat. U suprotnom, koristi se program less koji prikazuje datoteku sa pauzom nakon svakog punog ekrana.

cat

Komanda cat služi za povezivanje datoteka i njihovo prikazivanje na standardnom izlazu (najčešće na ekranu). Ime je dobila na osnovu funkcije koju obavlja - Concatenate and Append to the Terminal. Sintaksa komande cat je :

```
$ cat [–s] filename
```

U ovom obliku komanda cat prikazuje sadržaj datoteke filename na ekranu, bez zaustavljanja nakon svakog punog ekrana. Korisna opcija programa cat je -s (squeeze blank lines), kojom se iz prikazanog teksta izbacuje višak praznog prostora, odnosno nikad se ne prikazuje više od jedne prazne linije.

Sledeći primeri ilustruju prikazivanje sadržaja datoteke /etc/resolv.conf i konkatenacije datoteka /etc/resolv.conf i /etc/networks na ekranu, respektivno:

```
$ cat /etc/resolv.conf
search internal.vets.edu.yu
nameserver 172.16.32.1
$ cat /etc/resolv.conf /etc/networks
search internal.vets.edu.yu
nameserver 172.16.32.1
localnet 172.16.0.0
```

more

Komanda more se koristi za prikazivanje sadržaja datoteke ekran po ekran. U praksi se najčešće koristi naprednija varijanta (program less), tako da se komanda more navodi iz istorijskih razloga. Sintaksa komande more je :

```
$ more filename
```

Nakon zadavanja komande more prikazuje prvi ekran sadržaja datoteke, nakon čega se koriste tasteri <Space> za prikazivanje sledećeg punog ekrana, <Enter> za prikazivanje sledeće linije teksta i q ili Q za napuštanje programa.

less

Ova komanda je slična komandi more, s tim što dozvoljava dvosmernu navigaciju po datoteci (unapred i unazad). Program less ne mora da pročita celu datoteku pre nego što prikaže prvi ekran sadržaja datoteke, tako da u odnosu na editor vi radi brže sa velikim datotekama. Less koristi datoteku termcap (terminfo na nekim sistemima), tako da se može koristiti na različitim terminalima. Sintaksa komande less je:

```
$ less [-[+]ENsS] [-ologfile] filename
```

Argument filename je ime datoteke čiji sadržaj treba prikazati na ekranu, a značajnije opcije imaju sledeće značenje:

-E napušta program kada se dođe do kraja datoteke (EOF - end of file). Ukoliko se ova opcija ne navede, less se može napustiti interaktivnom komandom q (quit) ili slanjem signala za prekidanje izvršenja procesa kombinacijom tastera <Ctrl-C>;

-N ispisuje redni broj linije na početku svake linije na ekranu;

-s iz prikazanog teksta se izbacuje višak praznog prostora, odnosno nikad se ne prikazuje više od jedne prazne linije;

-S duge linije se seku i onaj deo koji se ne može prikazati prenosi se u novu liniju na ekranu (wrap text);

-ologfile ukoliko se navede opcija -o, ulazna datoteka se kopira u datoteku logfile. Ova opcija se može navesti ukoliko je ulazna datoteka pipe.

Program less ima veliki skup dodatnih opcija koje radi jednostavnosti opisa komande ovde nisu navedene. Potpuna dokumentacija o korišćenju programa less dostupna je u odgovarajućoj stranici uputstva (man less). Korisnik koji ne želi stalno da navodi iste parametre, može definisati promenljivu okruženja LESS i u njoj pobrojati parametre koji će se prosleđivati komandi less prilikom svakog izvršavanja.

```
$ export LESS="-options"
```

Karakteristike terminala koje su neophodne za rad sa ekranom, program less čita iz promenljive TERM.

Nakon pokretanja programa less prikazuje prvi ekran sadržaja datoteke, a zatim se koriste interaktivne komande za navigaciju po sadržaju datoteke i napuštanje programa:

h ili H	prikazuje pomoć pri korišćenju interaktivnih komandi
<Space>	prikazuje sledeći ekran sadržaja datoteke
b	prikazuje prethodni ekran sadržaja datoteke
<DOWN>	prikazuje sledeću liniju teksta
<UP>	prikazuje prethodnu liniju teksta
<RIGHT>	horizontalno pomeranje ekrana udesno (ukoliko se ne koristi word wrap)
<LEFT>	horizontalno pomeranje ekrana ulevo (ukoliko se ne koristi word wrap)
F	nastoji da prati kraj datoteke. Opcija je korisna ukoliko se na ekranu prati sadržaj datoteke u koju neki drugi proces istovremeno upisuje podatke. (slično komandi tail - f, koja je opisana u ovom poglavlju).
g	prikazuje se prva linija datoteke
G	prikazuje se poslednja linija datoteke
N	prikazuje se n-ta linija datoteke (korisnik unosi broj linije)
=	prikazuje tekući broj linije i procenat pročitanog sadržaja datoteke
q ili Q	napuštanje programa less.

Dodatno, za program less može se definisati ulazni predprocesor (input preprocessor). Predprocesor je izvršni program (ili shell skript) koji obrađuje datoteku pre nego što je less prikaže na ekranu. Ulazni predprocesor definiše se promenljivom LESSOPEN. Primer ulaznog predprocesora je program gzip kojim se može izvršiti dekompresija tekstualne datoteke pre prikazivanja na ekranu.

Sledeći primer ilustruje korišćenje komande less:

```
$ less /etc/passwd
$ ls /etc/ | less
```

Prikazivanje početka i kraja datoteke (komande head i tail)

Komanda head na standardnom izlazu prikazuje početak datoteke. Sintaksa komande je:

```
$ head [-n] filename
```

Opcijom –n, gde je n celobrojna vrednost, korisnik određuje broj linija koje komanda head treba da prikaže na ekranu (ukoliko se ne navede, podrazumeva se 10).

Za razliku od komande head, čija upotrebna vrednost nije velika, komanda tail se češće koristi. Tail prikazuje kraj datoteke na standardnom izlazu. Značenje opcije–n je identično kao i za komandu head - korisnik određuje broj linija koje komanda tail treba da prikaže na ekranu (ukoliko se ne navede, podrazumeva se 10). Sintaksa komande tail je:

```
$ tail [-n] [-f [--pid=PID]] file
```

Opcije -f i --pid imaju sledeće značenje: -f nalaže programu tail da prati rast datoteke (follow), a ukoliko se navede i --pid=PID, komanda završava rad sa završetkom rada procesa PID. Sa ovim opcijama, komanda tail se može iskoristiti za praćenje podataka koje neki program upisuje u datoteku.

Sledeći primer ilustruje korišćenje komandi head i tail:

```
$ head -3 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
$ tail -3 /etc/passwd
rt8900:x:888:890::/home/rt8900:/bin/bash
rt93799:x:889:891::/home/rt93799:/bin/bash
nrt2700:x:890:892::/home/nrt2700:/bin/bash
```

Brojanje karaktera, reči i linija

Komanda wc (word count) se koristi za brojanje karaktera, reči i linija u datoteci. Sintaksa komande je:

```
$ wc [-cwl] filename
```

Bez opcija komanda prikazuje sve tri vrednosti, dok se opcijama -c, -w i -l specificira prebrojavanje karaktera (bajtova), reči ili linija. Datoteka /etc/protocols je iskorišćena radi ilustracije upotrebe komande wc:

```
$ wc -c /etc/protocols
   1748 /etc/protocols
$ wc -w /etc/protocols
    297 /etc/protocols
$ wc -l /etc/protocols
     44 /etc/protocols
$ wc /etc/protocols
     44     297    1748 /etc/protocols
```

Komanda wc je korisna u raznim situacijama - pomoću nje se jednostavno može odrediti ukupan broj korisnika sistema, ili broj datoteka u direktorijumu:

```
$ wc -l /etc/passwd
     78 /etc/passwd
```

```
$ ls -w /etc | wc -l
    145
```

Upoređivanje sadržaja datoteka

Najčešće korišćeni programi za upoređivanje datoteka su cmp (compare) i diff (difference).

Komanda cmp

Program cmp upoređuje dve datoteke i prikazuje rezultat upoređivanja na standardnom izlazu (ekran). Ukoliko se ne navede opcija -s, program cmp:

- ne štampa ništa na ekranu ukoliko su datoteke jednake,
- ukoliko su datoteke različite prikazuje broj karaktera i linije u kojoj je pronađena prva razlika. Ukoliko je jedna datoteka identična početku druge datoteke, cmp prikazuje poruku da je pri upoređivanju stigao do kraja jedne datoteke, pre nego što su pronađene razlike.

Sintaksa komande cmp je:

```
$ cmp [-l | -s] file1 file2 [skip1 [skip2]]
```

gde su file1 i file2 datoteke koje se upoređuju. Opcije imaju sledeće značenje:

-l	prikazuje broj svakog karaktera koji se razlikuje i oktalnu vrednost ASCII koda tog karaktera,
-s	ne prikazuje rezultat na ekranu, već samo vraća izlazni status (silent).

Opcioni argumenti skip1 i skip2 su ofset u bajtovima od početka datoteka file1 i file2, odnosno redni brojevi karaktera od kojih počinje da se vrši upoređivanje. Ofset se navodi kao decimalni broj.

Komanda cmp vraća sledeće vrednosti kao izlazni status:

0	datoteke su identične,
1	datoteke se razlikuju (ovaj slučaj uključuje mogućnost da je jedna datoteka identična početku druge datoteke),
>1	pojavila se greška.

Izlazni status komande cmp je od značaja za shell programiranje. Sledeći primer ilustruje upotrebu komande cmp:

```
$ cmp dat1 dat2
dat1 dat2 differ: char 653, line 18
$ cmp dat1 dat3
cmp: EOF on dat3
$ cmp dat1 dat4
```

Komanda diff

Program diff je moćniji od programa cmp. Diff, kao što i samo ime kaže pronalazi sve razlike između datoteka. Sintaksa komande diff je:

```
$ diff [options] from-file to-file
```

U najjednostavnijem obliku komanda diff upoređuje sadržaj datoteke from-file sa sadržajem datoteke to-file. Ukoliko je from-file direktorijum, upoređuju se datoteke from-file/to-file i to-file (slično se postupa i ako je to-file direktorijum). Ukoliko su oba parametra direktorijumi, diff upoređuje datoteke sa istim imenima iz ovih direktorijuma. Ovo upoređivanje nije rekurzivno, osim ako se navede opcija -r.

Opcije koje se tiču detekcije razlika u sadržaju datoteka su:

-i	diff ignoriše razlike između malih i velikih slova
-w	diff ignoriše sav whitespace (odnosno tab karaktere i razmaknice)
-b	diff ignoriše sve blanko karaktere kojima se završava linija u datoteci.

U ostale opcije spadaju:

--brief	diff samo prijavljuje da li su datoteke različite ili ne
-r	rekurzivno upoređivanje datoteka po poddirektorijumima
-y	rezultat upoređivanja prikazuje se u dve kolone (side-by-side).

Komanda diff vraća sledeće vrednosti kao izlazni status:

0	datoteke su identične
1	datoteke se razlikuju
>1	pojavila se greška.

Izlazni status komande diff je od značaja za shell programiranje. Sledeći primer ilustruje upotrebu komande diff:

```
$ diff letter1.txt letter2.txt
2c2
< Please go away.
---
> Please go away now.
4c4
< This is dr. Jackyl !
---
> This is mr. Hyde !
$ diff -y letter1.txt letter2.txt
I do not want to talk to you.      I do not want to talk to you.
Please go away.                  | Please go away now.
Hey !                              Hey !
This is dr. Jackyl !             | This is mr. Hyde !
```

Uređivanje sadržaja datoteka

Uređivanje sadržaja datoteka po određenom kriterijumu (sort)

Sadržaj tekstualnih datoteka se programom sort može urediti u rastućem ili opadajućem redosledu. Uređivanje se vrši po alfanumeričkom kriterijumu ili po mesecima u godini.

Sintaksa komande sort je:

```
$ sort [OPTION]... [FILE]...
```

Komanda sort vrši "sortiranje" linija datoteka navedenih parametrom FILE i ispisuje konkatenaciju sortiranog teksta na standardni izlaz. Rezultat izvršenja ove komande se može redirekcijom preusmeriti u datoteku. U podrazumevanom stanju, sort vrši alfanumeričko uređivanje u rastućem redosledu, po prvoj koloni.

Opcije kojima se može modifikovati kriterijum uređivanja su:

-b	sort ignoriše blanko karaktere na početku linije
-f	sort ne pravi razliku između malih i velikih slova
-M	sortiranje se vrši po mesecima u godini (nepoznat < JAN < FEB < ... < DEC)
-r	inverzno sortiranje
+N	definiše kolonu tekstualne datoteke po kojoj se vrši sortiranje (+1 znači uređivanje na osnovu druge kolone, +2 na osnovu treće i tako redom). Opcija je korisna za uređivanje tekstualnih tabela.

U ostale značajnije opcije spadaju:

-oFILE	uređen tekst se upisuje u datoteku FILE umesto na ekran (slično redirekciji izlaza)
-tSEP	definiše se karakter koji predstavlja separator kolona, odnosno karakter kojim su kolone odvojene (podrazumevani karakter je razmaknica)

Sledeći primeri ilustruju upotrebu komande sort za:

- alfanumeričko uređivanje teksta

```
$ cat fullnames
Johnny  Knoxville
Steve   O
Chris   Pontius
$ sort fullnames
Chris   Pontius
Johnny  Knoxville
Steve   O
$ sort -r +1 fullnames
Chris   Pontius
Steve   O
Johnny  Knoxville
```

- uređivanje tabele po mesecima u godini, sa definisanjem separatora kolona (:)

```
$ cat birthdays2
Jordan:FEB 13 1979
Smith:JAN 11 1980
Cooling:SEP 12 1950
Wotson:DEC 09 1965
Forsth:MAR 22 1977
$ sort -M +1 -t: birthdays2
Smith:JAN 11 1980
Jordan:FEB 13 1979
Forsth:MAR 22 1977
Cooling:SEP 12 1950
Wotson:DEC 09 1965
```

Priprema tekstualnih datoteka za štampu (pr)

Komanda pr priprema tekstualne datoteke za štampu. Priprema obuhvata: podelu datoteke na stranice, numerisanje stranica i navođenje datuma i imena datoteke u zaglavlju.

Sledeći primer ilustruje funkcionisanje programa pr. Pretpostavljamo da hoćemo da odštampamo datoteku /etc/passwd radi analize. Zadajemo sledeću komandu:

```
$ pr /etc/passwd > /tmp/fpasswd
```

Datoteka /tmp/fpasswd je pripremljena za štampu i može se odštampati komandom lpr ili preusmeravanjem izlaza na štampač (cat /tmp/fpasswd > /dev/lp0). U nastavku teksta dat je isečak sadržaja datoteke /tmp/fpasswd:

```
2004-05-04 17:38              /etc/passwd              Page    1

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
...
...
jsmith:x:1029:1029::/home/jsmith:/bin/bash
jknoxville:x:1030:1030::/home/jknoxville:/bin/bash
cpontius:x:1031:1031::/home/cpontius:/bin/bash



2004-05-04 17:38              /etc/passwd              Page    2

steveo:x:1032:1032::/home/steveo:/bin/bash
rdunn:x:1033:1033::/home/rdunn:/bin/bash
...
...
```

Podela tekstualnih datoteka (split)

Svaka tekstualna datoteka se komandom split može podeliti na nekoliko manjih datoteka. Opšta sintaksa komande split je:

```
$ split condition inputfile prefix
```

gde su:

condition	uslov podele datoteka
inputfile	datoteka koja se deli
prefix	prefiks na osnovu koga se formiraju nove datoteke.

To znači da program split deli datoteku inputfile na osnovu nekog uslova (broj bajtova ili linija po jednom delu), pri čemu delove formira sa sledećim imenima: prefixaa, prefixab, prefixac, ..., prefixba, prefixbb, ...

Ukoliko je potrebno oformiti delove veličine n bajtova koristi se sledeća sintaksa:

```
$ split -bn inputfile prefix
```

Ukoliko je potrebno oformiti delove koji će u sebi sadržati n linija, koristi se sledeća sintaksa:

```
$ split -ln inputfile prefix
```

Na primer, pretpostavimo da imamo datoteku myfile, čiji je sadržaj:

```
Hi !
My name is Wile E. Coyote.
I am a genius.
A pure genius.
```

Zadajemo komandu split u sledećem obliku:

```
$ split -l2 myfile mypiece
```

Nakon toga će se formirati datoteke mypieceaa i mypieceab čiji su sadržaji:

```
$ cat mypieceaa
Hi !
My name is Wile E. Coyote.
$ cat mypieceab
I am a genius.
A pure genius.
```

Uklanjanje duplikata linija iz datoteke (uniq)

Komanda uniq iz tekstualne datoteke uklanja sve duplikate linija, odnosno izbacuje sve konsekventne identične linije, osim prve u nizu. Rezultat izvršenja prikazuje se na ekranu a po potrebi se može preusmeriti u datoteku.

```
$ cat duplicates
First Line
Second Line
Second Line
Third Line
$ uniq duplicates > noduplicates
$ cat noduplicates
```

```
First Line
Second Line
Third Line
```

Traženje teksta u datoteci

Jedan od mogućih načina kojim se može utvrditi da li određeni korisnik ima nalog na sistemu ili ne jeste pretraživanje datoteke /etc/passwd. Ukoliko se datoteka proverava programom less, a na sistemu postoji veliki broj korisničkih naloga, ova procedura može potrajati. Jednostavniji i brži način kojim se to može izvesti je korišćenje komande grep. Komanda grep traži navedeni uzorak teksta u svim linijama datoteke i na ekranu prikazuje one linije u kojima je uzorak nađen.

Sam naziv grep je skraćenica od izraza Global Regular Expression Print: grep izvršava globalnu pretragu datoteke, traži regularni izraz (Regular Expression) i štampa na ekranu linije u kojima je taj izraz pronađen.

Za ispravno korišćenje programa grep neophodno je elementarno poznavanje regularnih izraza, koji se, između ostalog, koriste i za pronalaženje teksta u datoteci. Regularne izraze koriste programi grep i egrep; treći program iz tog paketa, fgrep radi samo sa nizovima karaktera.

Regularni izrazi (regular expressions)

Regularni izraz je niz običnih i specijalnih karaktera.

U obične karaktere ubrajaju se:

- sva slova i brojevi
- tilda (~), obrnuti navodnik (`), znak uzvika (!), znak "at" (@), mrežica (#), underscore karakter (_), minus (-), znak jednakosti (=), dve tačke (:), tačka-zarez (;), zarez (,), slash karakter (/).

U specijalne karaktere ubrajaju se:

- backslash karakter (\)
- tačka (.)
- zvezdica (*)
- srednje zagrade ([) i (])
- carret karakter (^)
- znak za dolar ($).

U najjednostavnijem obliku regularni izraz uključuje samo obične karaktere i tada se naziva string (niz karaktera). Program grep pronalazi string u linijama teksta čak i kad je deo neke druge reči. Na primer, reč dent se može naći u rečima dental, president, residential i sličnim, tako da grep na ekranu štampa sve linije koje sadrže te reči.

Korišćenjem komplikovanijih oblika regularnih izraza komanda grep može na ekranu prikazati sve linije koje na primer sadrže:

- reč Linux,
- reči Linux ili LINUX (regularni izraz se formira kao karakter U iza koga slede karakteri i ili I, zatim n ili N, u ili U i na kraju x ili X),
- bilo koje četiri cifre.

Regularni izrazi se dele na osnovne (limited) i proširene (extended, full). Osnovni regularni izrazi su podskup skupa proširenih regularnih izraza.

Jednostavan oblik komande grep

U najjednostavnijem obliku, sintaksa komande grep je:

```
$ grep RE filename
```

gde je RE regularni izraz koji se traži u datoteci filename.

U nastavku teksta se radi ilustracije rada programa grep koristi sledeća datoteka:

```
$ cat myfile
A regular expression is a sequence of characters taken
from the set of uppercase and lowercase letters, digits,
punctuation marks, etc., plus a set of special regular
expression operators. Some of these operators may remind
you of file name matching, but be forewarned: in general,
regular expression operators are different from the
shell metacharacters we discussed in Chapter 1.
The simplest form of a regular expression is one that
includes only letters. For example, they would match only
the three-letter sequence t, h, e. This pattern is found
in the following words: the, therefore, bother. In other
words, wherever the regular expression pattern is found
— even if it is surrounded by other characters — it will
be matched.
```

Primer ilustruje traženje stringa "only" u datoteci myfile. U ovom slučaju, regularni izraz je jednostavan niz karaktera.

```
$ grep only myfile
includes only letters. For example, the would match only
```

Reč only se pojavljuje samo u jednoj liniji, i ta linija je prikazana na ekranu kao rezultat izvršenja komande grep.

Specijalni karakteri

Određeni karakteri imaju specijalno značenje bez obzira na mesto pojavljivanja u regularnim izrazima, dok drugi dobijaju specijalno značenje zavisno od mesta na kome se nalaze i drugih karaktera koji se nalaze pre ili posle njih.

Karakteri tačka (.), zvezdica, uglaste zagrade i backslash (\) smatraju se specijalnim osim kada se nalaze između para uglastih zagrada.

Karakter carret (^) je specijalan ukoliko je prvi karakter u regularnom izrazu ili prvi karakter nakon otvaranja leve uglaste zagrade.

Znak za dolar ($) je specijalan ukoliko je poslednji karakter u regularnom izrazu .

Karakteri za razdvajanje (par slash karaktera //) su specijalni jer razdvajaju regularni izraz.

Specijalni karakter kom prethodi backslash smatra se običnim karakerom (ecaping).

Zamena jednog karaktera

Tačka (.) u regularnom izrazu govori komandi grep da taj karakter interpretira kao bilo koji drugi, osim prelaska u novu liniju.

Na primer:

```
$ grep 'w.r' myfile
from the set of uppercase and lowercase letters, digits,
you of file name matching, but be forewarned: in general,
in the following words: the, therefore, bother. In other
words, wherever the regular expression pattern is found
```

Napomena: regularni izraz w.r mora se staviti između apostrofa (jednostrukih navodnika - single quotes). Ukoliko se to ne učini, shell će interpretirati izvesne specijalne karaktere kao shell specijalne karaktere, tako da se komandi grep u nekim slučajevima mogu proslediti nekorektni regularni izrazi.

Takođe, može se definisati opseg, odnosno skup karaktera koje grep interpretira kao karakter koji menjamo. Na primer, uzorak [abc12] menja bilo koji od karaktera a, b, c, 1 ili 2.

```
$ grep 'w[fhmkz]' myfile
words, wherever the regular expression pattern is found
```

Ukoliko je prvi karakter u uglastoj zagradi carret, to znači da grep može interpretirati tačku kao bilo koji karakter koji se ne nalazi u navedenom skupu:

```
$ grep 'w[^fhmkz]' myfile
from the set of uppercase and lowercase letters, digits,
you of file name matching, but be forewarned: in general,
shell metacharacters we discussed in Chapter 1.
includes only letters. For example, the would match only
in the following words: the, therefore, bother. In other
words, wherever the regular expression pattern is found
— even if it is surrounded by other characters — it will
```

Opseg se formuliše pomoću znaka minus, koji se nalazi između dva karaktera u okviru uglastih zagrada. Pri tome, ne može se zadati inverzan opseg zamene, ali se mogu navesti više opsega u okviru istih zagrada. Na primer, uzorak [a-zA-Z] odgovara svim velikim i malim slovima.

```
$ grep 'w[a-f]' myfile
```

```
from the set of uppercase and lowercase letters, digits,
you of file name matching, but be forewarned: in general,
shell metacharacters we discussed in Chapter 1.
```

Ponavljanje karaktera

Regularnim izrazom oblika karakter\{nmin,nmax\} može se specificirati koliko puta karakter treba da se ponovi. Na primer, uzorak X\{2,5\} znači najmanje dva i najviše pet uzastopnih pojavljivanja karaktera X (odnosno XX, XXX, XXXX ili XXXXX). Uzorak X\{2,\} znači najmanje dva pojavljivanja karaktera X (odnosno XX, XXX, ..., XXX...XXX). Uzorak X\{4\} znači tačno četiri uzastopna pojavljivanja karaktera X (odnosno XXXX). Sledi primer ovog oblika izraza:

```
$ grep 'p\{2,4\}' myfile
from the set of uppercase and lowercase letters, digits,
```

Ponavljanje regularnih izraza

Karakter zvezdica (*) označava nula, jedno ili više ponavljanja nekog regularnog izraza. Na primer, uzorak X* se interpretira: prazan string, X, XX, XXX, XXX...XXX. Ukoliko je potrebno jedno ili više ponavljanja, regularni izraz se formira kao XX*, što se interpretira kao X, XX, XXX, XXX...XXX.

Na primer,

```
$ grep 'p*' myfile
```

prikazuje celu datoteku myfile, dok:

```
$ grep 'pp*' REfile
```

prikazuje sve linije koje sadrže jedno ili više uzastopnih pojavljivanja slova p.

Upoređivanje početka i kraja linije teksta

Ukoliko se znak carret (^) navede kao prvi karakter regularnog izraza znači uzorak se upoređuje sa početkom linije. Na primer, ^[Tt]he odgovara svim linijama datoteke koje počinju stringom the ili The:

```
$ grep '^[Tt]he' myfile
The simplest form of a regular expression is one that
the three-letter sequence t, h, e. This pattern is found
```

Ukoliko se znak za dolar ($) navede kao poslednji karakter regularnog izraza, uzorak se upoređuje sa krajem linije. Na primer, [Tt]he$ odgovara svim linijama koje se završavaju nizovima karaktera The ili the:

```
$ grep '[Tt]he$' myfile
regular expression operators are different from the
```

Primer složenog regularnog izraza

Standardni američki format datuma je niz karaktera formiran na sledeći način: ime meseca s velikim početnim slovom (najmanje tri karaktera - maj, najviše devet karaktera - septembar), razmak, jedna ili dve cifre za dan, zarez, razmak, četiri cifre za godinu. Regularan izraz se formira na sledeći način

```
[A-Z][a-z]\{2,8\} [0-9]\{1,2\}, [0-9]\{4\}
```

i odgovara nizovima karaktera tipa: August 16, 2004.

Komanda grep

Najčešće korišćena komanda iz grep paketa je grep. Kompletna sintaksa komande grep je:

```
$ grep [options] LRE [file(s)]
```

gde je LRE osnovni regularni izraz (Limited Regular Expression), a file(s) skup datoteka u kojima se uzorak traži. Od značajnijih opcija pominjemo sledeće:

-c grep prikazuje broj linija u kojima je nađen uzorak,

-h grep ne prikazuje imena datoteka ukoliko se pretraživanje vrši u nekoliko datoteka,

-i grep ignoriše razlike između velikih i malih slova,

-n pre svake linije grep će prikazati i redni broj linije u datoteci.

Komanda egrep

Komanda egrep dozvoljava upotrebu proširenih regularnih izraza. Sintaksa komande je slična sintaksi komande grep:

```
$ egrep [options] FRE [files]
```

gde je FRE prošireni regularni izraz (full, limited regular expression). Ovi regularni izrazi predstavljaju proširenje skupa osnovnih regularnih izraza sledećim operatorima:

RE+ traži se jedno ili više pojavljivanja regularnog izraza RE,

RE? traži se jedno ili ni jedno pojavljivanje regularnog izraza RE,

RE1 | RE2 traži se regularan izraz RE1 ili regularan izraz RE2. Znak pipe (|) se ponaša kao logički ILI operator,

(RE) grupiše se veći broj regularnih izraza.

Sledeći primeri ilustruju

- traženje linija koje sadrže najmanje dva uzastopna velika slova u datotekama file1 i file2:

```
$ egrep '[A-Z][A-Z]+' file1 file2
```

- traženje linija koje sadrže niz UNIX ili Linux u datotekama file1 i file2:

```
$ egrep 'UNIX|Linux' file1 file2
```

- traženje svih linija koje sadrže reči cat ili cut u datoteci myfile

```
$ egrep 'c(a|u)t' myfile
```

Komanda fgrep

Komanda fgrep na ekranu prikazuje sve linije datoteke ili grupe datoteka koje u sebi sadrže zadati niz karaktera. Za razliku od komandi grep i egrep, fgrep interpretira svaki karakter u nizu kao običan karakter. Fgrep ne podržava rad sa regularnim izrazima, i u nekim slučajevima je pogodniji za rad (na primer, ukoliko se traži niz oblika "C:\A*.txt").

Sintaksa komande fgrep je:

```
$ fgrep [options] string [files]
```

Komanda koristi iste opcije kao i komande grep i egrep.

Editori teksta vi, joe i jed

vi editor

Vi editor je deo svakog UNIX sistema, počev od prvih verzija, koje su se pojavile ranih sedamdesetih godina. Iako je vi ekranski editor, jako je neugodan za korišćenje. Koristi se isključivo za izmenu sadržaja tekstualnih datoteka i ne podržava formatiranje (na primer, masna slova i kurziv). Vi editor koristi ekranski prikaz datoteke (full-screen), ali ne dozvoljava upotrebu miša za pozicioniranje kursora na odgovarajući karakter - navigacija i sve modifikacije vrše se isključivo pomoću tastature. Izmene sadržaja datoteka se mogu snimiti na disk ili odbaciti.

Vi je ponekad jedini ekranski editor dostupan za modifikaciju složenijih sistemskih datoteka (na primer, shell programa, datoteke /etc/passwd) i kao takav ulazi u skup neophodnih znanja ozbiljnog sistem administratora. Veština vladanja vi editorom je najčešće neophodna ukoliko na sistemu nema grafičkog okruženja, i ukoliko se administracija mrežnog servera vrši preko Telnet ili ssh sesije.

Režimi rada vi editora

Postoje tri osnovna režima rada vi editora: komandni, tekstualni i režim poslednje linije (linijski). Razumevanje ovih režima je ključ uspeha u radu sa vi editorom. Sledeća tabela ilustruje komande koje se koriste za prelazak iz jednog režima u drugi:

- Komandni -> tekstualni: i (input), o (open new line), a (append to existing line)
- Tekstualni -> komandni: Esc
- Komandni -> režim zadnje linije: dvotačka (:)

- Režim zadnje linije -> komandni režim: Enter

U nastavku teksta dat je opis režima rada vi editora.

Komandni mod je režim u kome se korisnik nalazi kada pokrene vi. Korisnik može da unese komande za pozicioniranje kursora i komande za modifikaciju sadržaja datoteke. Iz ovog režima korisnik može da pređe u tekstualni ili linijski režim. Da bi prešao iz tekstualnog u linijski režim korisnik prvo mora preći u komandni režim. Taster Esc je uvek prečica za prelazak u komandni režim, tako da korisnik uvek može da ga iskoristi ukoliko nije siguran u kom se režimu nalazi. Dok se nalazi u komandnom režimu, korisnik može da vrši navigaciju po tekstu i zadaje određene komande.

Da bi unosio tekst korisnik mora preći u tekstualni režim komandama I (input), o (open new line) ili a (apend). Naravno, u ovom režimu uneti tekst neće biti protumačen kao komanda. Nakon završenog unosa, korisnik se može vratiti u komandni režim tasterom Esc.

U režim zadnje linije korisnik može preći zadavanjem karaktera dvotačka (:) u komandnom režimu. Tada se kursor pomera na dno ekrana. U ovom režimu korisnik može da snimi datoteku, snimi datoteku i napusti vi, napusti vi bez snimanja datoteke, traži i zameni određene nizove karaktera i konfiguriše vi.

Otvaranje datoteka vi editorom

Vi editor je je UNIX aplikacija koja se pokreće iz komandne linije. Ukoliko korisnik zada komandu vi bez argumenata, ili kao argument navede ime nepostojeće datoteke, vi će početi rad na novoj datoteci. Ukoliko se kao argument navede ime postojeće datoteke, vi će datoteku učitati, nakon čega korisnik može da vrši izmene. Sadržaj datoteke na disku se ažurira sadržajem modifikovanim vi editorom tek kada korisnik zada komandu za snimanje datoteke.

Sintaksa komande vi je:

```
$ vi [option(s)] [filename]
```

Najšešća sekvenca koraka u radu sa vi editorom je: otvaranje nove ili postojeće datoteke, unošenje novog ili modifikacija postojećeg teksta, snimanje izmena na disk i napuštanje vi editora.

U mnogim Linux distribucijama, vi editor je zamenjen editorom vim (vi improved), koji se pokreće u slučaju da korisnik zada komandu vi.

U nastavku teksta prikazan je izgled ekrana prilikom korišćenja vi editora:

```
This ia a new file,
which is beeing created with vi.
Such a horrible experience, this vi.
Shall use joe tommorow.
~
~
~
~
~
```

```
myfile: unmodified: line 1
```

Interaktivne komande u vi editoru

Komande za prelazak iz komandnog u tekstualni režim rada:

a	unesi tekst posle kursora
A	unesi tekst na kraju linije
i	unesi tekst pre kursora
o	otvori novu liniju pre kursora

Komande za snimanje datoteka i napuštanje vi editora dostupne su iz režima poslednje linije:

:w	snimanje datoteke na disk pod tekućim imenom
:w filename	snimanje datoteke na disk pod imenom filename
:wq	snimanje datoteke i napuštanje vi editora
:q!	napuštanje vi editora bez snimanja izmena

Komande za navigaciju u komandnom režimu (kursorski tasteri - strelice se mogu koristiti za navigaciju ako konkretni terminal to podržava):

h (left arrow)	pomera kursor na prethodni karakter
l (right arrow, space)	pomera kursor na sledeći karakter
w	pomera kursor na sledeću reč
b	pomera kursor na prethodnu reč
k (up arrow)	pomera kursor na prethodnu liniju
j (down arrow)	pomera kursor na sledeću liniju
$	pomera kursor na kraj linije
0	pomera kursor na pocetak linije
Return	pomera kursor na početak sledeće linije

Komande za modifikaciju teksta dostupne su iz komandnog režima i režima zadnje linije:

x	briše karakter koji je obeležen kusorom
dw	brise reč desno od kursora
dd	briše liniju u kojoj se nalazi kursor
yy	kopira liniju u kojoj se nalazi kursor na clipboard (yank)
p	kopira liniju sa clipboarda ispod tekuće linije (paste)
P	kopira liniju sa clipboarda iznad tekuće linije (paste)
u	poništava prethodnu akciju (undo)

Komande za pretraživanje datoteke zadaju se u komandnom režimu ili u režimu zadnje linije. Neke verzije vi editora dozvoljavaju pretraživanje teksta na osnovu regularnih izraza (slično komandi grep).

G	pozicioniranje na zadnju liniju teksta
:n	pozicioniranje na n-tu liniju teksta
/string	traži niz karaktera string od tekuće pozicije do kraja datoteke
?string	traži niz karaktera string od tekuće pozicije do početka datoteke
n	traži sledeće pojavljivanje niza karaktera string
N	traži prethodno pojavljivanje niza karaktera string

Komande za konfigurisanje vi editora zadaju se iz režima zadnje linije, a počinju sa set.

Alternativni editori teksta

Vi editor je deo svakog UNIX sistema i njegovo poznavanje je značajno za svakog ozbiljnog sistem administratora. Komande za modifikaciju datoteka kao što su /etc/passwd (vipw) i /etc/group (vigr) koriste vi za izmenu sadržaja ovih datoteka). Međutim, korisnik koji planira da postane sistem administrator treba da se upozna i sa drugim editorima teksta kao što su joe, jed, emacs i pico, koji su lakši i prijatniji za korišćenje

Emacs (editor macros) je lakši za upotrebu jer ne koristi režime rada (mode-less editor). U emacs editoru, korisnik zadaje komande kombinovanjem Ctrl tastera sa nekim slovom: na primer, <Ctrl-d> služi za brisanje karaktera, a <Ctrl-s> za snimanje datoteke. Pico editor dolazi u paketu sa Pine mail programom. Kao i Pine, pico je lak za upotrebu (program je vođen menijem).

Kreiranje malih tekstualnih datoteka komandom cat

Kao što je već rečeno, komanda cat je dobar alat za pregledanje sadržaja malih datoteka. Ova komanda se takođe može koristiti i za kreiranje manjih tekstualnih datoteka na sledeći način: najpre se zada komanda u sledećem obliku:

```
$ cat > filename
```

gde je filename ime datoteke koju želimo da kreiramo, koje po potrebi može sadržati i apsolutnu ili relativnu putanju. Nakon toga se unese nekoliko linija teksta (u novu liniju se prelazi pomoću tastera Enter) i pritisne <Ctrl-d>. Prilikom unošenja teksta povratak na prethodne linije nije moguć. Na ovaj način će se kreirati datoteka koja će sadržati unešeni tekst.

joe (Joe's Own Editor)

Za razliku od vi editora, joe je krajnje prijatan za rad. Editor joe se pokreće sledećom komandom:

```
$ joe [filename]
```

Ukoliko korisnik zada komandu joe bez argumenata, ili kao argument navede ime nepostojeće datoteke, joe će početi rad na novoj datoteci. Ukoliko se kao argument navede ime postojeće datoteke, joe će datoteku učitati, nakon čega korisnik može da vrši izmene. Sadržaj datoteke na disku ažurira se sadržajem modifikovanim editorom joe tek kada korisnik zada komandu za snimanje datoteke.

Ovaj editor ne koristi režime: tekst se jednostavno unosi (korisnik se nalazi u insert režimu), a komande se zadaju korišćenjem kombinacije tastera <Ctrl-slovo>. U nastavku teksta prikazan je primer korišćenja editora joe (kraći isečak ekrana):

```
IW   myfile (Modified)    Row 3   Col 1   3:45  Ctrl-K H for help
This file is created using joe.
After such a horrible experience with vi, this is a summer breeze.
```

U svakom trenutku korisnik može dobiti pomoć kombinacijom tastera <Ctrl-K>, a zatim pritiskom na taster H. Ekran pomoći je prikazan u nastavku teksta:

```
   Help Screen    turn off with ^KH    more help with ESC . (^[.)
 CURSOR           GO TO            BLOCK      DELETE   MISC         EXIT
 ^B left ^F right ^U  prev. screen ^KB begin  ^D char. ^KJ reformat ^KX save
 ^P up   ^N down  ^V  next screen  ^KK end    ^Y line  ^T  options  ^C  abort
 ^Z previous word ^A  beg. of line ^KM move   ^W >word ^R  refresh  ^KZ shell
 ^X next word     ^E  end of line  ^KC copy   ^O word< ^@  insert   FILE
 SEARCH           ^KU top of file  ^KW file   ^J >line SPELL        ^KE edit
 ^KF find text    ^KV end of file  ^KY delete ^_ undo  ^[N word     ^KR insert
 ^L  find next    ^KL to line No.  ^K/ filter ^^ redo  ^[L file     ^KD save
```

Za navigaciju po tekstu mogu se koristiti kursorski tasteri (kao i PgUp i PgDn) ili sledeće komande, koje pomeraju kursor:

<Ctrl-B>	jedan karakter ulevo
<Ctrl-F>	jedan karakter udesno
<Ctrl-P>	jedan karakter na gore
<Ctrl-N>	jedan karakter na dole
<Ctrl-U>	na prethodni ekran teksta
<Ctrl-V>	na sledeći ekran teksta
<Ctrl-A>	na početak linije
<Ctrl-E>	na kraj linije
<Ctrl-K>, zatim U	na početak datoteke
<Ctrl-K>, zatim V	na kraj datoteke
<Ctrl-K>, zatim V	na određenu liniju, čiji se broj nakon toga zadaje

Za pretraživanje teksta koriste se komande <Ctrl-K>F (nakon čega se zadaje tekst koji se traži) i <Ctrl-L> koja traži sledeće pojavljivanje teksta. Nakon zadavanja ove komande korisnik može zameniti pronađeni tekst novim.

Za rad sa blokovima (clipboard) koriste se sledeće komande:

<Ctrl-K>, zatim B	markiranje početka bloka
<Ctrl-K>, zatim K	markiranje kraja bloka (blok je obeležen ako se markiraju i početak i kraj)
<Ctrl-K>, zatim M	pomeranje bloka na tekuću poziciju
<Ctrl-K>, zatim C	kopiranje bloka na tekuću poziciju
<Ctrl-K>, zatim F	snimanje bloka u datoteku čije se ime naknadno navodi
<Ctrl-K>, zatim Y	brisanje bloka
<Ctrl-K>, zatim /	izvršavanje neke druge komande joe editora na obeleženom bloku, a ne na celoj datoteci

Za brisanje karaktera, reči i linija mogu se koristiti sledeće komande, koje redom brišu:

<Ctrl-D>	tekući karakter
<Ctrl-Y>	tekuću liniju
<Ctrl-W>	niz karaktera do kraja tekuće reči, počev od tekućeg karaktera
<Ctrl-O>	niz karaktera od početka tekuće reči do tekućeg karaktera
<Ctrl-J>	niz karaktera do kraja tekuće linije, počev od tekućeg karaktera
<Ctrl-->	poništava izvršenje prethodne operacije

Za konfigurisanje joe editora na raspolaganju je komanda <Ctrl-T>. Nakon ove komande na dnu ekrana se prikazuje scroll traka sa opcijama koje korisnik može izmeniti čime vrši prilagođavanje editora svojim potrebama. Značajnije opcije su overtype (ukoliko se postavi na off, korisnik dodaje novi tekst; ukoliko se podesi na ON, korisnik menja postojeći tekst), širina TAB karaktera, leva i desna margina.

Za snimanje datoteke i napuštanje editora joe koriste se sledeće komande:

<Ctrl-K>, zatim X	snimanje datoteke na disk i napuštanje editora joe
<Ctrl-C>	napuštanje editora bez snimanja datoteke
<Ctrl-K>, zatim Z	privremeni izlazak u komandni interpreter (joe se kao proces stavlja u pozadinu, a u prvi plan se može vratiti komandom fg)

jed

Editor jed je mode-less editor, prijatan za rad kao i joe. Kao novina u jed editoru se uvode i padajući meniji koji se mogu koristiti za zadavanje komandi.

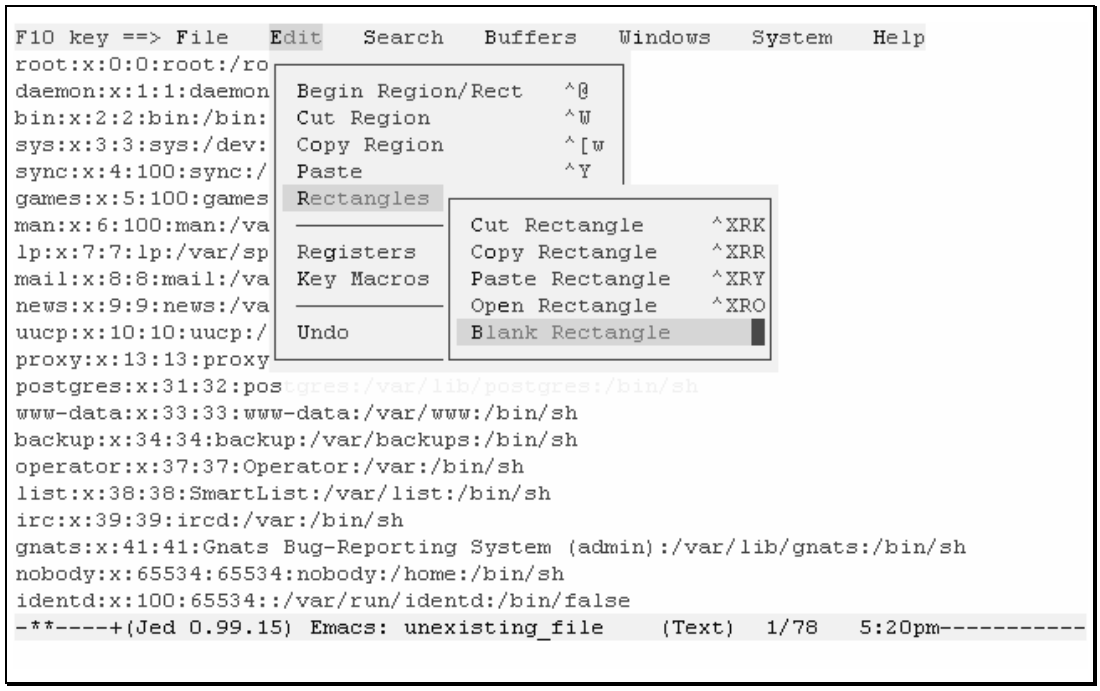
Jed se pokreće kao i većina ostalih editora:

```
$ jed [filename]
```

Isto što važi za vi i joe važi i u ovom slučaju: ukoliko korisnik zada komandu jed bez argumenata, ili kao argument navede ime nepostojeće datoteke, jed će početi rad na novoj datoteci. U suprotnom, jed će datoteku učitati, nakon čega korisnik može da vrši izmene.

Sadržaj datoteke na disku ažurira se sadržajem modifikovanim editorom jed tek kada korisnik zada komandu za snimanje datoteke.

Korisnik u jed editoru jednostavno unosti tekst i putem padajućih menija koristi komande za rad sa tekstom. Padajući meni sa opcijama dobija se pritiskom na taster <F10>. Opcije padajućih menija su intuitivne i kao takve nisu objašnjene. Na slici 5.4 prikazan je radni ekran editora jed.



Slika 5.4 *Editor jed*

Midnight Commander

Twin-panel interfejs za rad sa datotekama. Funkcije i mogućnosti alata Midnight Commander.

Kao što je već rečeno, grafičko radno okruženje se iz razloga rasterećenja sistema ne instalira na mrežnim serverima. Radi lakšeg rada sa datotekama na takvim sistemima se može instalirati programski paket Midnight Commander.

Twin-panel interfejs

Za razliku od klasičnih file managera, Midnight Commander koristi twin-panel interfejs. Programi koji koriste twin-panel interfejs nisu novost - osnovna ideja potiče iz programa Norton Commander (čiji je autor Peter Norton) koji je sredinom osamdesetih godina bio uz PC Tools najčešće korišćen program za rad sa datotekama. Nakon toga, napisan je

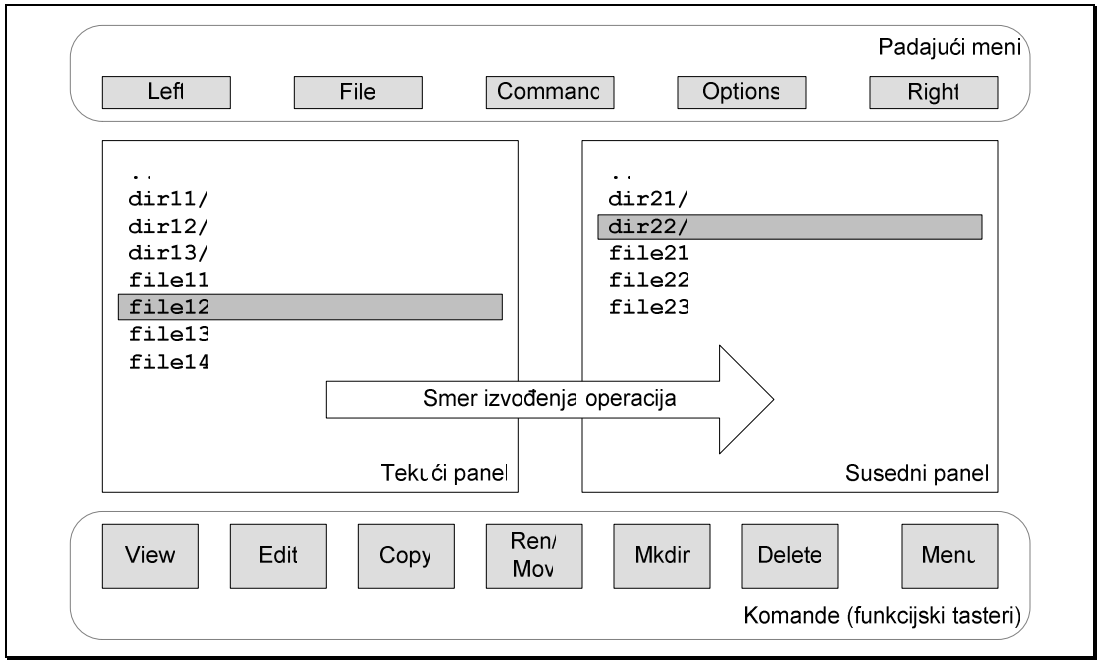
veliki broj komercijalnih i besplatnih programa za sve popularne operativne sisteme koji koriste ovaj interfejs. Najpoznatiji su: DOS Navigator i Total Commander (DOS/Windows), Midnight Commander (Linux, shell) i Krusader (Linux, KDE).

Ideja twin-panel interfejsa je sledeća: ekran, odnosno prozor (ukoliko se radi u grafičkom okruženju) deli se na dva dela (levi i desni panel), koji predstavljaju po jednu poziciju u aktivnom stablu (UNIX/Linux) ili na nekom logičkom disku (Windows). U panelu je prikazan sadržaj tog direktorijuma, pri čemu u sklop sadržaja ulazi i roditeljski direktorijum (obeležen sa ../). Korisnik se pozicionira na levi ili desni panel (tekući panel, u nastavku teksta) i obavlja određene akcije. Akcije se izvršavaju nad tekućom datotekom ili direktorijumom (odnosno datotekom nad kojom je pozicioniran marker) ili nad grupom odabranih datoteka i direktorijuma iz tekućeg panela. Na taj način korisnik jednostavno i pregledno može:

- pregledati sadržaj datoteka,
- modifikovati sadržaj datoteka,
- kopirati datoteku, grupu datoteka ili deo stabla iz tekućeg panela u susedni,
- promeniti ime datoteke ili direktorijuma, odnosno pomeriti datoteku, grupu datoteka ili deo stabla iz tekućeg panela u susedni,
- kreirati direktorijume,
- obrisati datoteke ili delove stabla.

Komande koje obavljaju ove akcije dostupne su preko funkcijskih tastera, koji su opisani na dnu ekrana, ili stavki padajućeg menija, koji se nalazi na vrhu ekrana. Padajući meni je takođe dostupan preko funkcijskih tastera. Norton Commander je uveo defacto standard za uparivanje operacija i funkcijskih tastera (ovaj standard koriste svi gore pomenuti programi, uključujući i Midnight Commander).

Na slici 5.5 prikazano je twin-panel okruženje:



Slika 5.5 *Twin-panel okruženje*

Rad u programu Midnight Commander

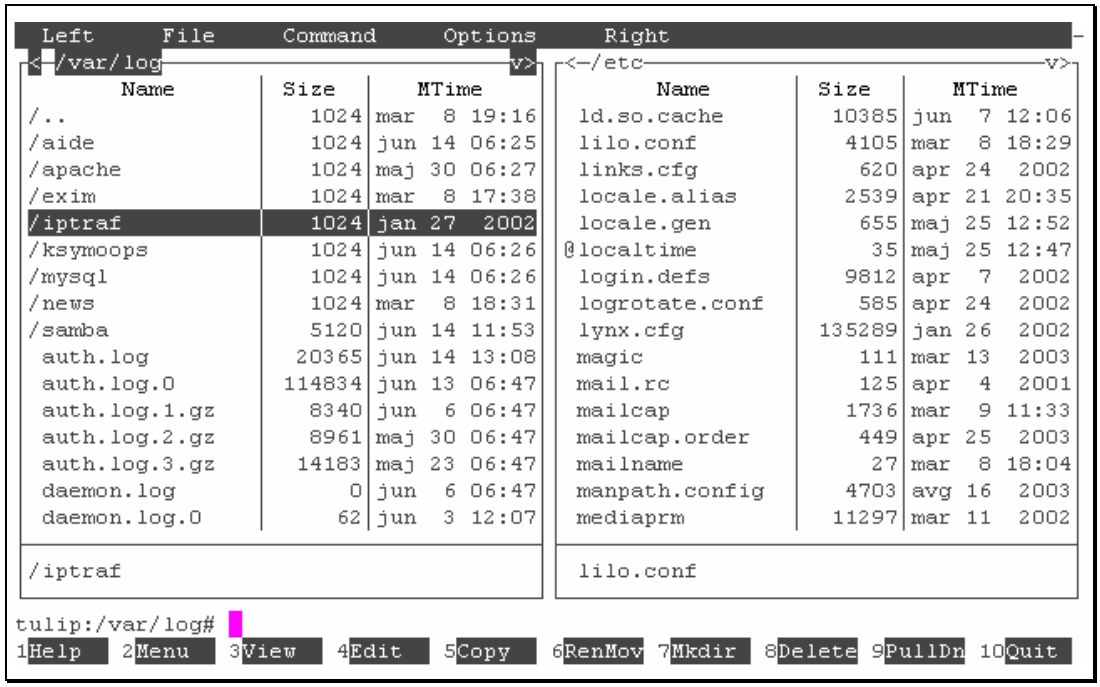
Midnight Commander je prvi poznatiji file manager ovog tipa koji se pojavio na Linux sistemu. Program se pokreće iz komandne linije jednostavnom komandom:

```
$ mc
```

Jako je sličan sa Norton Commanderom, odnosno DOS Navigatorom. Sličnost se između ostalog uočava u dodeljivanju operacija funkcijskim tasterima:

F1 (Help)	dobijanje pomoći
F2 (Menu)	korisnički definisan meni
F3 (View)	pregledanje sadržaja datoteka
F4 (Edit)	editor datoteka, ugrađen u Midnight Commander
F5 (Copy)	kopiranje datoteka ili delova stabla
F6 (RenMov)	promena imena ili pomeranje datoteka ili delova stabla
F7 (Mkdir)	kreiranje direktorijuma
F8 (Delete)	brisanje datoteka
F9 (PullDn)	padajući meni
F10 (Quit)	napuštanje programa

Na slici 5.6 prikazan je radni ekran programa Midnight Commander:



Slika 5.6 *Midnight Commander*

U okviru jednog panela datoteke i direktorijumi se mogu obeležiti pritiskom na taster Ins (ukoliko je datoteka obeležena, Ins će skinuti marker), a u poddirektorijume se prelazi pomoću tastera Enter. Dodatno, program dozvoljava korisniku da pri radu koristi miša, što u nekim slučajevima ubrzava rad.

Padajući meni nudi dosta opcija. Opcije menija Left i Right su identične i odnosne se na levi i desni panel respektivno:

Listing mode	dozvoljava korisniku da izabere način prikazivanja objekata u tekućem direktorijumu (da li želi da vidi samo ime, ime i sve atribute ili ime i određeni set atributa).
Quick View	levi panel se koristi kao Quick View, odnosno koristi se za brzi pregled tekuće datoteke iz desnog panela.
Info	levi panel se koristi za prikazivanje informacija o tekućoj datoteci iz desnog panela. Ove informacije obuhvataju: fizičku lokaciju objekta, pristupna prava i vlasničke odnose, broj linkova, sva vremena, ime uređaja na kome se nalazi i odgovarajući mount-point, tip sistema datoteka, slobodan prostor i broj slobodnih i-node struktura u sistemu datoteka na kome se taj objekat nalazi.
Tree	levi panel prikazuje direktorijumsko stablo. Tekuća pozicija u desnom panelu određuje se izborom direktorijuma u stablu.

Sort Order	određuje se kriterijum sortiranja datoteka u panelu. Sortiranje se može obaviti u rastućem ili opadajućem redosledu po imenu, ekstenziji, i-node broju, vremenima i veličini datoteke.
Filter	postavljanje filtra za datoteke na osnovu imena i ekstenzije.

Opcije menija file pokrivaju osnovne operacije koje korisnik može da izvrši nad datotekama. Tu spadaju operacije dodeljene funkcijskim tasterima kao i:

Chmod	promena pristupnih prava tekućeg objekta (uključuje mogućnost dodele SUID, SGID i sticky bita),
Chown	promena vlasničkih odnosa tekućeg objekta,
Link	kreiranje hard linka datoteke,
SymLink	kreiranje simboličkog linka objekta,
Edit SymLink	preusmeravanje simboličkog linka (promena pokazivača),

U meniju Command nalaze se opcije koje se ne odnose direktno na rad sa datotekama u panelima:

Directory Tree	prikazuje aktivno UNIX stablo, iz koga korisnik bira direktorijum čiji sadržaj želi da vidi u tekućem panelu,
Find File	omogućava korisniku da u aktivnom stablu pronađe neku datoteku,
Compare Directories	upoređuje sadržaje levog i desnog panela,
Background Jobs	prikazuje spisak poslova i daje mogućnost da se izvršenje nekog posla zaustavi, nastavi ili da se posao prekine.

U meniju Options nalaze se sve opcije koje se odnose na konfigurisanje i prilagođavanje programa Midnight Commander.

6

SHELL PROGRAMIRANJE

Shell program (shell script) je datoteka koju čini niz Linux komandi koje se mogu izvršavati sekvencijalno ili povezati programskim strukturama tipičnim za više programske jezike, kao što su uslovni skokovi i petlje. Za razliku od programa napisanih u višim programskim jezicima (C i Pascal), shell program se ne prevodi, tj. ne zahteva se postojanje posebnog prevodioca i linkera. Komandni interpreter, koji je sastavni deo sistema, interpretira i izvršava program. Shell script funkcioniše kao MS-DOS batch datoteka, ali u odnosu na nju ima znatno više mogućnosti. Sam razvojni proces je nekonformniji u odnosu na klasični programske jezike, zato što nema programa za kontrolu toka programa (debugera) niti izvršenja korak po korak. Ulazni parametri se shell programu mogu zadati u komandnoj liniji ili interaktivno, tokom izvršenja programa, a izlaz se može formirati i prikazati na ekranu. Shell programiranje je veoma korisna tehnika kojom korisnik može kreirati specifične komande u cilju automatizacije svakodnevnih poslova.

Shell programi se pišu za određeni komandni interpreter. Korisnik može biti siguran da će shell program napisan za jedan komandni interpreter na njemu raditi ispravno - za druge komandne interpretere ne može se dati garancija o ispravnom izvršenju programa. Npr. ukoliko je program napisan za bash, niko ne može sa sigurnošću garantovati da će se on ispravno izvršiti na csh. Shell programiranje obuhvaćeno ovim poglavljem odnosi se na bash (Bourne Again Shell).

Shell programiranje obuhvata nekoliko bitnh elemenata:

- tipove podataka (koristi se univerzalni tip promenljive koji se ne deklariše, već se promenljiva automatski prilagođava tipu podataka),
- naredbe (koriste se sve UNIX komande, bilo interne bilo eksterne),
- programske strukture (uslovni skokovi, petlje i funkcije, odnosno podprogrami).

Osnovi shell programiranja

Jednostavan shell script. Pokretanje shell programa. Elementi shell programa. Sistemske i korisničke promenljive.

U nastavku teksta dat je primer jednostavnog shell programa koji štampa tekst "Goodbye, Cruel World !" na ekranu. Za pisanje shell programa može se koristiti bilo koji Linux tekst editor, poput vi ili joe, ili komanda cat. Komanda cat se može iskoristiti za pisanje jednostavnih programa - najpre se komanda zada sa imenom datoteke kao argumentom za redirekciju izlaza, zatim se otkuca sadržaj programa i na kraju se pritisne kombinacija tastera <Ctrl-D>.

```
$ cat >ss1
#
# ss1: jednostavan shell program
#
clear
echo "Goodbye, Cruel World !"
<CTRL-D>
```

Program je na ovaj način upisan u datoteku first. Dalje je potrebno dati korisnicima x (execute) dozvolu nad datotekom i pokrenuti program:

```
$ chmod +x ss1
$ ./ss1
```

Efekat našeg prvog shell programa je sledeći. Najpre će se obrisati ekran (komanda clear), a zatim će se na ekranu prikazati poruka Goodbye, Cruel World !

Komentari

U shell programu, sve linije koje počinju karakterom # smatraju se komentarima i kao takve komandni interpreter ih ignoriše. Komentari služe da objasne korišćenje programa, da prikažu autora, ili da pruže neka objašnjenja specifičnih delova programa.

Pokretanje shell programa

Pokretanje shell programa uključuje dva koraka: dodelu dozvola za izvršavanje i samo pokretanje programa iz komandne linije. Shell programi su tekstualne datoteke čija pristupna prava nakon kreiranja ne uključuju dozvolu za izvršavanje, tako da se ona mora eksplicitno dodeliti određenoj grupi korisnika.

```
$ chmod +x ss1      # dodela dozvole za izvršavanje
$ ./ss1             # pokretanje programa
```

Tačka (.) ukazuje na tekući direktorijum i mora da se navede ukoliko se tekući direktorijum ne nalazi u sistemskoj putanji. Ukoliko se program ne nalazi u tekućem direktorijumu potrebno je navesti i putanju do programa (apsolutnu ili relativnu).

```
$ /home/jsmith/development/x12/scripts/beta/ss1
```

Da bi se izbeglo pisanje relativno dugih imena putanja korisnicima treba dati dozvolu za izvršavanje shell programa, a zatim ga postaviti u neki direktorijum koji je uključen u sistemsku putanju PATH, kao što je /usr/bin. Na ovaj način se program može pokrenuti iz bilo kog direktorijuma bez navođenja putanje.

```
$ ss1
```

Kada korisnik zada komandu, komandni interpreter najpre proverava da li je to interna komanda, i ukoliko jeste, odmah je izvršava. U protivnom, komandi interpreter traži komandu u direktorijumima koji su specificirani sistemskom putanjom, odnosno promenljivom PATH i ukoliko je nađe izvršava je. U protivnom se na ekranu prikazuje karakteristična poruka "bash:xxxx:command not found".

Prilikom pokretanja programa može se specificirati komandni interpreter u kome će se program izvršavati. Potrebno je u prvu liniju programa upisati sledeće:

```
#!/bin/bash
```

Ukoliko se komandni interpreter ne specificira na ovaj način, program se izvršava u tekućem interpreteru.

Shell program se može pokrenuti i na drugi način, bez eksplicitne dodele x dozvole - dovoljno je pozvati komandni interpreter da izvrši shell program:

```
$ bash ss1
```

ili

```
$ /bin/sh ss1
```

Ukoliko se shell program ne nalazi u tekućem direktorijumu, potrebno je specificirati putanju do programa.

```
$ bash /home/jsmith/ss1
```

Za razvoj i korišćenje shell programa preporučuje se sledeća procedura: shell program treba razviti na svom direktorijumu, zatim ga istestirati pomoću komande bash shell-program, i na kraju iskopirati u neki direktorijum koji je podrazumevano uključen u sistemsku putanju. Program se može kopirati u bin poddirektorijum home direktorijuma autora. Ukoliko veći broj korisnika želi da koristi program datoteku treba kopirati u direktorijume /bin ili /usr/bin kojima mogu pristupati svi korisnici. Dodatno, korisnicima treba dati dozvolu execute da bi mogli da pokreću program pomoću imena datoteke.

Pronalaženje komandnog interpretera

Gotovo po pravilu, shell script programi počinju linijom #!/bin/bash, što ukazuje da će program izvršavati komandni interpreter bash iz direktorijuma /bin. Ukoliko se bash ne nalazi na tom nestu potrebno je locirati njegovu poziciju, što se može učiniti komandama find, which i whereis.

```
$ whereis bash
bash: /bin/bash /etc/bash.bashrc /usr/share/man/man1/bash.1.gz
```

Bourne Again Shell se najčešće nalazi u jednom od sledećih direktorijuma: /bin, /sbin, /usr/local/bin, /usr/bin, /usr/sbin i /usr/local/sbin/bash.

Promenljive u Linux operativnom sistemu

Jedan od bitnih elemenata shell programiranja je upotreba promenljivih. Promenljive omogućavaju čuvanje podataka u operativnoj memoriji računara. Sledi ukratko objašnjenje pojma promenljiva: RAM memorija se deli na manje fragmente - memorijske lokacije koje imaju memorijsku adresu, odnosno jedinstveni broj koji se koristi za obraćanje toj memorijskoj lokaciji. Programer dodeljuje simboličko ime memorijskoj lokaciji i na taj način kreira memorijsku promenljivu, ili kraće, promenljivu. Promenljiva je imenovana memorijska lokacija koja može sadržati različite vrednosti, ali samo jednu vrednost u jednom trenutku. Na Linux sistemima postoje dva tipa promenljivih:

- sistemske promenljive, koje kreira i održava sam operativni sistem. Ne preporučuje se promena njihovog sadržaja. Ovaj tip promenljivih definiše se strogo velikim slovima (Capital Letters);
- korisnički definisane promenljive (User defined variables - UDV), koje kreiraju i održavaju korisnici. Ovaj tip promenljivih obično se definiše malim slovima (Lowercase Letters);

U shell programiranju promenljive se ne deklarišu za specifični tip podataka - dovoljno je dodeliti vrednost promenljivoj i ona će biti alocirana prema toj vrednosti. U Bourne Again Shellu, promenljive mogu sadržavati brojeve, karaktere ili nizove karaktera.

Važnije sistemske promenljive

Sistemske promenljive mogu se videti pomoću komande:

```
$ set
BASH=/bin/bash
HOME=/home/jsmith
PATH=/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
PS1=[\u@\h \W]\$
PWD=/tmp/junk
SHELL=/bin/bash
USERNAME=jsmith
...
```

U nastavku teksta dat je opis značajnijih sistemskih promenljivih.

BASH	lokacija komandnog interpretera
HOME	home directorijum korisnika
PATH	putanja u kojoj se traže izvršne datoteke
PS1	podešavanje prompta

PWD	tekući direktorijum
SHELL	ime komandnog interpretera
USERNAME	ime korisnika koji je u ovom režimu trenutno prijavljen na sistem.

Pojedinačno, sadržaj promenljive može se videti pomoću komande echo:

```
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
```

Definisanje korisničkih promenljivih

Svaka promenljiva je univerzalna i nema nikakvu deklaraciju tipa (integer, float, string) i definiše se na sledeći način: variablename = value. Na taj način se vrednost value dodeljuje promenljivoj sa imenom variable. Vrednost se mora nalaziti sa desne strane znaka jednakosti. Na primer:

```
$ br=10             # ispravno
$ 10=br             # neispravno – vrednost na levoj strani
```

Prilikom definisanja, odnosno dodele vrednosti, potrebno je primeniti sledeće konvencije o imenima promenljivih:

- Ime promenljive mora početi alfanumeričkim karakterom ili donjom crtom '_' (underscore character) praćenim sa jednim ili više alfanumeričkih karaktera. Na primer, korektne promenljive su: HOME, SYSTEM_VERSION, br, _ime;
- Prazne karaktere ne treba stavljati ni sa jedne strane znaka jednakosti prilikom dodele vrednosti promenljivim. Na primer, sledeće deklaracije neće biti korektne, odnosno napraviće grešku zbog postojanja praznih karaktera:

```
$ br  =10           # neispravno – prazni karakteri
$ br=  10           # neispravno – prazni karakteri
$ br  =  10         # neispravno – prazni karakteri
$ br=10             # ispravno
```

- Promenljive su osetljive na velika i mala slova (case sensitive), kao i imena datoteka u Linux sistemu. Na primer, br, bR, Br i BR su četiri različite promenljive;

```
$ bR=20
$ Br=30
$ echo $bR
20
$ echo $Br
30
```

- Može se definisati promenljiva nulte dužine (NULL variable), odnosno promenljiva koja nema vrednost u trenutku definisanja. Vrednosti ovih promenljivih se logično ne mogu prikazati pomoću komande echo, sve dok se promenljivoj ne dodeli neka vrednost;

```
$ br=
```

```
$ ime=""
```

- Imena promenljivih ne smeju sadržati specijalne znake (poput ? i *).

Prikazivanje i korišćenje vrednosti UDV promenljivih

Da bi prikazali ili pristupili vrednosti promenljive potrebno je ukazati na njenu vrednost, što se postiže uporebom znaka $ sa imenom promenljive.

Na primer, da bi prikazali vrednost promenljive ime upotrebićete komandu:

```
$ echo $ime          # prikazuje vrednost promenljive ime
johnny
$ echo ime           # prikazuje string ime
ime
```

U nastavku teksta dati su primeri koji služe za bolje razumevanje promenljivih.

Primer 1. Definišite promenljive x i xn koje će imati vrednosti 10 i abc respektivno i prikažite njihove vrednosti na ekranu u istom redu.

```
$ x=10
$ xn=abc
$ echo $x $abc
10 abc
```

Primer 2. Koristeći naredbu expr, prikazati zbir dva broja na ekranu

```
$ echo 6+3           # echo posmatra 6+3 kao niz karaktera
6+3
$ expr 6 + 3         # expr posmatra 6 + 3 kao matematički izraz
9
```

Komanda expr određuje rezultat neke matematičke operacije. Sintaksa komande expr je:

```
expr op1 operacija op2
```

gde su op1 i op2 celi brojevi, a operator +, -, *, /, &. Relzultat operacije je ceo broj (20/3=6, 20%3=2). Argumenti op1, op2 i operator se moraju razdvojiti praznim karakterom.

```
$ expr 6 + 3         # ispravno
9
```

Primer 3. Definisati dve promenljive, x i y, i promenljivu z kao njihov količnik. Vrednosti promenljivih x i y su 20 i 5 respektivno. Rezultat prikazati na ekranu, u jednom redu.

```
$ x=20
$ y=5
$ z=`expr $x / $y`
$ echo x/y=$z
x/y=4
```

Komande specifične za shell programiranje

Sastavni deo shell programa su komande. To mogu biti standardne Linux komande (poput cp ili mv) ili komande specifične za shell programiranje. Neke od komandi specifičnih za shell programiranje su gotovo kompletni programski jezici (na primer awk). U nastavku teksta opisani su samo njihovi osnovni elementi, dok je za više informacija potrebno konsultovati on-line uputstva (man pages).

echo

Komanda echo prikazuje tekst ili vrednost promenljive na ekranu. Sintaksa komande echo je:

```
$ echo [opcije] [string, promenljive...]
```

Opcije:

-n	ova opcija ne prebacuje kursor u novi red,nakon izvršenja echo komande	
-e	omogućava interpretaciju sledećih karaktera u kombinaciji sa obrnutom kosom crtom:	
\a	upozorenje	(alert bell)
\b	povratak unazad	(backspace)
\c	ne prelaziti u novi red	(suppress trailing new line)
\n	novi red	(new line)
\r	povratak na početak reda	(carriage return)
\t	horizontalni tabulator	(horizontal tab)
\\	obrnuta kosa crta	(backslash)

Na primer:

```
$ echo –e "Linux\n\t\tRulez !\n"
Linux
          Rulez !
```

Navodnici

Linux prepoznaje tri tipa navodnika.

- Dvostruki navodnici - "Double Quotes". Sve što se nalazi u ovim navodnicima gubi originalno značenje (osim \ i $).
- Jednostruki navodnici - 'Single quotes'. Sve što je zatvoreno jednostrukim navodnicima ostaje nepromenjeno.
- Obrnuti navodnici `Back quote`. Izraz zatvoren obrnutim navodnicima tretira se kao komanda koju treba izvršavati.

Na primer, ukoliko korisnik želi da na ekranu prikaže tekući datum navešće izraz date sa obrnutim navodnicima:

```
$ echo "Današnji datum : date"        # tretira date kao string
Today is : date
$ echo "Današnji datum : `date`"      # tretira date kao komandu
Today is : Fri Apr  2 16:30:35 CEST 2004
```

Komanda expr i shell aritmetika

Naredba expr koristi se da obavi jednostavne aritmetičke operacije. Sintaksa komande je:

```
$ expr op1 operator op2
```

gde su op1 i op2 celi brojevi, a operator:

+	Sabiranje
-	Oduzimanje
*	Množenje (za množenje se koristi *, a ne *, pošto je * džoker)
/	Deljenje
%	Ostatak po modulu

Primer komande expr su:

```
$ expr 1 + 3
4
$ expr 20 % 3
2
$ echo 6+3=`expr 6 + 3`      # ispravno (obrnuti navodnici)
6+3=9
$ echo 6+3="expr 6 + 3"      # neispravno (obični navodnici)
expr 6 + 3=9
```

Na osnovu poslednjeg primera ukazaćemo na sledeća sintaksna pravila:

Naredba expr se kao parametar komande echo zadaje u obrnutim navodnicima, a ne u dvostrukim ili jednostrukim. Samo u tom slučaju echo posmatra expr kao komandu, a ne kao običan string.

read

Komanda read se koristi za čitanje ulaznih podataka sa tastature i memorisanje unete vrednosti u promenljivu. Dodatno, to je jednostavan i dobar način da se u razvojnoj fazi ubace prekidne tačke u program i na taj način kontroliše njegovo izvršavanje.

Sintaksa komande read je:

```
$ read varible1, varible2,...varibleN
```

Upotreba ove komande ilustrovana je programom ss2 koji čita ulazni podatak sa tastature u promenljivu var1, a zatim ga ispisuje na ekran.

```
#
# ss2: upotreba komande read
#
echo "Unesite podatak:"
read var1
echo "Uneli ste: $var1"
```

Program se pokreće na standardan način, komandom bash ss2.

```
# bash ss2
Unesite podatak: 123
Uneli ste: 123
```

sed (stream editor)

Editor sed je neinteraktivni editor - ime izvorišne datoteke i instrukcije za rad sa tekstom se navode kao parametri u komandnoj liniji. Funkcionisanje editora opisano je pomoću sledeća dva primera:

```
$ cat data
123456789
Roadrunner
9876-12345
Wile E. Coyote, genius
1234
$ sed 's/12345/abc/g' /tmp/data
abc6789
Roadrunner
9876-abc
Wile E. Coyote, genius
1234
$ sed 2, 4d /tmp/data
123456789
1234
```

U prvom slučaju sed editor zamenjuje niz karaktera '12345' datoteke /tmp/data nizom karaktera 'abc'. U drugom slučaju sed editor prikazuje sve linije datoteke /tmp/data osim linija 2 do 4. Rezultat se prikazuje na standardnom izlazu (stdout, odnosno ekran), a ulazna datoteka se ne modifikuje. Ukoliko je potrebno promene snimiti u datoteku, koristi se redirekcija izlaza:

```
$ sed 's/12345/abc/g' /tmp/data > /tmp/data1n
$ sed 5, 10d /tmp/data  > /tmp/data2n
```

awk

Osnovna funkcija komande awk je pronalaženje uzorka teksta u datoteci i izvršavanje neke akcije, najčešće obrade pronađenog teksta. Generalno, awk je tekst procesor realizovan putem jednostavnog programskog jezika, a najpoznatiji interpreteri su GNU awk (gawk) i mawk. Funkcionisanje awk prikazano je sledećim primerima:

```
$ cat /tmp/data
123abc
```

```
Wile E.
aabcc
Coyote
$ awk '/abc/ {print}' /tmp/data
123abc
aabcc
```

U ovom slučaj awk traži uzorak ‘abc’ u datoteci /tmp/data, a akcija koja se pri tom obavlja nad nađenim uzorcima je prikazivanje teksta na ekranu (print). Drugi primer je štampanje rednog broja prve linije teksta iza koje se traženi uzorak ne pojavljuje:

```
$ awk '/abc/ {i=i+1} END {print i}' /tmp/data
3
```

Ukoliko se u datoteci traži više uzoraka i ukoliko se vrši više obrada, potrebno je prvo napraviti datoteku u kojoj su opisane akcije (na primer actionfile.awk). Prilikom zadavanja komande awk potrebno je zameniti tekst između navodnika, kojim su opisani uzorak i akcija, imenom datoteke: '-f actionfile.awk'.

grep

Osnovna funkcija grep komande je da pronađe sve linije sa traženim uzorkom. Dodatno, grep može umesto prikazivanja da obavi prebrojavanje linija u kojima je uzorak pronađen.

```
# grep bora /etc/group
bora-admin:x:1003:
bora:x:1047:
# grep bora /etc/group -c
2
```

U ovom slučaju, niz "bora" je pronađen 2 puta u datoteci /etc/group.

wc (word count)

Komanda wc prebrojava broj linija, reči i bajtova u datoteci. Pretpostavimo da datoteka /tmp/data sadrži sledeći tekst:

```
$ less /tmp/text_file
First line of the text file
Second line of the text file
$ wc /tmp/text_file
2      12      58 text_file
```

Kao što se vidi iz primera komanda wc prikazuje rezultat u standardnom poretku: broj linija, broj reči, broj bajtova.

sort

Komanda sort uređuje linije tekstualne datoteke po abecedi ili nekom drugom kriterijumu (kao što su meseci u godini). U ovom slučaju datoteka /tmp/data sadrži sledeći tekst:

```
$ cat /tmp/data
```

```
bash
crypt
awk
$ sort /tmp/data
awk
bash
crypt
```

bc (basic calculator)

Program bc služi za izračunavanje vrednosti izraza - interaktivno ili sa komandne linije. Dodatno, bc ima ugrađene neke elemente programskog jezika (petlje i uslovi). Sledeći primer demonstrira upotrebu programa (parametar -q se navodi ukoliko je potrebno izbeći pozdravnu poruku):

```
$ bc -q
2 ^ 8
256
sqrt(9)
3
quit
$ cat bctest
print "Unesite pocetnu vrednost: "; i=read ();
print "Unesite krajnju vrednost: "; j=read ();
while (i<=j) {print i^2; print "\n"; i=i+1}
quit
$ bc -q bctest
Unesi pocetnu vrednost: 5
Unesi krajnju vrednost: 8
25
36
49
64
```

tput

tput inicijalizuje terminal ili postavlja upit u bazu podataka u kojoj su opisani terminali.

```
$ tput cup 10 4     # postavlja prompt u poziciju (y=10,x=4)
$ tput reset        # briše ekran i postavlja prompt u (y=1,x=1)
$ tput cols         # ova komanda prikazuje broj kolona
80
```

Komande, argumenti i izlazni status

Pretpostavimo da korisnik zadaje sledeću komandu:

```
$ tail -1 /etc/passwd
jsmith:x:1051:1051:John Smith,,,:/home/jsmith:/bin/bash
```

Postavlja se sledeće pitanje: šta se dešava kada korisnik zada prethodnu komandu? Prva reč (tail) je ime interne komande koju treba izvršiti ili programa koji treba pokrenuti. Sve ostalo iza imena predstavlja ulazne parametre komande, odnosno programa. To znači da se pokreće program tail iz direktorijuma koji je specificiran sistemskom putanjom, a da se parametri -1 i /etc/password prosleđuju tom programu.

Komanda može biti zadana bez parametara (npr. date, clear, who), kao i sa jednim ili više parametara (ls -l, ls -l /etc, mount -t ntfs /dev/hda1 /mnt/winc).

Promenljiva $# memoriše broj argumenata specifirane komandne linije, a $* ili $@ upućuju na sve argumente koji se prosleđuju shell programu.

Zašto se zahtevaju komandni argumenti ?

Na primeru komande rm može se jednostavno objasniti zašto je ponekad neophodno navesti komandne argumente.Naime, komanda rm se koristi za brisanje datoteka, tako da je neophodno da korisnik navede šta želi da obriše.

```
$ rm filename1 filename2 dir1
```

Ovde je rm ime komande, a argumenti su imena datoteka i direktorijuma koje korisnik želi da obriše. Takođe, navođenjem dodatnih parametara može se osim imena datoteka specificirati i način izvršenja komande (interaktivno ili forsirano, rekurzivno).

Shell program i komandni argumenti

Komandni argumenti se na isti način mogu zadati i shell programu. Na primer, za shell program ss3, koji je pozvan pomoću sledeće linije, prvi komandni argument je arg1, drugi arg2, a treći arg3:

```
$ ss3 arg1 arg2 arg3
```

U shell programu se argumentima komandne linije pristupa pomoću sledećih promenljivih:

- vrednost promenljive $0 je ime programa (u ovom slučaju ss3)
- vrednost promenljive $1 je prvi komandni argument (u ovom slučaju arg1)
- vrednost promenljive $2 je drugi komandni argument (u ovom slučaju arg2)
- vrednost promenljive $3 je treći komandni argument (u ovom slučaju arg3)
- vrednost promenljive $# je broj komandnih argumenta (u ovom slučaju dva)
- vrednost promenljive $* su svi komandni argumenti. $* se proširuje u `$0,$1,$2...$9` (što je u ovom slučaju arg1, arg2, arg3).

U nastavku teksta su na osnovu nekoliko komandi analizirane vrednosti specijalnih promenljivih: ime programa ($0), broj argumenata ($#), i aktuelne vrednosti argumenata ($1,$2 i tako dalje).

```
$ df
```

```
$ less /etc/passwd
$ ls -l /etc
$ mount -r /dev/hda2 /mnt/winc
```

ime programa	broj argumenata	argumenti		
$0	$#	$1	$2	$3
df	0			
less	1	/etc/passwd		
ls	2	-l	/etc	
mount	-r	-r	/dev/hda2	/mnt/winc

Upotreba argumenata u samom programu objašnjena je sledećim primerom koji na ekranu ispisuje ime programa, broj argumenata i sve argumente redom. Program se pokreće na standardan način (na primer: bash ss3 arg1 arg2 arg3).

```
#
# ss3: korišćenje argumenata komandne linije
#
echo "Ukupan broj argumenata komandne linije: $#"
echo "$0 je ime programa, a $1 je prvi argument."
echo "Svi argumenti su redom: $*"
```

Izlazni status komande

Nakon izvršenja Linux komande vraćaju vrednost na osnovu koje se može odrediti da li je komanda izvršena uspešno ili ne. Ako je povratna vrednost 0, komanda je izvršena uspešno. Ako je povratna vrednost različita od 0 (veća od 0), komanda se nije uspešno završila, a taj broj predstavlja neku vrstu dijagnostičkog statusa koja se naziva izlazni status. Da bi se odredila ova vrednost koristi se sistemska promenljiva **$?**, čija je upotreba demonstrirana sledećim primerima:

```
$ rm plumph
rm: cannot remove `plumph': No such file or directory
$ echo $?
1               # izlazni status 1 -> komanda izvršena s greškom
$ date
$ echo $?
0               # izlazni status 0 -> komanda izvršena bez greške
```

Grupisanje komandi

Bash obezbeđuje dva načina grupisanja komandi, tj. izvršavanja više komandi u okviru jedne celine. Kada se komande grupišu redirekcije mogu da se primene na celu grupu komandi. Na primer, izlaz svih komandi u listu može da se redirektuje u jedan niz.

(list)

Postavljanje liste komandi između malih zagrada ima za posledicu kreiranje posebnog shell potprograma u kome se izvršavaju sve komande iz liste. Pošto se lista komandi izvršava u shell podprogramu dodeljene promenljive prestaju sa važenjem nakon završetka potprograma. Male zagrade su operatori, njih komandni interpreter prepoznaje kao specijalne karaktere, čak i ukoliko nisu razdvojeni od list komandi praznim karakterima.

{ list; }

Postavljanje liste komandi između vitičastih zagrada ima za posledicu izvršavanje komandi iz liste u tekućem shell kontekstu. Za razliku od prethodnog slučaja poseban shell podprogram (proces) se ne kreira. Na kraju liste zahteva se znak ; ili prazan red. Velike zagrade su rezervisane reči, tako da one praznim karakterima moraju biti odvojene od liste komande.

Izlazni status za obe ove konstrukcije je izlazni status liste.

Parametri komandnog interpretera (Shell Parameters)

Postoje dve klase parametara komandnog interpretera: pozicioni parametri koji predstavljaju argumente komandne linije i specijalni parametri, koji imaju specijalno značenje za shell.

Parametar je celina koja memoriše vrednosti. To može biti ime, broj, ili jedan od specijalnih karaktera koji su opisani u nastavku teksta. Za svrhu komandnog interpretera, promenljiva je parametar koji je označen imenom. Promenljiva ima vrednost i atribute (0 ili više atributa). Atributi se dodeljuju korišćenjem deklarativnih ugrađenih komandi. Parametar je postavljen ako mu je dodeljena vrednost. Niz veličine 0 je validna vrednost. Kad se promenljiva postavi, to se može poništiti korišćenjem ugrađenih komandi za poništenje (unset builtin command).

Promenljiva se postavlja naredbom u sledećoj formi: name=[value]. Ako parametar value nije dat promenljivoj se dodeljuje niz veličine 0 (null string). U vrednosti promenljive ubrajaju se tilda proširenje, parametarsko proširenje i proširenje varijabli, komandna zamena, aritmetičko proširenje, i upotreba navodnika. Ako promenljiva ima svoj celobrojni atribut postavljen, tada se vrednost value pokorava aritemtičkom proširenju, čak i ako se $((...)) proširenje ne koristi. Razdvajanje reči se ne izvršava, sa izuzetkom "$@" kao što će biti objašnjeno kasnije. Proširenje imena datoteke se ne izvršava. Naredbe za dodeljivanje mogu takođe da se pojave kao argument za deklarisanje, postavljanje tipa, eksportovanje, nepromenljivost (readonly) i lokalne ugrađene komande.

Pozicioni parametri (positional parameters)

Pozicioni parametar se označava pomoću jedne ili više cifara, različitih od jednocifrenog broja 0. Pozicioni parametri se dodeljuju iz argumenata komandnog interpretera kada se pozove, a mogu se ponovo dodeliti koristeći unutrašnje komande. Pozicioni parametar N referencira se kao ${N}, ili kao $N, u slučaju kada se N sastoji od jedne cifre. Pozicioni

parametri ne mogu se postavljati naredbama za postavljanje name=[value]. U tu svrhu koriste se interne komande set i shift. Pozicioni parametri se privremeno zamenjuju kada se shell funkcija izvršava.

Specijalni parametri (special parameters)

Komandni interpreter tretira neke parametre specijalno. Ovi se parametri mogu samo referencirati, njihovo dodeljivanje nije dozvoljeno.

*	Proširuje se u pozicione parametre, počev od prvog ($1). Kada se proširivanje dogodi unutar duplih navodnika proširuje se jedna reč sa vrednošću svakog parametra, razdvojenog prvim karakterom specijalne promenljive koja se zove IFS. To znači, "$*" je ekvivalentno sa "$1c$2c...", pri čemi je c prvi karakter vrednosti IFS promenljive. Ako IFS nije postavljena parametri se razdvajaju praznim karakterom (space). Ako je IFS niz veličine 0 (null), parametri se udružuju bez međusobnog separatora.
@	Proširuje se u pozicione parametre, počevši od jedan. Kada se proširivanje dogodi unutar duplih navodnika, svaki parametar se proširuje u posebnu reč. To znači, "$@" je ekvivalentno sa "$1" "$2" ...".Kada nema nijednog pozicionog parametra, "$@" i $@ se ne mogu proširiti, te se uklanjaju.
#	Proširuje broj pozicionih parametara u decimalni.
?	Proširuje se u izlazni status poslednje izvršene foreground komande.
-	Proširuje se u tekuće opcione zastavice (flegove) koje su specificirane nakon pozivanja, a koje su postavile unutašnje komande ili komandni interpreter (kao na primer –i opcija).
$	Proširuje se u broj procesa (process ID) komandnog interpretera. U slučaju podprograma, parametar se proširuje u ID komandnog interpretera koji je pozvao potprogram, a ne u ID podprograma.
!	Proširuje se u broj procesa poslednje izvršene background komande.
0	Proširuje se u ime komandnog interpretera ili programa za taj komandni interpreter. To se postavlja prilikom inicijalizacije komandnog interpretera. Ako se Bash pozove sa script datotekom, $0 se postavlja na ime te script datoteke. Ako se Bash startuje sa –c opcijom, $0 se postavlja na prvi argument nakon što niz počne da se izvršava, ako je taj argument prisutan. U suprotnom, postavlja se na ime datoteke koja je pozvala Bash, kao sa argumentom 0.

Redirekcija i pipe mehanizam

Redirekcija ulaza i izlaza

Standardni ulaz (stdin), standardni izlaz (stdout) i standardni izlaz za greške (stderr) su deskriptori datoteke (file descriptor) kojima su dodeljeni brojevi po sledećim pravilima: 0

predstavlja stdin, 1 predstavlja stdout i 2 predstavlja stderr. U osnovi, u okviru shell programa mogu da se obave sledeće redirekcije, odnosno preusmerenja:

- redirekcija stdout u datoteku. Sledeći primer demonstrira kreiranje datoteke myfiles.txt i upis rezultata izvršenja komande ls-l u datoteku;

```
$ ls –l > myfiles.txt
```

- redirekcija stderr u datoteku. Sledeći primer demonstrira kreiranje datoteke greperr.txt i upis poruka o greškama koje proizvodi komanda grep u datoteku;

```
$ grep kyuss * 2> greperr.txt
```

- redirekcija stdout u stderr;
- redirekcija stderr u stdout, koja je demonstrirana sledećim primerom. Rezultat izvršenja komande grep smešta se u bafer i može se naknadno videti, a poruke o greškama koje komanda grep proizvodi prikazuju se na standardnom izlazu, a to je u podrazumevanom stanju ekran;

```
$ grep kyuss * 2>&1 greperr.txt
```

- redirekcija stderr i stdout u stdout. Rezultat izvršenja komande i poruke o greškama koje komanda proizvodi prikazuju se na standardnom izlazu, a to je u podrazumevanom stanju ekran;
- redirekcija stderr i stdout u stderr;
- redirekcija stderr i stdout u datoteku. Ova vrsta redirekcije je korisna za programe koji rade u pozadini (background), tako da se od njih očekuje da poruke ne upisuju na ekran, već u neku datoteku. Dodatno, ukoliko korisnik ne želi da vidi "feedback" komande, izlaz i poruke o greškama mogu se preusmeriti na uređaj /dev/null, kao što je prikazano sledećim primerom:

```
$ rm –f $(find / –name core) &> /dev/null
```

- redirekcija datoteke u stdin.

Pomoću komande less mogu se videti i stdout i stderr - na primer, stdout će ostati u baferu dok se stderr prikazuje na ekranu. Bafer se može naknadno pregledati, nakon čega se briše.

Pipe mehanizam

Pipe mehanizam omogućava korišćenje standardnog izlaza jedne komande kao standardnog ulaza druge komande. Primer pipe mehanizma je komanda:

```
$ ls –l | wc –l
36
```

Komanda ls -l će izlistati tekući direktorijum (jedna datoteka u jednom redu), a pipe će standardni izlaz ove komande preusmeriti na standardni ulaz komande wc -l, koja broji redove teksta. Na ovaj način korisnik dobija informaciju o broju datoteka u tekućem direktorijumu.

Shell proširenja (Shell Expansions)

Proširenje preko zagrada, tilda proširenje, proširenje parametara i promenljivih, zamena komandi, aritmetičko proširenje, razdvajanje reči, proširenje imena datoteke.

Proširenje se izvršava na komandnoj liniji. Bash prepoznaje sedam proširenja i izvršava ih sledećim redom:

- proširenje preko zagrada (brace expansion)
- tilda proširenje (tilde expansion)
- proširenje parametara i promenljivih (parameter and variable expansion)
- aritmetičko proširenje (arithmetic expansion)
- zamena komandi (command substitution), koja se obavlja sleva nadesno
- razdvajanje reči (word splitting)
- proširenje imena datoteke (filename expansion).

Samo proširenje u zagradama, razdvajanje reči i proširenje imena datoteka mogu promeniti broj reči u proširenju, ostala proširenja proširuju jednu reč u jednu reč.

Proširenje preko zagrada (Brace Expansion)

Proširenje preko zagrada je mehanizam kojim se mogu proširiti proizvoljni nizovi. Ovaj mehanizam je sličan proširenju imena datoteke, ali generisana imena datoteka ne moraju da postoje. Uzorci koji se preko zagrada proširuju uzimaju formu opcionog uvodnog dela, koju prati serija zapetom razdvojenih nizova između para

zagrada, iza kojih ide opcioni dotatak. Uvodni deo je prefiks svakog niza koji se nalazi unutar zagrada, a dodatak se dodaje s desne strane na svaki rezultujući niz.

Proširenja preko zagrada mogu da se gnezde, odnosno da se umeću jedno u drugo. Rezultati svakog proširenog niza nisu sortirani, samo se poštuje poredak sleva nadesno, odnosno prefiks, zatim niz iz zagrade, i na kraju dodatak-sufiks. Jednostavan primer je proširenje komande echo:

```
$ echo a{d,c,b}e
ade ace abe
```

Proširenje preko zagrade se izvršava pre bilo kog drugog proširenja. Bilo koji karakter koji ima specijalno značenje za ostala proširenja čuva se u rezultatu, odnosno ne dira se. To je strogo tekstualno proširenje. Bash ne primenjuje interpretaciju u kontekstu proširenja ili teksta između zagrada. Da bi izbegavao konflikte sa parametarskim proširenjima niz "${" se ne smatra pogodnim za proširenje preko zagrada. Korektno formirano proširenje preko zagrada mora sadržavati otvorenu i zatvorenu zagradu koje su van navodnika, i barem jednu zapetu. Svako nekorektno proširenje se ne izvršava.

Ova konstrukcija se tipično koristi kao skraćenica kada se isti zajednički prefiks generiše više puta, kao što je prikazano u sledećim primerima. Tako se:

```
$ mkdir /home/jsmith/{data,video,mp3}
```

proširuje u:

```
$ mkdir /home/jsmith/data
$ mkdir /home/jsmith/video
$ mkdir /home/jsmith/mp3
```

Komplikovaniji slučaj je korišćenje ugnježđenih proširenja.

```
$ chown root /home/{jsmith/{ss1,ss2},nmacek/{data,ss3}}
```

proširuje se u:

```
$ chown root /home/jsmith/ss1
$ chown root /home/jsmith/ss2
$ chown root /home/nmacek/data
$ chown root /home/nmacek/ss3
```

Tilda proširenje (Tilde Expansion)

Ako reč počinje tilda karakterom koji nije pod navodnicima (~), svi karakeri do prve kose crte koja je takođe van navodnika (/ slash) tretiraju se kao tilda prefiks. Ukoliko nema kose crte onda su svi karakteri tilda prefiks.

Ukoliko nema karaktera pod navodnicima unutar tilda prefiksa, tilda prefiks se tretira kao potencijalno ime korisnika za login proceduru (login-name). Tilda prefiks se zamenjuje po sledećim pravilima: ako je login-name nulte dužine, tilda se zamenjuje vrednošću HOME shell promenljive, a ako je HOME promenljiva nepostavljena, tilda se zamenjuje home direktorijumom korisnika koji izvršava taj komandi interpreter.

U drugom slučaju tilda prefiks se zamenjuje home direktorijumom specificiranog korisnika (login-name). Ako je vrednost tilda prefiksa "~+", tada tilda prefiks uzima vrednost shell promenljive PWD koja predstavlja tekući radni direktorijum. Ako je vrednost tilda prefiksa "~-", tada tilda prefiks uzima vrednost shell promenljive OLDPWD koja predstavlja prethodni tekući radni direktorijum (pod uslovom da je OLDPWD setovana).

Ako je login-name pogrešan, tilda proširenje se ne izvršava, reč s leve stane ostaje nepromenjena. Svaka dodela promenljivoj se proverava za tilda prefikse van navodnika iza kojih neposredno ide : ili =. U ovim slučajevima tilda proširenje se takođe izvršava. Prema tome, nekom mogu koristiti imena datoteka sa tildom u dodeljivanju sistemskih promenljivih kao što je PATH, MAILPATH i CDPATH, a komandni interpreter će im dodeliti proširene vrednosti.

Sledeći primeri pokazuju kako Bash tretira tilda prefikse bez navodnika. Prvi primer demonstrira upotrebu tilda proširenja za pozicioniranje na home direktorijum:

~ vrednost promenljive $HOME (/home/jsmith)

~/data $HOME/data (/home/jsmith/data)

~jim home directorijum korisnika jim (/home/jim).

Primeri se oslanjaju na pretpostavku da je na sistem prijavljen korisnik jsmith, čiji je home direktorijum /home/jsmith i da on pokreće komande koje sadrže ova proširenja.

```
$ whoami
jsmith
$ pwd
/etc
$ cd ~/data              # poddirektorijum data home direktorijuma
$ pwd
/home/jsmith/data
$ cd ~jim                # home direktorijum korisnika jim
$ pwd
/home/jim
```

Sledeći primer demonstrira upotrebu tilda proširenja za promenljivu $OLDPWD:

~+/misc $PWD/misc

~-/temp $OLDPWD/temp

```
$ pwd
/etc
$ cd /bin
$ cd ~-/network
$ pwd
/etc/network
```

Parametarsko proširenje (Shell Parameter Expansion)

Znak $ uvodi parametarsko proširenje, komandnu zamenu ili aritmetičko proširenje. Ime parametra ili simbola koji se proširuje može biti zatvoreno u zagradama koje su opcione ali štite promenljivu koja se proširuje od karaktera koji je neposredno slede i koji bi se mogli pogrešno interpretirati kao deo imena. Kada se koriste zagrade, zadnja zagrada je prvi znak koji nije u sastavu obrnute kose crte ili pod navodnicima i nije u okviru aritmetičkih proširenja, komandnih zamena ili parametarskih proširenja.

Osnovna forma parametarskog proširenja je ${par}. Vrednost parametra (par) se zamenjuje. Zagrade se zahtevaju kada je reč o pozicionom parametru, sa više od jedne cifre, ili kada je parametar praćen karakterom koji se ne interpretira kao deo njegovog imena.

Ako je prvi karakter parametra znak uzvika "!", uvodi se nivo promenljive indirekcije. Bash koristi vrednost promenljive formirane od ostatka parametra kao ime promenljive. Ova promenljiva se zatim proširuje i ta vrednost se koristi u ostatku zamene, umesto vrednosti samog parametra. Ovo je poznato kao indirektno proširenje. Izuzetak ovog proširenja je slučaj ${!prefix*}.

${par:-word} Ako parametar ne postoji ili je null, zamenjuje se proširenjem word. U drugom slučaju zamenuje se vrednošću parametra.

${par:=word}	Ako parametar ne postoji ili je null, proširenje word se dodeljuje parametru. Vrednost parametra se tada zamenjuje. Pozicioni i specijalni parametri ne moraju se postavljati na ovaj način.
${par:?word}	Ako parametar ne postoji ili je null, proširenje word se upusuje na standardni izlaz za greške (standard error) i komandi interpreter prekida rad (exit). Vrednost parametra se tada zamenjuje.
${par:+word}	Ako parametar ne postoji ili je null, ništa se ne zamenjuje, a u drugom slučaju proširenje word se zamenjuje.
${par:offset:lenght}	Proširuje do dužine length karaktera parametra, počevši od karaktera specificiranog pomoću polja offset. Ako se polje length izostavi, proširuje se podniz počevši od karaktera specificiranog pomoću polja offset zaključno sa zadnjim karakterom. Polja length i offset su aritmetički izrazi. Ovo se još naziva i podnizno proširenje (Substring Expansion). Polje length mora biti broj veći ili jednak 0. Ako je polje offset broj manji od 0, vrednost se koristi kao pomeraj u odnosu na kraj vrednost i parametra. Ako je parametar "@", rezultat je polje length pozicionih parametara koji počinju u polju offset. Ako je parametar polje imena koje se indeksira pomoću "@" ili "*", rezultat je length polje članova polja koji počinju sa ${par[offset]}. Podnizno indeksiranje je bazirano na nuli, osim u slučaju kada se koriste pozicioni parametri, kada indeksiranje startuje u 1.
${!prefix*}	Proširuje imena promenljivih čija imena počinju prefiksom prefix, razdvojenim prvim karakterom IFS specijalne promenljive.
${#par}	Dužina proširene vrednosti parametra se zamenjuje. Ako je parametar "*" ili "@", zamenjena vrednost je broj pozicionih parametara. Ako je parametar polje imena koja se indeksiraju pomoću "@" ili "*", zamenjena vrednost je broj elemenata u polju.

Komandna zamena (Command Substitution)

Komandna zamena dozvoljava da se izlaz komande zameni samom komandom, odnosno da izlaz jedne komande postaje argumenat druge. Komanda zamena se izvršava kada se komanda zatvori zagradama ili navodnicima, kao u sledećim primerima:

```
$(command)
```

ili

```
`command`
```

Bash izvršava proširenje izvršavanjem komande command i zamenjuje komandnu substituciju sa standardnim izlazom komande. Ugrađene nove linije se ne brišu, ali mogu da se uklone za vreme razdvajanja reči.

Kada se koristi zamena sa starim stilom forme obrnutog navodnika, karakter obrnuta kosa crta zadržava doslovno značenje osim kada je praćen sa "$", "`", ili "\". Prvi obrnuti navodnik, koji nije praćen obrnutom kosom crtom, prekida komandnu zamenu. Kada se koristi $(command) forma, svi karakteri između malih zagrada tretiraju se kao komande, ništa se ne tretira specijalno.

Ako se zamena pojavljuje sa duplim navodnicima, razdvajanje reči i proširenje imena datoteka datoteka se ne izvršava.

Komandu zamenu demonstriraćemo preko kompresije grupe *.bak datoteka. Komanda find pronaći će sve takve datoteke:

```
$ find / –name ‘*.bak’ –print
```

Komprimovanje u jednoj komandi može se izvršiti na dva načina:

```
$ gzip ` find / –name ‘*.bak’ –print `
```

ili

```
$ gzip $( find / –name ‘*.bak’ –print )
```

Dodatno, pomoću komandne zamene mogu se dodeliti vrednosti promenljivim.

```
$ x = `date`
$ echo $x
Thu Apr 15 09:53:44 CEST 2004
$ y = `who am i;pwd`
$ echo $y
nmacek pts/0 Apr 15 09:40 (nicotine.internal.vets.edu.yu)
/home/nmacek
```

Aritmetičko proširenje (Arithmetic Expansion)

Aritmetičko proširenje omogućava izračunavanje aritmetičkog izraza i zamenu rezultata. Format aritmetičkog izraza je:

```
$(( expression ))
```

ili

```
$[ expression ]
```

Izraz se tretira kao da je bio u duplim navodnicima, ali dupli navodnici unutar zagrada se ne tretiraju specijalno. Svi simboli u izrazu podležu parametarskom proširenju, komandnoj zameni i navodničkom uklanjanju. Aritmetičke zamene mogu da se gnezde.

Izračunavanje se izvršava prema pravilima shell aritmetike. Ako je izraz pogrešan bash prikazuje poruku koja prijavljuje otkaz i zamena se ne izvršava.

Evo nekoliko primera:

```
$ echo 1 + 1        # shell interpretira 1 + 1 kao string
1 + 1
$ echo $((1+1))     # $((1+1)) je aritmetičko proširenje
2
$ echo $((7/2))     # bash koristi celobrojnu aritmetiku
3
$ echo 3/4|bc -l    # celobrojna aritmetika
0.75
$ a=1
$ b=2
$ echo $(($a+$b))   # promenljive i aritmetičko proširenje
3
```

Bash koristi celobrojnu aritmetiku - komanda echo $[3/4] na ekranu prikazuje 0. Ukoliko je potrebno izvršiti neku operaciju sa realnim rezultatom ili više matematičkih operacija, može se koristiti program bc - rezultat komande echo 3/4|bc -l je 0.75, što je korektno.

Aritmetičko proširenje se može iskoristiti za određivanje istinitosti izraza. U tom slučaju, proširenje vraća status 0 ili 1 zavisno od izračunavanja uslovnog izraza expression. Izraz se komponuje pomoću operatora <, <=, >, >=, == i ! =. Dodatno, izrazi mogu da se kombinuju koristeći sledeće operatore:

(expression)	vraća vrednost izraza expression. Ovo se može koristiti za nadjačavanje normalnog prioriteta operatora.
! expression	tačno ukoliko je expression netačan (negacija)
exp1 && exp2	tačno samo pod uslovom ako su oba izraza (exp1 i exp2) tačni
exp1 ‖ exp2	tačno ako je bar jedan od izraza (exp1 ili exp2 tačan).

Operatori && i ‖ ne izračunavaju vrednost exp2 ako je vrednost izraza exp1 dovoljna da odredi povratnu vrednost celog uslovnog izraza.

```
$ echo $((1>3||2<4))
1
$ echo $((1>3&&2==2))
0
```

Kada se koriste operatori "==" i "!=" niz desnog operatora smatra se uzorkom, a provera identičnosti odgovara pravilima za pronalaženje uzorka (Pattern Matching). Vrednost 0 se vraća ako niz odgovara uzorku, a vrednost 1 ako ne odgovara. Razdvajanje reči i proširenje imena datoteka se ne izvršavaju unutar ovog proširenja; tilda proširenje, parametarsko proširenje, komandna zamena, procesna zamena i upotreba navodnika se izvršavaju.

```
$ ime=jsmith
$ echo $(($ime==jsmith))
1
$ echo $(($ime!=jsmith))
0
```

Proširenje imena datoteka (Filename Expansion)

Nakon zadavanja komande, Bash razdvaja reči koje predstavljaju parametre i u parametrima koji predstavljaju datoteke traži karaktere "*", "?", i "[". Ako se jedan od tih karaktera pojavi tada se reč smatra uzorkom i zamenjuje se alfabetski sortiranom listom imena datoteka koja odgovara uzorku. Ukoliko je Bash pokrenut sa parametrom -f ova zamena se ne izvršava.

Pronalaženje uzorka (Pattern Matching)

Prilikom pronalaženja uzorka specijalni karakteri imaju sledeće značenje:

*	odgovara bilo kom nizu uključujući i niz nulte dužine. Tako će, na primer, komanda ls * prikazati sve datoteke, ls a* sve datoteke čije ime počinje sa a, a ls *.c sve datoteke koje imaju ekstenziju .c;
?	odgovara bilo kom karakteru. Tako će, na primer, ls ? prikazati sve datoteke čije ime ima tačno jedan karakter, a ls fo? sve datoteke čije ime ima tačno tri karaktera, od kojih su prva dva fo;
[...]	odgovara jednom od karaktera koji je naveden između zagrada. Ukoliko je prvi karakter iza otvorene zagrade "!" ili "^" tada odgovaraju svi karakteri koji nisu navedeni između zagrada. Na primer, ls [abc]* će prikazati sve datoteke čije ime počinje slovima a,b ili c, a ls [^abc]* sve datoteke čije ime ne počinje tim slovima;
[..-..]	par karaktera razdvojen znakom "-" označava zonu, odnosno opseg. Ukoliko je prvi karakter iza otvorene zagrade "!" ili "^" tada odgovaraju svi karakteri koji ne pripadaju opsegu. Na primer, ls /bin/[a-f]* će prikazati sve datoteke direktorijuma /bin čije ime počinje slovima a,b,..f, a ls /bin/[^a-e]* sve datoteke direktorijuma /bin čije ime ne počinje tim slovima;

Konstrukcije u shell programiranju

Uslovne konstrukcije (if, case), petlje (while, until, for, select). Funkcije.

Od konstrukcija koje su karakteristične za više programske jezike, u shell programima se mogu koristiti:

- uslovne konstrukcije (if, case),
- petlje (while, until, for, select),
- funkcije.

Uslovne konstrukcije

Uslovna konstrukcija if

if konstrukcija se može primeniti u tri osnovna oblika:

- if-then-fi
- if-then-else-fi
- if-then-elif-else-fi.

Glavni deo svake if naredbe je provera uslova koja se obavlja pomoću interne komande test i na osnovu čijih rezultata se odlučuje koje će se naredbe izvršavati.

U najkompleksnijem obliku sintaksa if komande je:

```
if test-commands;
then
    consequent-commands;
[elif more-test-commands;
    then
    more-consequents;]
[else alternate-consequents;]
fi
```

If petlja funkcioniše na sledeći način. Najpre se ispituju uslovi izvršavanjem liste komandi test-commands čiji povratni status određuje da li su uslovi ispunjeni. Uslovi su ispunjeni ako je povratni status 0, što znači da se tada izvršava lista komandi consequent-commands i to je kraj if petlje. Ukoliko uslovi nisu ispunjeni test-command će vratiti status različit od 0, a onda će se svako elif testiranje izvršavati redom do prvog ispunjenja uslova, kada će se izvršiti more-consequents naredbe iz odgovarajuće elif strukture. Ukoliko je else struktura prisutna, a svi prethodni testovi otkažu, počevši od glavne if strukture pa preko svih elif struktura, tada će se izvršiti lista naredbi alternate-consequents koju sadrži else struktura. Povratni status cele if strukture je izlazni status zadnje izvršene komande, ili 0 ako nijedan od postavljenih uslova u if i elif strukturama nije ispunjen, a else ne postoji.

if-then-fi

U ovom obliku if komande izvršiće se jedna ili grupa komandi ukoliko je uslov ispunjen, a u protivnom cela if struktura ne radi ništa.

Sintaksa najprostijeg oblika je:

```
if condition
then
    command 1
    ...
    command n
fi
```

U najprostijem svom obliku komanda if testira uslov condition i ako je on ispunjen, izvršava jednu ili grupu komandi (zaključno sa komandom pre komande fi, koja je kraj if strukture).

Provera uslova i test naredba

Provera uslova realizuje se preko izlazne vrednosti koja može biti 0, što znači da je uslov ispunjen, ili različita od 0, što znači da uslov nije ispunjen.

Uslov condition može biti, na primer, komparacija dve vrednosti koju obavlja komanda test ili njen skraćeni oblik pisanja [expression]. Izraz expression je kombinacija vrednosti, relacionih operatora kao što je >,<, == ili matematičkih operatora kao što su +, - , /.

Drugi oblik uslova condition je izlazni status komande. Svaka Linux komanda vraća status koji opisuje da li je uspešno izvršena ili ne. Status zadnje izvršene komande smešta se u promenljivu ?, što je demonstrirano sledećim shell programom:

```
#
# ss4: korišćenje izlaznog statusa komande
#
if rm $1
then
    echo "Datoteka $1 je uspešno obrisana"
fi
```

Program se pokreće pomoću komande bash ss4 filename. Argument filename je prvi argument ($1). U uslovnom delu mi kompariramo da li datoteka postoji: if rm $1 (odnosno if rm filename). Ako komanda rm pronađe datoteku i uspešno je obriše, njen izlazni status je 0, te će se izvršiti naredba ispod then (echo "Datoteka $1 je uspešno obrisana). U protivnom, izlazni status je različit od 0, i komanda se izvršava.

```
$ bash ss4 non_existing
rm: cannot remove `non_existing': No such file or directory
$ bash ss4 existing
Datoteka existing je uspešno obrisana
```

Provera tačnosti izraza komandom test

Komanda test se koristi za proveru tačnosti izraza u uslovnim konstrukcijama. Ukoliko je izraz tačan komanda vraća vrednost 0, odnosno vrednost veću od nule ako je izraz netačan.

Sintaksa komande test je:

```
test  expression
```

ili:

```
[ expression ]
```

U nastavku teksta dat je jednostavan shell program koji određuje da li je numerički argument pozitivan.

```
#
# ss5: Da li je argument pozitivan broj ?
#
if test $1 -gt 0        # alternativno: if [ $1 -gt 0 ]
then
    echo "$1 je pozitivan broj"
fi
```

Program se pokreće komandom: bash ss5 arg, gde je arg numerička vrednost.

```
$ bash ss5 5
5 je pozitivan broj
$ bash ss5 -4
```

Linija if test $1 -gt 0, utvrđuje tačnost izraza $1>0, odnosno proverava da li je prvi argument komande ($1) veći od 0. U prvom slučaju uslov je ispunjen, komanda vraća vrednost 0, a if konstrukcija pokreće komandu echo koja će prikazati poruku "5 je pozitivan broj". U drugom slučaju argument je -4, izraz 4>0 nije tačan, pa se komanda echo ne izvršava.

Pomoću komande test, odnosno [expr] mogu se upoređivati celi brojevi, upoređivati nizovi karaktera i može se odrediti da li datoteka postoji, da li je regularna, izvršna, itd.

Sledeći matematički operatori se koriste za upoređivanje celih brojeva:

Operator	Značenje	if test	if [ ]
-eq	jednakost (5=6)	if test 5 -eq 6	if [5 -eq 6]
-ne	nejednakost (5 != 6)	if test 5 -ne 6	if [5 -ne 6]
-lt	strogo manje od (5<6)	if test 5 -lt 6	if [5 -lt 6]
-le	manje od ili jednako (5<= 6)	if test 5 -le 6	if [5 -le 6]
-gt	strogo veće od (5>6)	if test 5 -gt 6	If [5 -gt 6]
-ge	veće od ili jednako (5>=6)	if test 5 -ge 6	If [5 -ge 6]

Za upoređivanje nizova koriste se sledeći operatori:

string1 = string2	da li je niz string1 jednak nizu string2
string1 != string2	da li je niz string1 različit od niza string2
string1	da li je string1 definisan i ako jeste da li nije NULL
-n string1	da li string1 nije NULL
-z string1	da li ¸je string1 NULL

Komandom test takođe se mogu izvršiti testovi nad datotekama i direktorijumima:

-s file	da li datoteka ima neki sadržaj
-f file	da li datoteka postoji, da li je obična a ne direktorijum
-d dir	da li direktorijum postoji, i da li nije obična datoteka

-w file	da li je datoteka sa pravom upisa
-r file	da li je datoteka bez prava upisa (read-only)
-x file	da li je datoteka izvršna

Dodatno, u komandi test mogu se koristiti logički testovi za kombinovanje dva ili više uslova istovremeno:

! expression	logička negacija
exp1 -a exp2	logička AND funkcija
exp1 -o exp2	logička OR funkcija

if-then-else-fi

Osnovna osobina ovog oblika if komande je postojanje jednog uslova i dve grupe komandi od kojih će, zavisno od toga da li je uslov ispunjen ili ne, izvršiti samo jedna.

Sintaksa if-then-else-fi strukture je:

```
if  condition
then
    command1-1
    ...
    ...
else
    command2-1

    ...
    ...
fi
```

Ukoliko je uslov condition ispunjen (izraz je tačan, ili je izlazni status komande 0), izvršiće se komanda ili grupa komandi command1, a ako nije izvršiće se command2.

U nastavku teksta dat je jednostavan shell program koji određuje da li je numerički argument pozitivan ili negativan.

```
#
# ss6: Da li je argument pozitivan ili negativan broj ?
#
if [ $# -eq 0 ]
then
     echo "$0 : Morate navesti jedan numerički argument"
     exit 1
fi
if test $1 -gt 0
then
     echo "$1 je pozitivan broj"
else
     echo "$1 je negativan broj"
fi
```

Program se pokreće komandom: bash ss6 arg, gde je arg numerička vrednost.

```
$ bash ss6 5
5 je pozitivan broj
$ bash ss6 -4
-4 je negativan broj
$ bash ss6
ss6: Morate navesti jedan numerički argument
$ bash ss6 0
0 je negativan broj
```

Program proverava broj ulaznih argumenata - ako je bilo koji argument prosleđen programu tada ($#) nije jednak 0 i dalje se nastavlja testiranje znaka broja. U protivnom, izvršiće se komanda echo "$0 : Morate navesti jedan numerički argument", a zatim exit 1, čime se prekida program sa izlaznim statusom 1, koji govori o otkazu programa. Ako je argument prosleđen programu prelazi se na drugu if strukturu, koja će odrediti da li je broj pozitivan ili negativan, pri čemu se 0 tretira kao negativan broj.

if-then-elif-else-fi

Ovaj oblik if strukture predstavlja proširenje prethodnog slučaja dodatnim uslovima i dodatnim grupama naredbi koji se mogu izvršiti. Osim prvog uslova i else grupe u strukturu se uvodi N-1 novih uslova i isto toliko novih grupa naredbi, od kojih se izvršava samo jedna, zavisno od toga koji je uslov tačan. Ukoliko nijedan uslov nije ispunjen izvršiće se grupa komandi u else bloku.

Sintaksa if-then-elif-else-fi strukture je:

```
if condition1
then
    command1
    ...
elif condition2
then
    command2
    ...
elif conditionN-1
then
    commandN-1
    ...
else
    commandN
    ...
fi
```

U nastavku teksta dat je jednostavan shell program koji određuje da li je argument numerički, i ukoliko jeste, da li je pozitivan, negativan ili nula.

```
#
# ss7: Da li je argument pozitivan, negativan ili nula
#
if [ $1 -gt 0 ]
then
     echo "$1 je pozitivan broj"
elif [ $1 -lt 0 ]
then
```

```
     echo "$1 je negativan broj "
elif [ $1 -eq 0 ]
then
     echo "Argument je nula"
else
     echo "$1 nije numerički argument"
fi
```

Program se pokreće komandom: bash ss7 arg:

```
$ bash ss7 1
1 je pozitivan broj
$ bash ss7 -2
-2 je negativan broj
$ bash ss7 0
Argument je nula
$ bash ss7 a
ss7: [: -gt: unary operator expected
ss7: [: -lt: unary operator expected
ss7: [: -eq: unary operator expected
a nije numerički argument
```

U poslednjem slučaju program obavlja tri poređenja celih brojeva sa nečim što nije broj i zato daje tri poruke o greškama, a tek posle toga štampa informaciju "a nije numerički argument".

Uslovna konstrukcija case

Naredba case je dobra alternativa višeslojnoj if-then-else-fi stukturi. Fleksibilnija je i lakša za pisanje koda. Kompaktna sintaksa case komande je:

```
case word in
[ [(] pattern [| pattern]...) command-list ;;]...
esac
```

Naredba case selektivno izvršava listu komandi koja odgovara prvom uzorku koji je identičan sa promenljivom word. Karakter `|' se koristi da razdavaja višestruke uzorke, a karakter `)' služi da označi kraj jedne liste uzoraka. Lista uzoraka i pridružena lista komandi naziva se član ili klauzula (clause) case naredbe. Svaki član se mora završiti karakterom `;;'. Ulaznu promenljivu word moguće je podvrgnuti tilda proširenju, parametarskom proširenju, komandnoj zameni, aritmetičkom proširenju i upotrebi navodnika pre nego što se zada pretraživanje uzoraka u case naredbi. Takođe, svaki uzorak moguće je podvrgnuti tilda proširenju, parametarskom proširenju, komandnoj zameni i aritmetičkom proširenju. Broj case članova je proizvoljan, a svaki član se završava sa `;;'. Prvi uzorak sa kojim se nalazi podudarnost u case naredbi određuje listu komandi koja će se izvršavati.

U nastavku teksta dat je pregledniji oblik sintakse:

```
case  $variable-name  in
    pattern1)
        command
        ...
        ...
```

```
        command;;
    pattern2)
        command
        ...
        ...
        command;;
    patternN)
        command
        ...
        ...
        command;;
    *)
        command
        ...
        ...
        command;;
esac
```

Promenljiva $variable-name upoređuje se sa svim uzorcima do prvog podudaranja, nakon čega shell izvršava sve naredbe u tom bloku, zaključno sa naredbom iza koje se nalaze dve dvotačke ;;. U slučaju da nema podudaranja izvršava se grupa naredbi iza *) (default).

Povratni status cele case strukture je 0, ukoliko nijedan uzorak ne odgovara ulaznoj promenljivoj word. U suprotnom, povratni status jednak je izlaznom statusu poslednje naredbe iz komandne liste koja će se izvršiti.

Korišćenje case strukture demonstrirano je programom ss8:

```
#
# ss8: korišćenje case strukture
#
if [ -z $1 ]
then
    echo "*** Unesite korisničko ime ***"
    exit 1
fi
echo -n "Korisnik sistema: "
case $1 in
     "jsmith") echo "John Smith, jr.";;
     "nmacek") echo "Nemanja Maček";;
     "bora") echo "Borislav Đorđević";;
     "dragan") echo "Dragan Pleskonjić";;
     *) echo "*** Nepostojeći korisnik ***";;
esac
```

Program najpre proverava da li je zadat ulazni argument, a zatim na osnovu njegove vrednosti prikazuje na ekranu odgovarajuću poruku.

Petlje

while petlja

Računar može izvršavati grupu instrukcija više puta sve dok su ispunjeni izvesni uslovi. Grupa instrukcija koja se ponavlja zove se petlja (loop). Bash komandni interpreter

podržava until, while, for i select petlje. Naznačimo da pojava simbola ";" u opisu sintakse komande znači da ";" može biti zamenjena jednim ili više novih redova od kojih svaki sadrži komandu. Interne naredbe komandnog interpretera break i continue mogu se koristiti za kontrolu izvršenja petlji.

Komanda while izvršava grupu komandi u petlji sve dok su ispunjeni odgovarajući uslovi. Kompaktna sintaksa while petlje:

```
while test-commands;
do
   consequent-commands;
done
```

While petlja se izvršava na sledeći način: najpre se provere uslovi, i ako su ispunjeni (izlazni status grupe test-commands je 0), izvršava se grupa komandi consequent-commands. Petlja prestaje sa izvršavanjem ako uslov više nije ispunjen, odnosno kada izlazni status grupe test-commands postane različit od 0. Izlazni status while petlje jednak je izlaznom statusu zadnje izvršene komande u grupi consequent-commands, ili 0, ako nijedna komanda iz grupe nije izvršena (na primer, ako uslovi nisu odmah ispunjeni, pa petlja praktično nema ni jednu iteraciju).

Pregledniji oblik sintakse while petlje je:

```
while [ condition ]
do
    command1
    command2
    ...
    commandN
done
```

Program ss9 demonstrira upotrebu while petlje:

```
#
# ss9: tablica množenja realizovana while petljom
#
if [ $# -eq 0 ]
then
    echo "Greška - numerički argument nije naveden"
    echo "Sintaksa : $0 broj"
    echo "Program prikazuje tablicu množenja za dati broj"
    exit 1
fi
n=$1      # Postavi vrednost argumenta u promenljivu n
i=1       # Inicijalizacija promenljive i (brojača)
while [ $i -le 10 ]     # Uslov petlje - blok se izvršava dok je
i<10
do
    echo "$n * $i = `expr $i \* $n`"  # Prikazuje npr. 6*4=24
    i=`expr $i + 1`     # Inkrementira promenljivu i
done
```

Na osnovu ovog primera mogu se uočiti tri bitne činjenice vezane za petlje:

- promenljiva koja se koristi u uslovu petlje mora da se inicijalizuje pre početka petlje,

- provera uslova se obavlja na početku svake iteracije (condition),
- telo petlje završava se ili mora da sadrži naredbu koja modifikuje vrednost promenljive koja se koristi u uslovu, inače je petlja beskonačna (nema izlaska iz petlje).

until petlja

Until petlja, potpuno suprotno while petlji, izvršava grupu komandi u petlji sve dok se odgovarajući uslovi ne ispune. Sintaksa until komande je:

```
until test-commands;
do
    consequent-commands;
done
```

until petlja se izvršava na sledeći način: najpre se provere uslovi i ako nisu ispunjeni (izlazni status grupe test-commands je različit od 0), izvršava se grupa komandi consequent-commands. Petlja prestaje sa izvršavanjem kad se uslovi ispune, odnosno kada izlazni status grupe test-commands postane 0. Izlazni status until petlje jednak je izlaznom statusu poslednje izvršene komande iz grupe consequent-commands, ili 0, ako nijedna komanda iz grupe nije izvršena (na primer, ako su uslovi odmah ispunjeni, pa petlja praktično nema ni jednu iteraciju).

Program ss10 demonstrira upotrebu while petlje:

```
#
# ss10: upotreba until petlje
#
c=20
until [  $c -lt 10 ]
do
    echo c = $c
    let c-=1
done
```

for petlja

Sintaksa for petlje ne liči mnogo na sintaksu C jezika:

```
for name [in words ...];
do
    commands;
done
```

Komanda for će razviti listu words, i za svakog člana redom u rezultantnoj listi će izvrštiti grupu komandi commands, pri čemu promenljiva name dobija vrednost tekućeg člana liste. Ako se `in words' izostavi u naredbi select, ili ako se specificira `in "$@"', tada će name uzimati vrednost pozicionih parametara. Izlazni status for petlje jednak je izlaznom statusu zadnje izvršene komande u grupi commands. Ako je lista words prazna nijedna komanda se neće izvršiti i tada će izlazni status biti 0.

Program ss11 demonstrira upotrebu for petlje:

```
#
# ss11: upotreba for petlje
#
if [ $# -eq 0 ]
then
    echo "Greška – numerički argument nije naveden"
    echo "Sintaksa : $0 broj"
    echo "Program prikazuje tablicu množenja za dati broj"
    exit 1
fi
n=$1
for i in 1 2 3 4 5 6 7 8 9 10
do
     echo "$n * $i = `expr $i \* $n`"
done
```

For petlja najpre kreira promenljivu i, a zatim joj redom dodeljuje vrednosti iz liste (u ovom slučaju numeričke vrednosti od 1 do 10). Shell izvršava echo naredbu za svaku vrednost promenljive i.

Alternativna forma for petlje podseća na sintaksu for petlje programskog jezika C:

```
for (( expr1 ; expr2 ; expr3 )) ;
do
    commands ;
done
```

U ovoj konstrukciji najpre se izračunava aritmetički izraz expr1 saglasno pravilima za aritmetiku komandnog interpretera, čime se obično postavljaju početni uslovi. Petlja funkcioniše na sledeći način: aritmetički izraz expr2 se izračunava u iteracijama sve dok ne postane 0. Svaki put kada je izraz expr2 različit od 0, izvršavaju se komande u for petlji a takođe se izračunava izraz expr3. Ako se bilo koji od tri izraza izostavi, for naredba ga tretira kao da ima vrednost 1. Izlazni status cele for petlje jednak je izlaznom statusu zadnje izvršene komande iz grupe commands, ili 0, ako je bilo koji od tri izraza pogrešno zadat (invalid).

Sledeći primer ilustruje upotrebu alternativne for petlje:

```
for i in `seq 1 10`
do
    echo $i
done
```

Naredba select

Naredba select omogućava jednostavnu konstrukciju menija. Sintaksa naredbe select je slična sintaksi for petlje:

```
select name [in words ...];
do
    commands;
done
```

Lista reči se proširuje generišući listu stavki (item). Skup proširenih reči prikazuje se na standardnom izlazu za greške, pri čemu svakoj prethodi redni broj. Ako se `in words' izostavi u naredbi select, ili ako se specificira `in "$@"', tada se prikazuju pozicioni parametri. U slučaju `in "$@"' PS3 prompt se prikazuje i linije se čitaju sa standardnog ulaza. Ako se linija sastoji od broja koji odgovara jednoj od prikazanih reči tada se vrednost promenljive name postavlja u tu reč. Ukoliko je linija prazna reči i prompt se prikazuju ponovo. Ako se pročita EOF select komanda završava rad. Svaka druga pročitana vrednost uzrokuje da promenljiva name bude postavljena na nulu. Pročitana linija se čuva u promenljivoj REPLAY.

Komande se izvršavaju posle svake selekcije sve dok se ne izvrši break komanda, čime se komanda select završava.

Primer ilustruje upotrebu naredbe select: program dozvoljava korisniku da sa tekućeg direktorijuma izabere datoteku čije će ime i indeks nakon toga biti prikazani.

```
select fname in *;
do
    echo Datoteka: $fname \($REPLY\)
    break;
done
```

Sledeći primer ilustruje kreiranje prostog menija:

```
opcije="Pozdrav Kraj"
select op in $opcije;
do
    if [ "$op" = "Kraj" ];
    then
        echo OK.
        exit
    elif [ "$op" = "Pozdrav" ];
        then
        echo Linux Rulez !
    else
        clear
        echo Opcija ne postoji.
    fi
 done
```

Funkcije

Pomoću funkcija komandnog interpretera komande se mogu grupisati u imenovanu celinu. Funkcije se izvršavaju kao i ostale komande - funkcija se poziva po imenu, kao i obična shell komanda, a lista komandi koja je pridružena funkciji se izvršava. Shell funkcije se izvršavaju u tekućem shell kontekstu.

Funkcije se deklarišu pomoću sledeće sintakse:

```
[ function ]
name ()
{
    command-list;
```

```
}
```

Ova konstrukcija definiše shell funkciju po imenu name. Rezervisana reč function je opciona, a ukoliko se navede opcione su srednje zagrade. Telo funkcije čini lista komandi između vitičastih zagrada { }, i izvršava se prilikom pozivanja funkcije. Vitičaste zagrade su rezervisane reči i shell ih prepoznaje samo ako su razdvojene praznim karakterima ili novim redovima od tela funkcije. Takođe, lista komandi command-list mora biti završena sa ; ili sa novim redom.

Prilikom izvršavanja argumenti funkcije postaju pozicioni parametri, specijalni parametar "#", koji opisuje broj pozicionih parametara, se ažurira, a ime funkcije se upisuje u promenljivu FUNCNAME. Pozicioni parametar 0 se ne menja.

Ako se povratak iz ugrađene komande izvrši u funkciji, funkcija se završava i izvršenje se nastavlja sa sledećom komandom iza funkcijskog poziva. Kada se funkcija izvrši, vrednosti pozicionih parametara i specijalni parametar # se vraćaju na vrednosti koje su imali pre izvršenja. Ako je izlazni argument specificiran, funkcija će vratiti taj argument, a u protivnom funkcija vraća status svoje poslednje izvršene komande.

Izlazni status funkcije jednak je izlaznom statusu poslednje izvršene komande u telu funkcije.

Funkcije mogu biti rekurzivne, pri čemu ne postoji limit na broj rekurzivnih poziva.

Lokalne promenljive

Lokalne varijable mogu se deklarisati unutar funkcije pomoću ključne reči local, i vidljive su samo unutar funkcije.

```
x=globalx
function myfunc {
    local x=localx
    echo $x
}
echo $x
myfunc
echo $x
```

Ovaj primer dovoljno jasno ilustruje korišćenje lokalnih promenljivih.

Primeri složenijih shell programa

Backup home direktorijuma. Promena imena grupe datoteka.

U nastavku teksta data su dva složenija primera shell programa.

Backup home direktorijuma

Program hbackup kreira backup home direktorijuma za jednog korisnika, čije se korisničko ime navodi kao parametar, ili za sve korisnike, ukoliko se kao parametar navede allusers. Backup se kreira pomoću programa tar i smešta u direktorijum /var/hbackup, ukoliko korisnik ne navede drugi direktorijum pomoću argumenta -d dest_dir. Ime backup datoteke formira se na osnovu imena korisnika čiji se backup kreira, i tekućeg datuma. Program zahteva bash za korektno izvršenje.

```
#!/bin/bash
# hbackup: backup home direktorijuma
if [ -z $1 ]; then
    echo "$0 [-d dest_dir] allusers (back-up home direktorijuma
svih korisnika)"
    echo "$0 [-d dest_dir] user (back-up home direktorijuma
korisnika user)"
    echo "$0: Ako se ne specificira direktorijum, koristi se
/var/hbackup"
    exit 0
fi

if [ "$1" = "-d" ]; then
    if [ -z $2 ]; then
        echo "$0: niste naveli putanju home direktorijuma"
        exit1
    fi
    hdir = $2
    if [ -z $3 ]; then
        echo "$0: niste naveli ime korisnika ili allusers"
        exit1
    elif [ "$3" = "allusers" ]; then
        hmode = allusers
    else
        hmode = oneuser
        hname = $3
    fi

else
    hdir = "/var/hbackup"
    if [ -z $1 ]; then
        echo "$0: niste naveli ime korisnika ili allusers"
        exit1
    elif [ "$1" = "allusers" ]; then
        hmode = allusers
    else
        hmode = oneuser
        hname = $1
    fi
fi

if  [ "$hmode" = "allusers" ]
    for hname in $( ls /home )
        do
        echo "Korisnik: $hname, datoteka: $hdir/$hname-$(date
+%Y%m%d).tar.gz"
```

```
        tar -czf  $hdir/$hname-$(date +%Y%m%d).tar.gz /home/$hname/
        exit 0
    done

else
    tar -czf  $hdir/$hname-$(date +%Y%m%d).tar.gz /home/$hname/
    exit 0
fi
```

Promena imena grupe datoteka

Program renamer vrši promenu imena datoteke ili grupe datoteka. Imena se mogu menjati dodavanjem prefiksa, sufiksa i zamenom niza karaktera u imenima datoteke. Program zahteva bash za korektno izvršenje.

```
#!/bin/bash
# renamer: promena imena grupe datoteka
if [ $1 = p ]; then
    prefix=$2 ; shift ; shift
    if [$1 = ]; then
        echo "$0: niste naveli ime datoteke(a)."
        exit 0
    fi
    for file in $*
        do
        mv ${file} $prefix$file
    done
    exit 0
fi

if [ $1 = s ]; then
    suffix=$2 ; shift ; shift
    if [$1 = ]; then
        echo "$0: niste naveli ime datoteke(a)."
        exit 0
    fi
    for file in $*
        do
        mv ${file} $file$suffix
    done
    exit 0
fi

if [ $1 = r ]; then
    shift
    if [ $# -lt 3 ] ; then
        echo "Sintaksa: $0 r [izraz] [zamena] datoteka(e) "
        exit 0
    fi
    OLD=$1 ; NEW=$2 ; shift ; shift
    for file in $*
        do
        new=`echo ${file} | sed s/${OLD}/${NEW}/g`
        mv ${file} $new
        done
```

```
    exit 0
fi

echo "Sintaksa:"
echo "$0 p [prefiks] datoteka(e)"
echo "$0 s [sufiks] datoteka(e)"
echo "$0 r [izraz] [zamena] datoteka(e)"
exit 0
```

7

MREŽNO OKRUŽENJE

Računarska mreža je skup računara relativno visokog stepena autonomije. Korisnicima umreženih računara dostupni su deljeni mrežni resursi poput štampača i direktorijuma, kao i mehanizmi centralizovane autentifikacije i administracije. UNIX je mrežni (networking) operativni sistem - integralni deo sistema čini TCP/IP skup protokola. Softver za umrežavanje, čiji su razvoj započeli programeri sa Berkeley univerziteta u Kaliforniji (grupa koja je realizovala BSD UNIX), omogućava funkcionalnost operativnog sistema u LAN i WAN mrežama. Na savremenim varijantama UNIX i Linux sistema prisutan je relativno potpun skup mrežnih alata, koji su dostupni regularnim korisnicima sistema i sistem administratorima. Većina osnovnih servisa Linux sistema, poput sistema datoteka, štampanja i arhiviranja podataka može se realizovati pomoću mrežnih funkcija operativnog sistema. U ovom poglavlju objašnjene su osnove umrežavanja, TCP/IP protokola i postupaka za korišćenje i administraciju mrežnih servisa.

Uvod u mreže i TCP/IP

Lokalne računarske mreže. TCP/IP skup protokola. IP adresiranje.

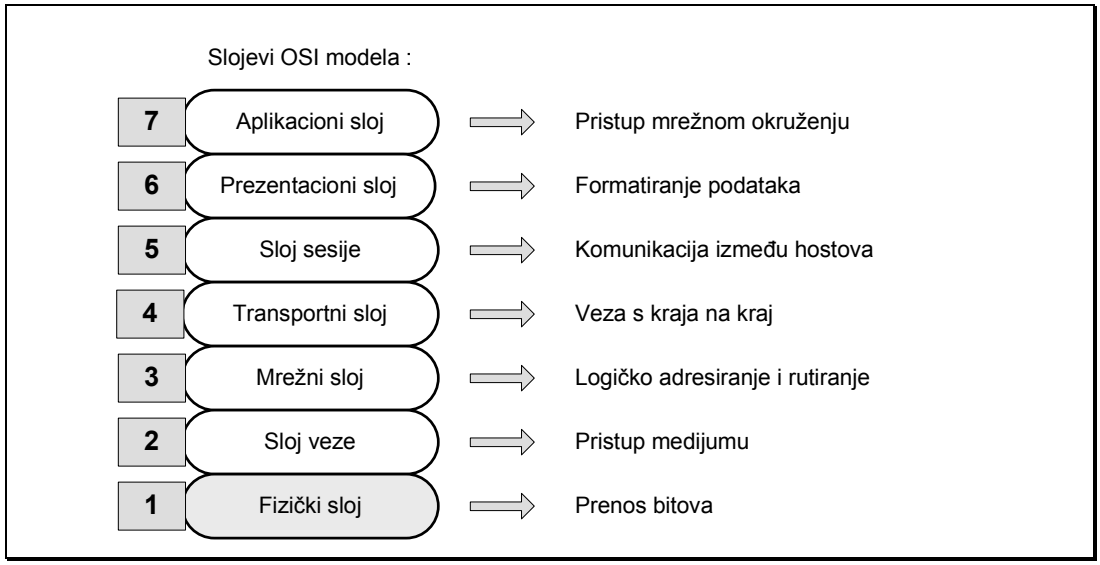
Računarska mreža omogućava komunikaciju međusobno povezanih autonomnih računara. Da bi se formirala računarska mreža potrebna su najmanje dva uređaja (radne stanice, štampači ili serveri) povezana bakarnim ili optičkim kablovima, ili bežičnom vezom, u cilju razmene informacija i/ili deljenja resursa. U resurse koji se mogu deliti ubrajaju se:

- izlazni uređaji (štampači, ploteri, ...),
- ulazni uređaji (skeneri, digitalne kamere, ...),
- masovni memorijski medijumi,
- modemi i Internet konekcije,
- podaci i aplikacije, u cilju efikasnog iskorišćenja prostora na diskovima i bolje saradnje na višekorisničkim projektima.

Prema veličini računarske mreže se mogu podeliti na: lokalne (Local area network - LAN), Mreže gradskog područja (Metropolitan area network - MAN) i WAN mreže (Wide area network). Lokalne računarske mreže povezuju računare (radne stanice i servere) i periferne uređaje na ograničenom geografskom području, poput jedne ili više zgrada. Maksimalna udaljenost uređaja je nekoliko hiljada metara. Koriste se u ustanovama poput manjih preduzeća, škola i državnih institucija. Za razliku od njih WAN mreže pokrivaju geografska područja veličine grada i države. Primer WAN mreže je Internet. Dve lokalne mreže u različitim delovima države ili sveta mogu komunicirati preko WAN-a. Korisnici se vezuju na WAN mreže preko lokalnih mreža.

Mrežno-komunikacioni zadaci su OSI referentnim modelom (Open Systems Interconnection) podeljeni na sedam manjih, lakše upravljivih celina (slojeva), od kojih svaki definiše određene funkcije mreže i može se zameniti drugim slojem istog nivoa, bez uticaja na ostatak sistema. Prednosti upotrebe OSI modela su smanjenje kompleksnosti, standardizacija interfejsa, sprečavanje uticaja promene jednog sloja na druge slojeve (čime je olakšan razvoj pojedinačnih funkcija) i omogućavanje lakšeg izbora pravog mrežnog uređaja za željenu namenu.

Slojevi OSI modela mogu se podeliti na niže i više slojeve. Za razumevanje principa rada lokalnih računarskih mreža i TCP/IP skupa protokola potrebno je detaljno poznavanje svih slojeva OSI modela (prikazani na slici 7.1), kao i mrežnih uređaja. Sa tačke gledišta administratora Linux sistema dovoljno je poznavati mrežni i transportni sloj, rutere i firewall uređaje.



Slika 7.1 *OSI referentni model*

Mrežni uređaji

Ukoliko sistemi nisu direktno povezani transmisionim medijumom koriste se posredni uređaji, koji pripadaju kategoriji otvorenih sistema i implementiraju niže slojeve OSI modela. U zavisnosti od slojeva koji su implementirani za te uređaje će od značaja biti bitovi, ramovi i paketi, ali ne i jedinice podataka viših slojeva. To određuje funkciju

uređaja: fizički sloj- repeater i hub, sloj veze- bridge i switch, mrežni sloj- ruter i layer 3 switch. S obzirom da nema nikakve modifikacije podataka viših slojeva postojanje ovih uređaja je za korisnika transparentno.

Za administratora Linux sistema u Linux/Windows mreži zanimljive su serverske funkcije (na primer web i mail). Linux takođe može da obavlja funkcije uređaja mrežnog sloja - rutiranje i filtriranje paketa.

Ruter je višeportni uređaj mrežnog sloja OSI modela čija je osnovna funkcija rutiranje paketa po mreži. Ruter posmatra mrežu u celini i na osnovu toga donosi odluke o najboljoj putanji za slanje paketa. Izbor putanje se svodi na izbor sledećeg skoka u mreži (hop). Ruter se može realizovati kao host računar u čiji su operativni sistem implementirani protokoli mrežnog sloja i odgovarajući softver.

Security gateway (firewall) uređaji predstavljaju širok opseg značajnih tehnologija, od jednostavnog filtriranja paketa na mrežnom sloju, do sofisticiranog filtriranja na sloju aplikacije. Filtriranje paketa je najjednostavniji oblik implementacije sigurnosti. Softver za rutiranje analizira izvorišnu i odredišnu adresu svakog paketa i broj pristupnog porta, te ga na osnovu unapred definisanih dozvola propušta ili odbija. Filtriranje na osnovu broja porta omogućava blokiranje određenih protokola, poput FTP i rlogin. Implementacija filtriranja najčešće se izvodi na već postojećem uređaju (ruteru), sposobnom da pruži određene usluge kontrole saobraćaja ili se u mrežu ugrađuje poseban firewall uređaj, ukoliko je pitanje sigurnosti kritičnije.

TCP/IP skup protokola

TCP/IP je familija protokola koju je razvila agencija DARPA (Defense Advanced Research Projects Agency), a koja je kasnije uključena u Berkeley Software Distribution of UNIX (BSD). Internet je zasnovan na TCP/IP familiji protokola, koja je de facto standard za povezivanje računara i mreža.

TCP/IP model čine sloj pristupa mreži, Internet sloj, transportni sloj i aplikacioni sloj.

TCP/IP model ne specificira sloj veze podataka i fizički sloj, već koristi različite protokole (SLIP, PPP) i tehnologije (Ethernet) na tom sloju. SLIP (Serial Line Internet Protocol) je adaptacija TCP/IP steka za računare koji su na mrežu povezani preko serijskog porta. Smatra se zastarelim i zamenjen je PPP protokolom. PPP (Point-to-Point Tunelling Protocol) je adaptacija TCP/IP steka za računare koji su na mrežu povezani modemom (dial-up veza). PPP koristi kompresiju podataka u cilju povećanja propusnog opsega.

Internet sloj odgovara sloju mreže u OSI modelu. Bavi se IP adresiranjem i rutiranjem paketa, čime obezbeđuje vezu između računara koji se ne moraju nalaziti na fizički istoj mreži. Na ovom sloju prisutni su sledeći protokoli:

- IP (Internet Protocol) - fundamentalan protokol koji obezbeđuje transfer informacija od računara do računara;
- ICMP (Internet Control Message Protocol) - protokol koji se u steku nalazi iznad IP protokola i obezbeđuje kontolne poruke IP protokolu, kao što je "Host or Network

Unreachable". Najčešća upotreba ICMP protokola je slanje ICMP ECHO paketa (ping) kojim se proverava da li je host računar dostupan;

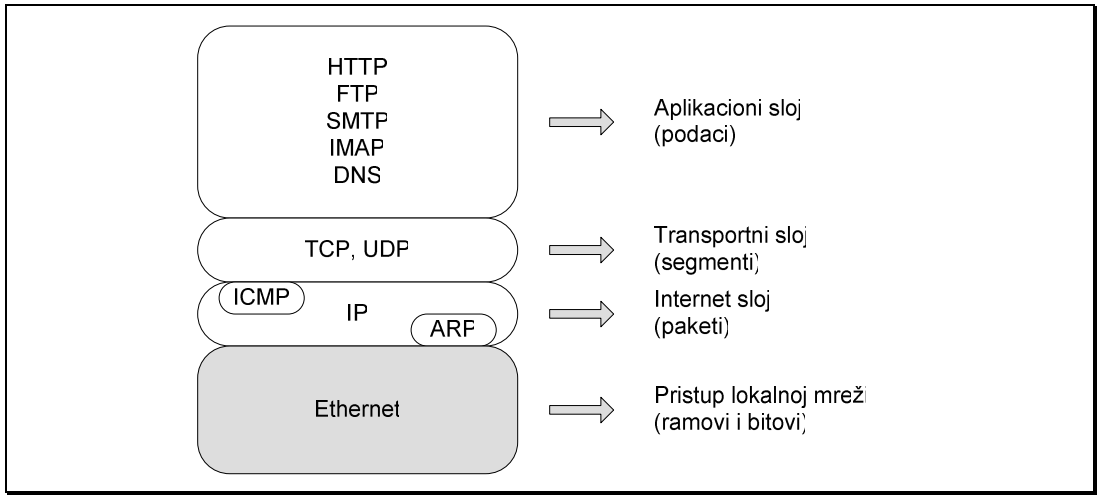
- ARP (Adress Resolution Protocol) - osim logičke adrese (IP), svaki mrežni uređaj se karakteriše i fizičkom adresom (MAC) dužine 48 bita. MAC adrese dodeljuju proizvođači mrežnih adaptera i one su relativno nepromenljive. MAC adrese se koriste prilikom transporta podataka, odnosno ramova po fizički istoj mreži. ARP protokol razrešava IP adrese u MAC adrese. RARP je inverzan protokol ARP-u i pomoću njega se vrši određivanje IP adrese hosta na osnovu fizičke adrese;

Transportni sloj preuzima podatke sa višeg nivoa, po potrebi vrši segmentaciju podataka u datagrame ili uspostavljanje virtuelnih veza i prenosi podatke do destinacije koristeći mrežni sloj. Na transportnom sloju prisutni su:

- TCP (Transmission Control Protocol) - protokol koji obezbeđuje pouzdanu vezu između dva procesa, otkriva i ispravlja greške,
- UDP (User Datagram Protocol) - protokol koji ne uspostavlja virtuelne veze niti obezbeđuje mehanizam za detekciju grešaka.

Aplikacioni sloj omogućava aplikacijama, odnosno korisnicima da pristupe servisima Internet mreže. Na aplikacionom sloju između ostalih prisutni su i sledeći protokoli:

- HTTP (HyperText Transport Protocol) - pristup Web stranicama
- FTP (File Transport Protocol) - transfer datoteka
- SMTP (Simple Mail Transport Protocol) - dolazeća pošta
- POP3 (Post Office Protocol v3) - odlazeća pošta
- DNS (Domain Name System) - razrešavanje imena u IP adrese



Slika 7.2 *TCP/IP model*

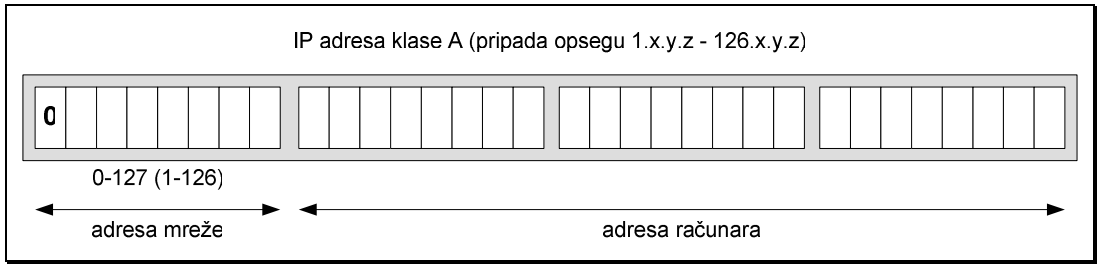
IP adresiranje

Svaki računar i ruter na Internetu ima svoju jedinstvenu IP adresu (ili više IP adresa). IP adrese su 32-bitne, sastoje se od 4 okteta i obično se predstavljaju u decimalnoj notaciji sa tačkom (na primer: 192.198.3.1). Svaka IP adresa ima dva dela:

- deo koji predstavlja adresu IP mreže, koji je isti za sve računare na jednoj IP mreži,
- deo koji predstavlja adresu računara, koji je jedinstven za svaki računar na istoj IP mreži.

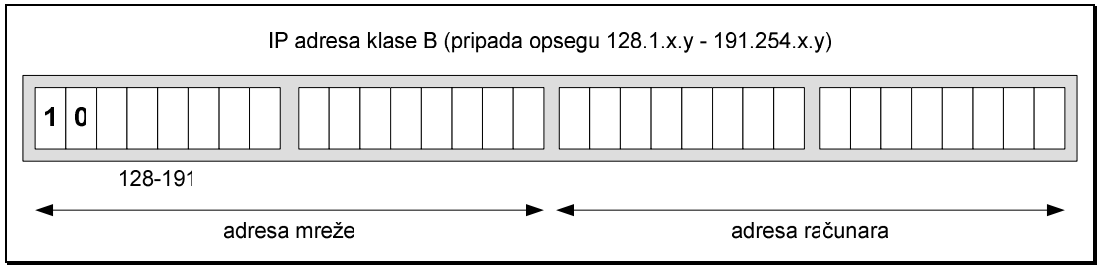
Na osnovu broja okteta koji pripadaju adresi mreže, odnosno adresi hosta, IP adrese se dele u klase A, B, C, D i E.

U binarnom obliku IP adrese klase A počinju sa 0. Prvi oktet predstavlja adresu mreže, a sledeća tri okteta adresu računara. Kako su adrese 0.x.y.z i 127.x.y.z rezervisane, IP adrese klase A se nalaze u opsegu 1.x.y.z do 126.x.y.z. Dodeljuju se mrežama sa ogromnim brojem hostova (2^{24}=16777214 hosta po jednoj IP mreži). U praksi, IP mreže klase A se dele na podmreže zbog velikog broadcast domena.



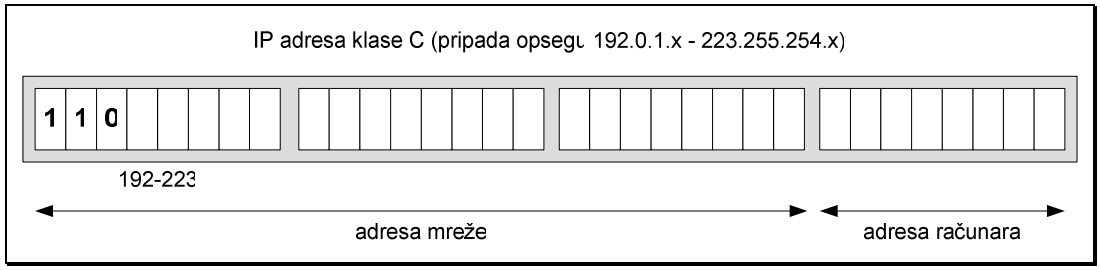
Slika 7.3 *IP adresa klase A*

U binarnom obliku IP adrese klase B počinju sa 10. Prva dva okteta pretstavljaju adresu mreže, a sledeća dva adresu računara. IP adrese klase B nalaze se u opsegu 128.1.x.y do 191.254.x.y. Ovim adresama se mogu formirati 16382 mreže sa relativno velikim brojem hostova (65534 hosta po jednoj IP mreži). Većina univerziteta i većih kompanija koriste IP mreže u klasi B, koje su često podeljene na nekoliko podmreža.



Slika 7.4 *IP adresa klase B*

U binarnom obliku IP adrese klase C počinju sa 110. Prva tri okteta predstavljaju adresu mreže, a poslednji adresu računara. IP adrese klase C nalaze se u opsegu 192.0.1.x do 223.255.254.x. Ovim adresama se može formirati veliki broj IP mreža (2097150) sa malim brojem hostova (245 hosta po mreži).



Slika 7.5 *IP adresa klase C*

IP adrese klase D (u binarnom obliku počinju sa 1110) se dodeljuju grupi računara i namenjene su za multicast saobraćaj. IP adrese klase E (počinju sa 11110) su eksperimentalne i ne koriste se za adresiranje mreža i računara.

U posebne slučajeve IP adresa spadaju:

127.0.0.1	adresa lokalne petlje (local loopback adress)
0.0.0.0	obično označava da host još nije konfigurisan sa IP adresom
255.255.255.255	broadcast poruka namenjena svim računarima na lokalnoj mreži
x.255.255.255	broadcast poruka namenjena mreži klase A čija je adresa x.0.0.0
x.y.255.255	broadcast poruka namenjena mreži klase B čija je adresa x.y.0.0
x.y.z.255	broadcast poruka namenjena mreži klase C čija je adresa x.y.z.0
x.0.0.0	adresa mreže u klasi A
x.y.0.0	adresa mreže u klasi B
x.y.z.0	adresa mreže u klasi C

IP adrese se dalje mogu podeliti na javne i privatne. Javne adrese dodeljuje InterNIC i mogu se koristiti na Internetu. Privatne adrese (10.0.0.0, 172.16.0.0 i 192.168.0.0) su namenjene mrežama koje nisu direktno povezane na Internet i ne mogu se koristiti na Internetu.

Maska podmreže je 32-bitni broj koji se formira tako što se umesto bitova koji u IP adresi pretstavljaju adresu mreže i podmreže stavi 1, a umesto bitova koji predstavljaju adresu host računara stavi 0. Podrazumevane maske podmreže za mreže u klasi A, B i C su redom 255.0.0.0, 255.255.0.0 i 255.255.255.0.

Podmreže su segmenti iste IP mreže odvojeni ruterima. Podmrežavanjem se smanjuje broj hostova po segmentu, odnosno broadcast domen IP mreža sa velikim brojem hostova. Svaka podmreža ima svoj jedinstven identifikator, koji se formira tako što se određen broj bitova pozajmi iz dela IP adrese koji predstavlja adresu računara. Za ostatak Interneta, IP mreža podeljena na podmreže je još uvek jedna mreža, tako da nema potrebe za izmenom tabela rutiranja.

Podmrežavanje se može jednostavno objasniti na primeru IP mreže u klasi B. Za kreiranje podmreža može se pozajmiti deo okteta ili ceo oktet. U ovom slučaju pozajmljuje se ceo treći oktet. Ukoliko je adresa IP mreže 166.60.0.0, adrese podmreža će redom biti:

166.60.1.0, 166.60.2.0, do 166.60.255.0. Na ovaj način je kreirano 256 segmenata sa po 245 host računara na svakom.

Rutiranje

Prava upotrebna vrednost IP mreže iskazuje se njenom mogućnošću da kontaktira druge IP mreže. Da bi se paket poslao hostu koji se nalazi na drugoj mreži, potrebno je da u mreži postoji uređaj koji zna kako i gde isporučiti paket. Ovaj oblik isporuke paketa je poznat kao rutiranje. U IP svaka mreža čuva informacije samo o prvim skokovima ka nodovima koji predstavljaju vezu sa ostatkom sveta. Ovi nodovi znaju sledeći skok paketa, tako da se paket prenosi do odredišta putem skokova u mreži (hop).

Postoji nekoliko tipova rutiranja:

- standardno rutiranje - svi paketi koji nisu namenjeni mreži šalju se na podrazumevani izlaz. Ovaj tip rutiranja je primenljiv ukoliko mreža ima samo jedan izlaz;
- statičko rutiranje - pomoću određene komande formiraju se statičke rute u tabeli. Ovaj način rutiranja upotrebljava se ukoliko iz mreže postoji nekoliko izlaza ka drugim mrežama, pri čemu se jedan koristi kao podrazumevani izlaz ka svim ostalim mrežama;
- dinamičko rutiranje - sistem “osluškuje” broadcast pakete rutiranje i automatski podešava tabele. Mnogi Internet ruteri koriste ovaj metod rutiranja.

Linux može obavljati funkciju mrežnog rutera. Rutiranje obavljaju razni daemon programi kao što su routed (Route Daemon) i mrouted (Multicast Route Daemon).

Broj porta

Osim IP adrese protokoli koriste i broj porta, odnosno 16 bitni kvantitet koji dozvoljava više istovremenih konekcija na jednom čvoru. Svaka konekcija se odnosi na određeni port, koji se dodeljuje određenom mrežnom servisu na korišćenje. Korisnici mogu otvoriti ili zatvoriti određeni port, čime se omogućava, odnosno onemogućava uspostavljanje konekcije ka nekom tipu servisa. Portovi čiji je broj manji od 1024 smatraju se povlašćenim i za njihovo otvaranje potrebne su privilegije koje ima root.

Razrešavanje imena računara

Korisnici se svakom računaru mogu obratiti putem IP adrese. To naravno znači da korisnici koji koriste servise 150 različitih računara moraju znati 150 IP adresa. Da bi se komunikacija pojednostavila koristi se sistem dodele logičkih imena IP adresama. Na primer, korisnici se mogu obratiti računaru čija je IP adresa 166.60.10.15 imenom nicotine, ukoliko je ime nicotine dodeljeno toj IP adresi. Linux sistemi koriste sledeće metode za razrešavanje imena računara:

- razrešavanje imena na osnovu datoteka u /etc/direktorijumu (/etc/hosts, /etc/networks),
- razrešavanje imena pomoću NIS servisa (Network Information Service),

- razrešavanje imena pomoću DNS servera (Domain Name System).

Mrežni servisi

Mrežni servisi (daemons) prihvataju zahteve za uspostavljanje konekcije na određenom portu. Imena servisa su određena datotekom /etc/hosts, koja povezuje servis sa odgovarajućim protokolom (TCP ili UDP) i brojem porta. Funkcionisanje većine servisa je pod kontrolom wrapper daemon programa, kao što su inetd (Internet dispatch Daemon) i xinetd.

Na Linux sistemima postoji veliki broj mrežnih servisa koji pružaju različite usluge korisnicima sistema:

- otvaranje udaljene interaktivne sesije (telnet). Zahtevi za otvaranjem telnet sesije šalju se na TCP port 23, a prihvataju ih telnetd ili odgovarajući wrapper programi (inetd ili xinetd);
- transfer datoteka (ftp - file transfer protocol). Zahtevi za otvaranjem ftp sesije šalju se na TCP port 20, a prihvataju ih ftpd ili odgovarajući wrapper programi;
- zahtev za razrešavanjem imena (DNS) - šalje se na UDP port 53;
- pristup Web stranicama preko http protokola (servis obezbeđuje hppd iz Apache paketa);
- pristup elektronskoj pošti (servis obezbeđuju Sendmail i Postfix);
- pristup mrežnom sistemu datoteka (NFS);
- centralizovanu autentifikaciju (NIS).

Konfigurisanje Linux mrežnog okruženja

Konfiguracione datoteke i programi za administraciju TCP/IP skupa protokola. Mrežni servisi.

Nakon povezivanja na mrežu potrebno je konfigurisati mrežno okruženje Linux sistema. Većina Linux distribucija automatski prilagođava sistem mreži nakon instalacije. Ukoliko to nije slučaj, za konfigurisanje se mogu koristiti razni programi (netconfig u Slackware distribuciji, ili LinuxConf u Red Hat distribuciji, koji radi u grafičkom okruženju). Princip funkcionisanja ovih programa je jednostavan - korisniku se postavljaju razna pitanja, a na osnovu odgovora popunjavaju se konfiguracione datoteke. Ovi programi se isto tako mogu koristiti za izmenu postojeće TCP/IP konfiguracije.

Konfiguracione datoteke

Korisnici takođe mogu konfigurisati mrežno okruženje ručnom izmenom sadržaja konfiguracionih datoteka. Inicijalne konfiguracione datoteke kreiraju se u direktorijumu

/etc nakon instalacije sistema i u većini slučajeva su dovoljne za rad u mreži (posmatrano sa tačke gledišta klijenta). Ukoliko računar obavlja funkciju mrežnog servera, najčešće postoji potreba za dodatnom konfiguracijom mrežnog okruženja. Kako se na serverima grafičko okruženje najčešće ne instalira (smanjuje se opterećenje sistema), od alata za konfigurisanje dostupni su samo alati koji rade u tekstualnom režimu i editori teksta. U nastavku teksta dat je kratak opis značajnijih konfiguracionih datoteka.

/etc/hostname

U datotekci /etc/hostname upisano je ime računara. Datoteka sadrži jednu liniju u kojoj je upisano ime računara.

```
tulip
```

/etc/hosts

Tabela u kojoj su opisana imena računara na UNIX sistemima čuva se u datoteci /etc/hosts. U ovoj tabeli su osim imena računara opisana i sva imena dodeljena odgovarajućim IP adresama (na primer imena mreža, podmreža, maski podmreže i broadcast adresa):

127.0.0.1	localhost (lokalna petlja)
172.16.48.10	tulip (računar)
172.16.45.75	nicotine (računar)
172.16.0.0	mynetwork (mreža)
172.16.255.255	mybroadcast (broadcast adresa)
255.255.0.0	mynetmask (maska podmreže)

Ova imena mogu menjati IP adrese u svim komandama koje kao parametar zahtevaju IP adresu. Računar će razrešiti ime u IP adresu korišćenjem ove datoteke. Ovaj metod razrešavanja IP adresa zahteva da korisnik, odnosno administrator sistema redovno osvežava datoteku /etc/hosts (na primer, preuzimanjem datoteke sa Interneta i ručnim dodavanjem značajnih IP adresa).

U nastavku teksta dat je kratak isečak datoteke /etc/hosts. U datoteci su odgovarajućim IP adresama dodeljena imena (jedno ili više). Datoteka je realizovana u vidu tabele u kojoj prva kolona predstavlja IP adresu, a ostale imena koja su dodeljena tim IP adresama. Svaki red u tabeli predstavlja jednu IP adresu. Linije koje počinju znakom # smatraju se komentarima i ignorišu se.

```
# Internet host table
# IP-address name nicnames ...
#
127.0.0.1          localhost
172.16.48.10       tulip         server1
172.16.45.75       nicotine      workstation54
172.16.0.0         labnet        mynetwork
172.16.255.255     mybroadcast
```

```
255.255.0.0        mynetmask
```

Napomena: jedna IP adresa može se pojaviti samo u jednoj liniji. Ukoliko se jedna IP adresa navede u više linija, može doći do konfuzije ukoliko se na računaru koristi NIS servis za autentifikaciju.

/etc/hosts.allow i /etc/hosts.deny

Datotekama /etc/hosts.allow i /etc/hosts.deny određuje se kojim računarima je pristup sistemu na kom se te datoteke nalaze dozvoljen, odnosno zabranjen. Datoteke su realizovane u vidu liste, pri čemu se u jednoj liniji navodi ime jednog računara, domena ili ALL-EXCEPT konstrukcija. Na primer, sledeća datoteka /etc/hosts.allow dozvoljava pristup svim računarima domena mydomain i svim računarima domena theirdomain.com osim računaru hack.

```
ALL: .mydomain.com
ALL: .theirdomain.com EXCEPT hack.theirdomain.com
```

Slično, datotekom /etc/hosts.deny zabranjuje se pristup računarima. Na primer sledeća datoteka eksplicitno zabranjuje pristup računaru hack iz domena pwdcrack.com i svim računarima iz domena hackheaven.org.

```
hack.pwdcrack.com
ALL: .hackheaven.org
```

/etc/networks

Slično datoteci /etc/hosts, na Linux sistemima postoji i datoteka /etc/networks u kojoj se čuva tabela u kojoj su opisana imena mreža. U nastavku teksta dat je kratak isečak datoteke /etc/networks. Datoteka je realizovana u vidu tabele u kojoj prva kolona predstavlja IP adresu mreže, a ostale imena koja su dodeljena tim IP adresama. Svaki red u tabeli predstavlja jednu IP adresu.

```
localnet      172.16.0.0
hackheaven    166.60.0.0
```

/etc/network/interfaces

Konfiguraciona datoteka koju koriste ifup i ifdown skriptovi. Ovu datoteku kreira sam Linux instalacioni program i u njoj su opisani mrežni adapteri prisutni na sistemu. Datoteka se najčešće ne menja ručno, osim ukoliko se na sistem ne doda još jedan mrežni adapter.

```
# The loopback interface
auto lo
iface lo inet loopback

# The first network card
auto eth0
iface eth0 inet static
        address   172.16.48.10
```

```
        netmask   255.255.0.0
        network   172.16.0.0
        broadcast 172.16.255.255
```

/etc/protocols

Prilikom prikazivanja poruka na ekranu razni dijagnostički alati koriste ovu datoteku radi prevođenja broja protokola u simboličko ime. Ovu datoteku obezbeđuje proizvođač konkretne distribucije Linux sistema i njen sadržaj se najčešće ne menja. Datoteka je univerzalna - u njoj je opisan veliki broj protokola, od kojih svaki sistem podržava određeni podskup protokola. U nastavku teksta dat je kraći isečak datoteke /etc/protocols:

```
# Internet (IP) protocols
#
ip      0       IP              # internet protocol, pseudo
                                # protocol number
icmp    1       ICMP            # internet control message protocol
igmp    2       IGMP            # Internet Group Management
tcp     6       TCP             # transmission control protocol
udp     17      UDP             # user datagram protocol
ospf    89      OSPFIGP         # Open Shortest Path First IGP
```

/etc/services

Proces na klijentu uspostavlja konekciju sa procesom na serveru preko određenog porta. Port je dodeljen određenom mrežnom servisu na korišćenje. Na klijentima i serverima postoji datoteka /etc/services u kojoj su opisani servisi i portovi na kojima ti servisi pružaju usluge klijentima. Ova datoteka je jako velika i kompletna i u nastavku je dat njen kraći isečak:

```
# Network services, Internet style
# sevice-name port/protocol nicknames...
#
echo        7/tcp
echo        7/udp
ftp         21/tcp
ssh         22/tcp              # SSH Remote Login Protocol
ssh         22/udp              # SSH Remote Login Protocol
telnet      23/tcp
smtp        25/tcp      mail
www         80/tcp      http    # WorldWideWeb HTTP
www         80/udp              # HyperText Transfer Protocol
pop3        110/tcp     pop-3   # POP version 3
pop3        110/udp     pop-3
irc         194/tcp             # Internet Relay Chat
irc         194/udp
imap3       220/tcp             # Interactive Mail Access
imap3       220/udp             # Protocol v3
```

Datoteka /etc/services je realizovana u formi tabele: prva kolona predstavlja ime servisa, druga kolona broj porta i protokol (TCP ili UDP) razdvojene slash karakterom. Nakon toga se navode alternativna imena servisa, i eventualno komentar. Jedna linija opisuje

jedan servis, osim u slučajevima servisa koji su dostupni i preko UDP i preko TCP protokola, pa se u datoteci navode u dve linije.

/etc/resolve.conf

Ukoliko se za razrešavanje imena koristi DNS, računari (DNS klijenti) šalju upit DNS serveru, odnosno računaru na kom je pokrenut DNS servis. DNS server razrešava ime u IP adresu na osnovnu tabela i šalje odgovor klijentu. DNS predstavlja hijerarhijsku distribuiranu bazu podataka koja omogućava razrešavanje imena računara u IP adresu. DNS je hijerarhijski u smislu organizacije imena računara u logičku strukturu stabla, u kom je svaki čvor jedinstveno identifikovan na osnovu svog potpuno kvalifikovanog imena (Fully qualified domain name). Grane stabla predstavljaju DNS domene u kojima se mogu nalaziti računari i poddomeni. Svaki čvor u stablu ima i svoje kratko ime (labelu), koja je jedinstvena samo u svom domenu, ali ne i globalno. Puno kvalifikovano ime se dobija navođenjem labele čvora i svih ostalih labela do korena, razdvojenih tačkama (na primer, nicotine.kyuss.org). DNS je distribuiran u smislu da održavanje baze nije centralizovano - razrešavanje se ne vrši samo na jednom serveru, već na velikoj grupi servera, od kojih je svaki zadužen za razrešavanje imena u svojoj zoni (delegirana odgovornost). Zona je deo DNS stabla (domen, i opciono deo poddomena ili svi poddomeni) koji se čuvaju na jednom serveru i za koji je odgovorna jedna organizacija.

Ime domena i IP adresa DNS servera specificiraju se u datoteci /etc/resolve.conf. U nastavku teksta dat je primer datoteke /etc/resolve.conf.

```
search mydomain.co.yu
nameserver 172.16.48.1
```

Na nekim starijim UNIX sistemima postojanje ove datoteke eksplicitno povlači i upotrebu DNS servisa za razrešavanje imena. Na novijim sistemima potrebno je specificirati metod (ili više metoda) koji se koristi za razrešavanje imena. Takođe, potrebno je specificirati i metode koje se koriste za proveru lozinki korisnika, određivanje članstva u grupama, itd.

/etc/nssswitch.conf - konfigurisanje metoda

Sadržajem datoteke /etc/nsswitch.conf (Name Service Switch) određene su metode koje se koriste za:

- proveru lozinki (uključujući i shadow datoteku),
- određivanje članstva u grupama,
- specificiranje auto-mount sistema datoteka,
- razrešavanje imena u IP adrese,
- prevođenje MAC adresa u imena računara,
- dodeljivanje simboličkih imena protokolima,
- povezivanje servisa sa određenim brojem porta,
- proveru javnih ključeva.

Datoteka /etc/nsswitch.conf dozvoljava primenu sledećih metoda:

files (compat, db)	upotreba lokalnih konfiguracionih datoteka
nis (yp)	upotreba NIS servisa (NIS version 2), koji je poznat pod imenom Yellow Pages (YP)
nisplus (nis+)	upotreba NIS+ servisa (NIS version 3)
dns	upotreba DNS servisa (samo za razrešavanje imena)

Sledeća linija u datoteci /etc/nsswitch.conf znači da se provera lozinki najpre vrši na osnovu lokalne datoteke /etc/passwd. Ukoliko korisnik nije specificiran u lokalnoj datoteci, proverava se da li korisnik postoji u bazi na NIS+ serveru.

```
passwd:         files nisplus
```

Sledeća linija u datoteci /etc/nsswitch.conf specificira da se razrešavanje imena vrši isključivo preko DNS servera, čija je IP adresa navedena u datoteci /etc/resolve.conf.

```
hosts:          files dns
```

Ukoliko datoteka nsswitch.conf ne postoji koriste se podrazumevane vrednosti:

```
passwd:         compat
group:          compat
shadow:         compat
hosts:          dns [!UNAVAIL=return] files
networks:       nis [NOTFOUND=return] files
ethers:         nis [NOTFOUND=return] files
protocols:      nis [NOTFOUND=return] files
rpc:            nis [NOTFOUND=return] files
services:       nis [NOTFOUND=return] files
```

Podrazumevane vrednosti imaju sledeće značenje: provera lozinki i određivanje članstva u grupama isključivo se vrše putem lokalnih konfiguracionih datoteka. Imena računara se razrešavaju preko DNS servera, a ukoliko on ne postoji u mreži ili ne može da razreši ime u IP adresu, koristi se lokalna datoteka /etc/hosts. Za dodelu imena protokolima, vezivanje servisa za portove i dodelu imena mrežama koristi se NIS server, a ukoliko on ne postoji u mreži, sve se razrešava lokalnim datotekama.

Programi za TCP/IP administraciju

Kao i svi ostali aspekti računarskog sistema, mreža nije imuna na greške. Kako mreža predstavlja skup većeg broja računara, otklanjanje problema vezanih za mrežu je komplikovanije od otklanjanja problema vezanih za ostale aspekte računarskog sistema. Osnovna pitanja koja administrator treba da postavi ukoliko mreža ne funkcioniše ispravno su:

- da li je mreža ispravno konfigurisana (komandom ifconfig se utvrđuje da li su mrežni interfejsi konfigurisani i aktivirani),
- da li je rutiranje ispravno konfigurisano (utvrđuje se komandom netstat),

- da li računar može da komunicira sa ostalim čvorovima u mreži (utvrđuje se komandama ping i traceroute).

/sbin/ifdown i /sbin/ifup

Koriste se za zaustavljanje i pokretanje mrežnog adaptera specificiranog argumentom interface u sledećoj sintaksi:

```
$ ifup [-a|interface]
$ ifdown [-a|interface]
```

Ukoliko se navede parametar -a, zaustavljaju se, odnosno pokreću svi mrežni adapteri.

ifconfig

Komanda ifconfig koristi se za konfigurisanje mrežnih interfejsa rezidentnih u kernelu. Komandom ifconfig mogu se postaviti parametri poput IP adrese, maske podmreže i broadcast adrese, a takođe se može prikazati i trenutna konfiguracija mrežnog interfejsa, kao i MAC adresa mrežne kartice. Komanda ifconfig koristi se prilikom podizanja sistema radi postavljanja parametara mrežnog interfejsa. Nakon toga se najčešće koristi u dijagnostičke svrhe.

Ukoliko se ifconfig zada bez parametara na ekranu se prikazuje status svih aktivnih interfejsa (uključujući i loopback):

```
# ifconfig
eth0
    Link encap:Ethernet  HWaddr 00:60:97:BA:B9:00
    inet addr:172.16.48.10  Bcast:172.16.255.255  Mask:255.255.0.0
    UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
    RX packets:677062 errors:0 dropped:0 overruns:0 frame:0
    TX packets:628851 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:78879099 (75.2 MiB)  TX bytes:369619770 (352.4 MiB)
    Interrupt:17 Base address:0xf0c0
lo
    Link encap:Local Loopback
    inet addr:127.0.0.1  Mask:255.0.0.0
    UP LOOPBACK RUNNING  MTU:16436  Metric:1
    RX packets:309 errors:0 dropped:0 overruns:0 frame:0
    TX packets:309 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:0
    RX bytes:21832 (21.3 KiB)  TX bytes:21832 (21.3 KiB)
```

Kao argument može se navesti ime interfejsa (interface), i u tom slučaju na ekranu se prikazuje status tog interfejsa. Ukoliko se umesto imena interfejsa navede parametar -a, komanda će prikazati status svih interfejsa, uključujući i neaktivne.

```
$ ifconfig interface
```

U ostalim oblicima, ifconfig se koristi za konfigurisanje mrežnog interfejsa:

```
$ ifconfig interface [aftype] options | address ...
```

Ime mrežnog interfejsa (interface) najčešće se formira na osnovu imena drajvera i rednog broja interfejsa (eth0 je prvi Eternet interfejs).

Opcioni argument aftype (adress family type) nakon imena interfejsa određuje tip adrese mrežnog sloja. Podržani su sledeći tipovi adresa:

inet	TCP/IP, podrazumevana vrednost ukoliko se tip adrese ne navede
inet6	IPv6
ipx	Novell IPX

Nakon toga se navode parametri mrežnog interfejsa:

up	aktiviranje mrežnog interfejsa. Izvršava se automatski ukoliko se interfejsu dodeli adresa
down	deaktiviranje mrežnog interfejsa
[-]arp	aktiviranje/deaktiviranje ARP protokola za mrežni interfejs
netmask	dodela maske podmreže
adress	dodela adrese mrežnom interfejsu

netstat

Netstat je glavni dijagnostički alat pomoću kog administrator dobija izveštaj o mrežnom interfejsu, tabelama rutiranja, mrežnim konekcijama i statistici korišćenja TCP/IP skupa protokola.

Izveštaj o mrežnom interfejsu obuhvata informacije o korišćenju mreže i broju kolizija:

```
$ netstat –i
Kernel Interface table
Iface   MTU Met   RX-OK RX-ERR RX-DRP RX-OVR   TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0   1500   0  680355      0      0      0  630404      0      0      0 BMRU
lo    16436   0     309      0      0      0     309      0      0      0 LRU
```

Na osnovu ovog izveštaja može se zaključiti:

- da li računar komunicira sa mrežom (računar komunicira sa mrežom ukoliko vrednosti RX-OK, odnosno broj primljenih paketa i TX-OK, odnosno broj poslatih paketa rastu),
- da li postoji greška u kabliranju, odnosno u pasivnoj mrežnoj opremi (vrednosti RX-ERR i TX-ERR rastu).
- da li je mreža preopterećena (ukoliko je broj kolizija, odnosno vrednosti RX-DRP i TX-DRP reda veličine 1-2% ukupnog broja paketa, mreža nije opterećena).

Izveštaj o rutiranju prikazuje kako je rutiranje konfigurisano. Komanda netstat -r je ekvivalentna komandi route.

```
# netstat –r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
localnet        *               255.255.0.0     U         0 0          0 eth0
```

Na osnovu izveštaja (tabele rutiranja) određuje se prisutnost rute i prateći flegovi, koji prikazuju status i način kreiranja rute. Detaljnije informacije o značenju flegova dostupne su u pratećem uputstvu komande (man netstat).

Komandom netstat takođe se može prikazati izveštaj o aktivnim TCP konekcijama:

```
# netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:1024                  *:*                     LISTEN
tcp        0      0 *:printer               *:*                     LISTEN
tcp        0      0 *:swat                  *:*                     LISTEN
..
..
tcp        0      0 *:smtp                  *:*                     LISTEN
tcp        0      0 tulip.internal.vets:ssh nicotine.internal.:1131 ESTABLISHED
tcp        0      0 tulip.inter:netbios-ssn rs33.internal.vets:1386 ESTABLISHED
udp        0      0 *:1024                  *:*
udp        0      0 localhost:1025          localhost:1025          ESTABLISHED
udp        0      0 tulip.intern:netbios-ns *:*
..
..
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ACC ]     STREAM     LISTENING     7640   private/lmtp
unix  2      [ ACC ]     STREAM     LISTENING     7644   private/cyrus
unix  2      [ ACC ]     STREAM     LISTENING     7648   private/uucp
..
..
unix  2      [ ]         DGRAM                    509
unix  2      [ ]         DGRAM                    385
```

Statistički izveštaj o TCP/IP skupu protokola obično sadrži sledeće informacije:

- broj primljenih, prosleđenih i odbačenih IP paketa,
- broj primljenih i odbačenih ICMP paketa,
- izveštaj o TCP konekcijama,
- broj primljenih i poslatih UDP paketa.

Upoređivanjem broja grešaka u različitim vremenskim trenucima administratori mogu odrediti prosečno opterećenje mrežnog servera. Primer ilustruje isečak statističkog izveštaja:

```
$ netstat -s
Ip:
    523055 total packets received
    0 forwarded
    0 incoming packets discarded
    512159 incoming packets delivered
    592678 requests sent out
Icmp:
    346 ICMP messages received
    0 input ICMP message failed.
    ICMP input histogram:
        destination unreachable: 269
```

```
        echo requests: 77
    375 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 298
        echo replies: 77
Tcp:
    139 active connections openings
    87 passive connection openings
    0 failed connection attempts
    32 connection resets received
    2 connections established
    290830 segments received
    471696 segments send out
    375 segments retransmited
    0 bad segments received.
    6917 resets sent
Udp:
    225366 packets received
    37 packets to unknown port received.
    0 packet receive errors
    120607 packets sent
...
...
```

arp

Komanda arp prikazuje keširanu arp tabelu, odnosno imena računara na lokalnoj mreži i odgovarajuće MAC adrese. Ukoliko se navede opcija -a, izveštaj se prikazuje u alternativnom (BSD) formatu.

```
# arp
netfinity.mydomain.com   ether   00:60:94:19:C5:08   C     eth0
nicotine.mydomain.com    ether   00:40:C7:7B:18:B7   C     eth0
valus.mydomain.com       ether   00:04:AC:15:D1:AC   C     eth0
ws22.mydomain.com        ether   00:09:6B:05:8D:3A   C     eth0
ws61.mydomain.com        ether   00:0D:60:67:20:07   C     eth0
...
...
# arp -a
netfinity.mydomain.com (172.16.32.100) at 00:60:94:19:C5:08 [ether] on eth0
nicotine.mydomain.com (172.16.40.75) at 00:40:C7:7B:18:B7 [ether] on eth0
valus.mydomain.com (172.16.0.1) at 00:04:AC:15:D1:AC [ether] on eth0
ws22.mydomain.com (172.16.40.133) at 00:09:6B:05:8D:3A [ether] on eth0
ws61.mydomain.com (172.16.40.232) at 00:0D:60:67:20:07 [ether] on eth0
...
...
```

ping

Komandom ping računaru se šalje ICMP ECHO_REQUEST paket, nakon čega se na osnovu odziva zaključuje da li je računar dostupan. Takođe, komandom ping se može utvrditi da li postoji validna ruta do računara. Sledeći primer ilustruje slanje 4 ECHO_REQUEST paketa računaru netfinity:

```
# ping -c 4 netfinity
PING netfinity.mydomain.com (172.16.32.100): 56 data bytes
64 bytes from 172.16.32.100: icmp_seq=0 ttl=128 time=0.3 ms
64 bytes from 172.16.32.100: icmp_seq=1 ttl=128 time=0.2 ms
64 bytes from 172.16.32.100: icmp_seq=2 ttl=128 time=0.2 ms
64 bytes from 172.16.32.100: icmp_seq=3 ttl=128 time=0.2 ms

--- netfinity.internal.vets.edu.yu ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.3 ms
```

Napomena: neki računari ne odgovaraju na ECHO_REQUEST paket. Na primer, mrežni server može biti dostupan putem nekog servisa (na primer http), ali je u isto vreme konfigurisan da ne odgovara na ECHO pakete. Ignorisanje ECHO paketa je zaštitna mera od zlonamernih napada na sistem (bombardovanjem računara ECHO_REQUEST paketima).

route

Komanda route se koristi za prikazivanje i modifikaciju IP tabele rutiranja. Izveštaj koji komanda route prikazuje je identičan izveštaju komande netstat -r. Dodatno, navođenjem opcije -C, komanda route može prikazati kernel keš tabele rutiranja.

traceroute

Komanda traceroute koristi ICMP ECHO pakete za identifikaciju rute do odredišnog računara. Koristi se u dijagnostičke svrhe ukoliko su odredišni host ili odredišna mreža nedostupni - ovom komandom se može odrediti mesto (ruter) na kom se paketi gube, odnosno sa kog se ne mogu dalje proslediti.

nslookup

Komandom nslookup (Name Server lookup) šalje se upit DNS serveru za razrešavanje imena računara.

```
# nslookup nicotine
Note:  nslookup is deprecated and may be removed from future
releases.
Consider using the `dig' or `host' programs instead.  Run nslookup
with
the `-sil[ent]' option to prevent this message from appearing.
Server:         172.16.32.1
Address:        172.16.32.1#53

Name:   nicotine.mydomain.com
Address: 172.16.40.75
```

Komanda nslookup smatra se zastarelom i umesto nje se preporučuje korišćenje komandi host ili dig (Domain Information Gropper).

```
# host nicotine
nicotine.mydomain.com has address 172.16.40.75
```

```
# dig nicotine.mydomain.com

; <<>> DiG 9.2.1 <<>> nicotine.mydomain.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37793
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 0

;; QUESTION SECTION:
;nicotine.mydomain.com. IN      A

;; ANSWER SECTION:
nicotine.mydomain.com. 1200 IN  A       172.16.40.75

;; Query time: 4 msec
;; SERVER: 172.16.32.1#53(172.16.32.1)
;; WHEN: Fri May 28 11:30:47 2004
;; MSG SIZE  rcvd: 63
```

Mrežni servisi i wrapper programi

Većinu mrežnih servisa ne obavlja sam operativni sistem već programi koji se aktiviraju na zahtev. Ovi programi se pokreću najčešće prilikom podizanja operativnog sistema i nakon toga miruju u pozadini (idle) dok se ne pojavi zahtev za uspostavljanjem konekcije na određenom portu. Konekciju prihvata onaj servis koji je dodeljen tom portu i on preko tog porta pruža servis klijentu.

U često korišćene servise spadaju:

ftpd	file transfer protocol daemon - upravlja ftp konekcijama (transfer datoteka sa jednog sistema na drugi);
telnetd	terminal sessions daemon - upravlja telnet sesijama (udaljeno prijavljivanje na sistem). Telnet je nesigurna sesija koja se lako prisluškuje;
sshd	secure shell daemon - servis sličan telnet servisu (udaljeno prijavljivanje na sistem). Za razliku od telnet servisa, ssh koristi kriptografske metode zaštite, odnosno šifruje svu komunikaciju između dva računara, tako da je sigurnost sesije mnogo veća;
sendmail	servis za elektronsku poštu. Teško se konfiguriše, tako da mnogi administratori umesto njega koriste alternativne servere, kao što je Postfix;
httpd	Apache web server;
biod	NFS server (mrežni sistem datoteka);
nfsd	NFS klijent (mrežni sistem datoteka).

Ukoliko je server multifunkcionalan potrebno je aktivirati nekoliko servisa, odnosno programa koji će osluškivati portove i po potrebi odgovarati na zahteve za uspostavljanjem konekcija. U tom slučaju na sistemu će biti podignut veliki broj procesa u pozadini i

performanse sistema će opasti. Da bi se to sperečilo uvode se takozvani wrapper daemon programi (obvojnice) koji osluškuju sve portove opisane u konfiguracionoj datoteci wrapper programa. Nakon zahteva za uspostavljanjem konekcije na nekom portu, wrapper pokreće odgovarajući program koji pruža servis na tom portu i predaje mu kontrolu. Neke servise, kao što su daytime i echo, pruža sam wrapper. Na Linux sistemima wrapper funkcije obavljaju inetd (Internet Services Daemon) i xinetd.

inetd

Inetd upravlja TCP i UDP servisima na sledeći način:

- inetd osluškuje zahteve za uspostavljanjem konekcije na svim TCP portovima specificiranim u inetd konfiguracionoj datoteci. Nakon primanja zahteva za uspostavljanjem konekcije, inetd pokreće program koji pruža servis na tom portu i predaje mu kontrolu nad zahtevom;
- nakon primanja UDP datagrama, inetd pokreće odgovarajući program koji pruža servis na tom UDP portu.

Datoteka /etc/inetd.conf je konfiguraciona datoteka inetd wrapper programa. Svaka linija u datoteci sadrži nekoliko polja razdvojenih razmaknicama. Zadnje polje koje predstavlja argumente programa završava se krajem linije ili karakterom #, nakon čega se u liniju može upisati komentar. Polja redom predstavljaju:

Servis	ime servisa, navedeno kao u datoteci /etc/services;
Tip priključka	stream (za TCP servise) ili dgram (za UDP servise);
Protokol	tcp ili udp;
Čekanje	ukoliko se navede wait, inetd će sačekati da se servis izvrši, odnosno da program prekine sa radom, nakon čega će nastaviti sa osluškivanjem na tom portu. Wait se navodi za sve servise koji, ukoliko se jednom pokrenu, sami nastavljaju osluškivanje na tom portu, kao i za servise koji dozvoljavaju samo jednu konekciju na portu. Ukoliko se navede nowait, inetd će nastaviti da osluškuje port odmah nakon pokretanja servisa. Vrednost nowait se navodi za većinu TCP servisa;
User ID	UID korisnika koji se koristi za pokretanje servera. Neki servisi, kao što je telnetd zahtevaju root privilegije za pravilno izvršenje. Takvi servisi se pokreću sa UID superusera. Ostali servisi koriste UID nobody;
Process to Run	ime programa koji pruža servis;
Command String	argumenti programa navedenog u prethodnom polju.

U nastavku teksta dat je kratak isečak datoteke /etc/inetd.conf:

```
ftp     stream  tcp  nowait  root  ftpd
telnet  stream  tcp  nowait  root  telnetd
shell   stream  tcp  nowait  root  rshd
login   stream  tcp  nowait  root  rlogind
```

Nakon izmene sadžaja konfiguracione datoteke potrebno je poslati signal HUP inetd procesu, nakon čega će inetd ponovo pročitati konfiguracionu datoteku, što se može učiniti komandom kill. Alternativno, inetd se može ponovo pokrenuti sledećom komandom:

```
$ /etc/init.d/inetd restart
Restarting internet superserver: inetd.
```

xinetd

Program xinetd obavlja identičnu funkciju: osluškuje zahteve za uspostavljanjem konekcije i prima UDP datagrame, nakon čega pokreće odgovarajući servis i predaje mu kontrolu nad zahtevom. Konfiguracione datoteke programa xinetd su sledeće:

/etc/xinetd.conf	xinetd globalna konfiguraciona datoteka,
/etc/xinetd.d/service	direktorijum u kome se nalaze sve informacije o konkretnom servisu.

Datoteka /etc/xinetd.conf sadrži podešavanja koja se tiču svih servisa koji su pod kontrolom programa xinetd. Datoteka se čita samo prilikom pokretanja xinetd programa. Ukoliko se sadržaj datoteke modifikuje, xinetd se mora ponovo pokrenuti da bi izmene bile validne. U nastavku teksta dat je isečak datoteke /etx/xinetd.conf:

```
defaults
{
        instances               = 60
        log_type                = SYSLOG authpriv
        log_on_success          = HOST PID
        log_on_failure          = HOST
        cps                     = 25 30
}
includedir /etc/xinetd.d
```

Linije u datoteci imaju sledeće značenje:

instances	određuje maksimalan broj zahteva kojima xinetd može rukovati istovremeno;
log_type	određuje tip praćenja aktivnosti. U ovom slučaju koristi se sistemski authpriv metod, koji upisuje informacije u datoteku /var/log/secure. Ukoliko se umesto njega navede direktiva FILE /var/log/xinetdlog, informacije o praćenju aktivnosti će biti upisane u datoteku /var/log/xinetdlog;
log_on_success	u slučaju uspešnog ostvarivanja konekcije, u log datoteku se upisuju IP adresa računara s kojim je konekcija ostvarena i PID servisa koji upravlja konekcijom;
log_on_failure	u slučaju neuspešnog ostvarivanja konekcije, u log datoteku se upisuje IP adresa računara koji je pokušao da ostvari konekciju;
cps	xinetd će dozvoliti najviše 25 konekcija u sekundi ka svakom servisu. Kada se dostigne limit, servis postaje nedostupan za otvaranje novih konekcija narednih 30 sekundi.

Direktiva includedir /etc/xinetd.d/ specificira xinetd programu da pročita sve konfiguracione datoteke u /etc/xinetd.d direktorijumu. U ovim datotekama su navedeni konfiguracioni parametri specifični za konkretne servise kojima upravlja xinetd. Imena datoteka u /etc/xinetd.d su u korelaciji sa imenima servisa. Kao i datoteka /etc/xinetd.conf, i ove datoteke se čitaju samo prilikom pokretanja programa xinetd. Ukoliko se njihov sadržaj modifikuje, xinetd se mora ponovo pokrenuti da bi izmene bile validne.

Format datoteka u direktorijumu /etc/xinetd.d/ je sličan formatu datoteke xinetd.conf. Osnovni razlog zbog koga se konfiguracija svakog servisa čuva u posebnoj datoteci je mogućnost postavljanja specifičnih parametara za određeni servis koji neće imati uticaja na ostale servise. U nastavku teksta prikazan je sadržaj konfiguracione datoteke telnet servisa (/etc/xinetd.d/telnet):

```
service telnet
{
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/sbin/in.telnetd
        log_on_failure  += USERID
        disable         = yes
}
```

Linije u datoteci imaju sledeće značenje:

service	ime servisa, onako kako je navedeno u datoteci /etc/services;
flags	prateći atributi konekcije;
socket_type	tip priključka;
wait	Za servise koji mogu upravljati većim brojem konekcija (multi-threaded) navodi se no, što znači da se ne mora čekati na servis da zatvori aktivnu konekciju pre otvaranja nove. Ukoliko servis može upravljati samo jednom konekcijom u jednom trenutku (single-threaded), navodi se yes;
user	UID korisnika koji se koristi za pokretanje procesa;
server	izvršna datoteka servisa;
log_on_failure	parametri koji se upisuju u log datoteku prilikom neuspelog ostvarivanja konekcije. U ovom slučaju se koristi log_on_failure += USERID, što znači da se pored parametara definisanih u datoteci xinetd.conf u loga datoteku upisuje i UID korisnika koji nije uspeo da ostvari konekcije;
disable	definiše se da li je servis aktivan ili ne (u ovom slučaju je iz sigurnosnih razloga zabranjeno korišćenje telnet servisa).

Administrator može da navede šta tačno želi da prati. S tim u vezi, kao vrednosti parametara log_on_success i log_on_failure u datoteci xinetd.conf i u datotekama iz direktorijuma /etc/xinetd.d mogu se navesti:

ATTEMPT	beleži se neuspeli pokušaj ostvarivanja konekcije (log_on_failure);
DURATION	beleži se dužina trajanja konekcije (log_on_success);
EXIT	beleži se izlazni status servisa (log_on_success);
HOST	beleži se IP adresa računara koji želi da ostvari konekciju (log_on_failure i log_on_success);
PID	beleži se PID servisa (log_on_success);
RECORD	beleže se informacije o sistemu ukoliko se servis ne može pokrenuti. Ove opcije koriste specifični servisi, kao što je finger;
USERID	beleži se ID korisnika udaljenog sistema za sve multi-threaded stream servise (log_on_failure i log_on_success)-

xinetd i kontrola pristupa

Na sistemima na kojima funkciju wrapper programa obavlja xinetd kontrola pristupa sa udaljenih sistema može se obaviti sledećim metodama:

- korišćenjem datoteka /etc/hosts.allow i /etc/hosts.deny (metod koji koristi inetd),
- korišćenjem xinetd konfiguracionih datoteka,
- kombinovanim korišćenjem prethodnih metoda (primenjuje se unija restrikcija).

Napomena: promene u kontroli pristupa, kao i sve ostale promene u xinetd konfiguracionim datotekama postaju validne tek nakon ponovnog pokretanja xinetd.

Za razliku od datoteka hosts.allow i hosts.deny, kojima se reguliše pristup svim servisima, xinetd dozvoljava da se za svaki servis u odgovarajućoj datoteci definiše sa kog se računara može, odnosno ne može pristupiti servisu.

Xinetd podržava sledeće opcije koje se tiču kontrole pristupa:

only_from	pristup servisu se dozvoljava samo sa navedenih računara (ili IP mreža), pri čemu se računari (mreže) navode pomoću imena ili IP adresa;
no_access	zabranjuje se pristup servisu sa navedenih računara (ili IP mreža), pri čemu se računari (mreže) navode pomoću imena ili IP adresa;
access_times	specificira se vreme kada se specifični servis sme koristiti. Vreme se navodi u 24-časovnom formatu HH:MM-HH:MM.

Sledeći primer ilustruje datoteku /etc/xinetd.d/telnet kojom se blokira pristup Telnet servisu sa mreže čija je IP adresa 166.60.00, a svim ostalim dozvoljava upotreba Telnet servisa u periodu 8:30-17:30h.

```
service telnet
{
        disable         = no
```

```
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/sbin/in.telnetd
        log_on_failure  += USERID
        no_access       = 166.60.00
        log_on_success  += PID HOST EXIT
        access_times    = 08:30-17:30
}
```

Klijentu sa mreže 166.60.0.0 koji pošalje zahtev za otvaranjem Telnet sesije, sistem šalje sledeću poruku:

```
Connection closed by foreign host.
```

Dodatno, pokušaj prijavljivanja na sistem preko telnet sesije se beleži u datoteci /var/log/secure:

```
Sep 15 14:18:42 boo xinetd[11002]: START: telnet pid=11006
from=166.60.25.20
Sep 15 14:18:42 boo xinetd[11006]: FAIL: telnet address
from=166.60.25.20
Sep 15 14:18:42 boo xinetd[11002]: EXIT: telnet status=0 pid=16256
```

xinetd - vezivanje servisa za IP adresu i redirekcija

Dodatno, xinetd podržava mehanizme vezivanja servisa za IP adresu i redirekciju zahteva za otvaranjem konekcije na drugu IP adresu, na drugo ime ili na drugi port.

Vezivanje (binding) se definiše direktivom bind u konfiguracionoj datoteci specifičnog servisa. Na ovaj način se servis povezuje sa nekom IP adresom. Nakon konfigurisanja, servisu se može pristupiti isključivo preko IP adrese s kojom je povezan. Na ovaj način se različiti servisi mogu izvršavati preko različitih mrežnih interfejsa, što je korisno ukoliko se na sistemu nalazi više mrežnih adaptera ili ukoliko je sistem konfigurisan sa više IP adresa. Na primer, nesigurni servisi kao što je Telnet ne moraju se isključiti - servis se može vezati za IP adresu kojoj se može pristupiti samo sa lokalne mreže (interno), ali ne i sa Interneta.

Redirekcijom se zahtevi za uspostavljanje konekcije na određenom portu preusmeravaju na drugi računar, odnosno na drugu IP adresu i port. Na primer, zahtev za uspostavljanjem konekcije na TCP portu 80 može biti preusmeren na Web server.

Prednosti vezivanja i redirekcije dolaze do izražaja ukoliko se metode koriste zajedno. Vezivanjem servisa za konkretnu IP adresu sistema i preusmeravanjem na IP adresu drugog sistema koju samo taj računar može videti, obezbeđuje se sigurno pružanje servisa računarima sa druge IP mreže.

Na primer, ukoliko se računar koristi kao firewall, xinetd konfiguracija za telnet servis može biti realizovana na sledeći način:

```
service telnet
{
        socket_type     = stream
```

```
        wait            = no
        server          = /usr/sbin/in.telnetd
        log_on_success  += DURATION USERID
        log_on_failure  += USERID
        bind            = 166.60.25.25
        redirect        = 172.16.48.10 21 23
}
```

Na ovaj način je Telnet servis vezan za eksternu IP adresu, odnosno IP adresu koja se može videti sa Interneta (166.60.25.25). Svi zahtevi za uspostavljanjem konekcije preko Telnet servisa preusmeravaju se na internu IP adresu (172.16.48.10) kojoj mogu pristupiti samo firewall i računari povezani u internu mrežu. Na taj način se štite računari u internoj mreži.

xinetd i upravljanje resursima

Xinetd obezbeđuje elementarne mehanizme upravljanja resursima, koji se mogu koristiti i kao zaštita od napada tipa Denial of Service (DoS), a ostvaruju se sledećim direktivama:

per_source	najveći broj konekcija istog servisa ka jednoj IP adresi. Prihvata ceo broj kao kao argument i može se navesti i u xinetd.conf datoteci i u datotekama u /etc/xinetd.d direktorijumu;
cps	definiše najveći broj konekcija po sekundi. Direktiva zahteva dva argumenta odvojenih razmaknicom - maksimalni broj konekcija u sekundi i broj sekundi čekanja na dostupnost servisa nakon dostizanja cps limita. Može se navesti i u xinetd.conf datoteci i u datotekama u /etc/xinetd.d direktorijumu;
max_load	definiše maksimalno opterećenje procesora od strane servisa u procentima. Kao argument se može navesti ceo ili decimalni broj.

Linux kao mrežni server

Mrežni sistem datoteka (NFS). Centralizovana autentifikacija (NIS). Apache web server.

Linux se može koristiti kao operativni sistem na radnim stanicama i tada je korisnicima na raspolaganju grafičko radno okruženje sa velikim skupom korisničkih aplikacija, kao što su integrisani office paketi (KOffice, OpenOffice), programi za obradu slika (gimp), razne multimedijalne aplikacije (xmms) i programi za pregledanje sadržaja Web stranica i primanje i slanje elektronske pošte (Mozilla). Ukoliko se Linux koristi kao server, administratorima je dostupna samo komandna linija. Iako to nije zabranjeno, instaliranje grafičkog okruženja na mrežnom serveru se ne preporučuje, jer X windows server troši resurse računara, a takođe opada i sigurnost.

Linux se može koristiti kao server za lokalnu računarsku mrežu i u njoj može obavljati funkcije:

- servera za mrežni sistem datoteka (NFS),
- centralizovane autentifikacije (NIS).

Takođe, Linux se može koristiti i kao server dostupan WAN mreži, i tada najčešće obavlja funkcije:

- rutera,
- mrežne barijere (firewall uređaja),
- Web servera (Apache),
- servera za elektronsku poštu (Postfix).

Mrežni sistem datoteka (NFS)

Pomoću mrežnog sistema datoteka (NFS - Network File System), bilo koji deo aktivnog UNIX stabla može se učiniti dostupnim korisnicima mreže. Najmanja jedinica podataka koja se može učiniti dostupnom na mreži je direktorijum sa kompletnim sadržajem (poddirektorijumi i datoteke). Kako kernel pristupa datotekama na masovnim memorijskim medijumima preko virtuelnog sistema datoteka (VFS), UNIX tretira NFS kao običan sistem datoteka. Kao takav, NFS se aktivira montiranjem na mount-point direktorijum.

Kada rutine kernela na klijent računaru žele da pristupe i-node strukturi koja se nalazi na NFS sistemu datoteka, zahtev se prosleđuje prvom slobodnom nfsd (NFS daemon) procesu (ili niti), koji obrađuje zahtev. Nfsd određuje na kom se udaljenom računaru fizički nalazi taj sistem datoteka, a zatim šalje zahtev za pristup datoteci procesu biod na udaljenom računaru (NFS serveru) preko UDP protokola. NFS server (biod) pristupa datoteci koristeći standardne sistemske pozive za rad sa datotekama i šalje informaciju klijentu preko UDP protokola.

Kako biod pristupa datotekama preko stadnardnih I/O procedura, podaci se keširaju u operativnoj memoriji servera. Podaci kojima se često pristupa su na taj način dostupni u keš memoriji servera – biod ih može dobaviti bez ikakve aktivnosti diska, čime se dobija na brzini.

NFS koristi UDP protokol radi omogućavanja većeg broja paralelnih transakcija na relaciji biod–nfsd. Kako je UDP nepouzdan protokol, NFS klijent (nfsd) koristi time-out mehanizam i na taj način obezbeđuje da će primiti one podatake za koje je poslao zahtev serveru.

NFS server

Svaki sistem čiji je deo aktivnog stabla dostupan na mreži je NFS server. Na NFS serveru se definiše koji su direktorijumi dostupni na mreži (export-points) i sa kojih se računara može tim direktorijumima pristupiti. Jednostavno rečeno, definiše se sadržaj NFS sistema datoteka, odnosno "deljeni direktorijumi" i kontrola pristupa. Minimalna jedinica podataka koja se može učiniti dostupnom na mreži je direktorijum sa svim poddirektorijumima i datotekama - pojedinačne datoteke se ne mogu učiniti dostupnim.

BSD i SVR4 UNIX sistemi koriste različite mehanizme za konfigurisanje i pokretanje NFS servera. Na BSD UNIX i Linux sistemima sadržaj NFS sistema datoteka se definiše u datoteci /etc/exports. Ovu datoteku čita program exportfs koji ažurira sadržaj tekućeg NFS servera. U nastavku teksta dat je primer datoteke /etc/exports:

```
/share/project –access=ws1:ws2:gateway
/share/doc –ro –access=ws1:ws2:gateway
/share/public
/mnt/cdrom –ro
```

Svaka linija sadrži putanju i ime direktorijuma i prateću listu opcija, koja nije obavezna. Datoteka /etc/exports iz prethodnog primera ima sledeće značenje:

- direktorijum /share/project je dostupan kao NFS sistem datoteka sa računara ws1, ws2 i gateway u režimu čitanja i pisanja,
- direktorijum /share/doc je dostupan sa računara ws1, ws2 i gateway u režimu čitanja (read-only),
- direktorijum /share/public je dostupan sa svih računara u režimu čitanja i pisanja,
- direktorijum /mnt/cdrom je dostupan sa svih računara u režimu čitanja. Time se omogućava većem broju korisnika da dele sadržaj kompakt diska (na primer instalacionog diska nekog softverskog paketa).

Superuser može ažurirati sadržaj tekućeg NFS servera pokretanjem komande exportfs -a. Dodatno, superuser može komandom exportfs zaustaviti deljenje nekog direktorijuma, dok ostali korisnici mogu samo videti koji se direktorijumi dele.

Aktiviranje NFS sistema datoteka na klijentima

Pristupanje masovnim memorijskim medijumima udaljenih računara je dvodelna procedura: udaljeni sistem mora da proglasi direktorijume deljenim (exportfs), nakon čega ih klijent može aktivirati. NFS se aktivira kao i svaki drugi sistem datoteka - montiranjem na mount-point direktorijume. Osnovna razlika u odnosu na lokalne sisteme datoteka je u tome što NFS dozvoljava da se:

- montira ceo deljeni direktorijum,
- montira deo deljene hijerarhije, odnosno poddirektorijum deljenog direktorijuma.

Time je omogućena realizacija home direktorijuma korisnika putem NFS sistema datoteka. Na primer, na serveru se može eksportovati struktura /share/home u kojoj se nalaze home direktorijumi korisnika (na primer /share/home/jsmith). Kada se korisnik jsmith prijavi na udaljeni sistem moguće je izvesti da se za tog korisnika montira direktorijum /share/home/jsmith umesto cele strukture.

Montiranje NFS sistema datoteka obavlja se komandom mount, koja kao argumente zahteva ime servera, ime deljenog direktorijuma ili direktorijuma koji predstavlja deo eksportovane hijerarhije i mount-point direktorijum u lokalnom aktivnom stablu:

```
$ mount –t nfs server://share mount-point
```

Na primer, deljeni direktorijum /share/doc sa servera fserver1 može se montirati na direktorijum /doc sledećom komandom:

```
$ mount –t nfs fserver1://share/doc /doc
```

Nakon toga, korisnici lokalnog računara mogu pristupati podacima na serveru jednostavnom navigacijom kroz aktivno UNIX stablo lokalnog računara.

Napomena: ukoliko je NFS server://share naveden u datoteci /etc/fstab, dovoljno je navesti samo mount-point direktorijum. To znači da se prethodna komanda može svesti na:

```
$ mount –t nfs /doc
```

Prilikom aktiviranja, dodatne opcije karakteristične za NFS sisteme datoteka mogu se navesti u komandi mount pomoću argumenta -o pre imena servera.

Statistički izveštaj o korišćenju NFS servera

NFS server vodi statistiku o korišćenju NFS sistema datoteka radi ocene performansi i eventualne dijagnostike. Statistički izveštaj se može dobiti pomoću komande nfsstat, koja obezbeđuje dva tipa izveštaja:

- izveštaj koji opisuje brzinu odziva servera na zahteve NFS klijenata (odziv je određen veličinom srrt - smoothed round trip time, koja treba biti što manja);

```
# nfsstat –m
/files3 from ws2:/files3
Flags: hard,intr,dynamic read size=8192, write size=8192, retrans =
5
Lookups: srtt=7 (17ms), dev=3 (15ms), cur=2 (40ms)
Reads: srtt=15 (37ms), dev=3 (15ms), cur=3 (60ms)
Writes: srtt=28 (70ms), dev=6 (30ms), cur=6 (120ms)
All: srtt=12 (30ms), dev=3 (15ms), cur=3 (60ms)
```

- izveštaj koji opisuje broj i vrstu operacija koje su klijenti izvršili na NFS sistemu datoteka. Na osnovu ovog izveštaja administrator može odrediti kako treba postaviti NFS servere, odnosno koji podaci treba da se nađu na kom serveru, radi raspodele opterećenja.

```
# nfsstat –n
Server nfs:
calls     badcalls
4162132   0
null      getattr      setattr     root     lookup       readlink
read
14 0%     694625 17%   33302 1%    0 0%     2579204 62%  12561 0%
167293 4%
wrcache   write        create      remove   rename       link
symlink
0 0%      154051 4%    6310 0%     4870 0%  709 0%       1665 0%
0 0%
mkdir     rmdir        readdir     statfs
368 0%    367 0%       505799 12%  994 0%
Client nfs:
calls     badcalls     nclget      nclcreate
```

```
26512     0           26512     0
null      getattr     setattr   root     lookup      readlink
read
0 0%      3771 14%    169 1%    0 0%     3775 14%    4 0%
6495 24%
wrcache   write       create    remove   rename      link
symlink
0 0%      11643 44%   182 1%    74 0%    133 1%      0 0%
0 0%
mkdir     rmdir       readdir   statfs
0 0%      0 0%        124 0%    142 1%
```

Centralizovana autentifikacija (NIS)

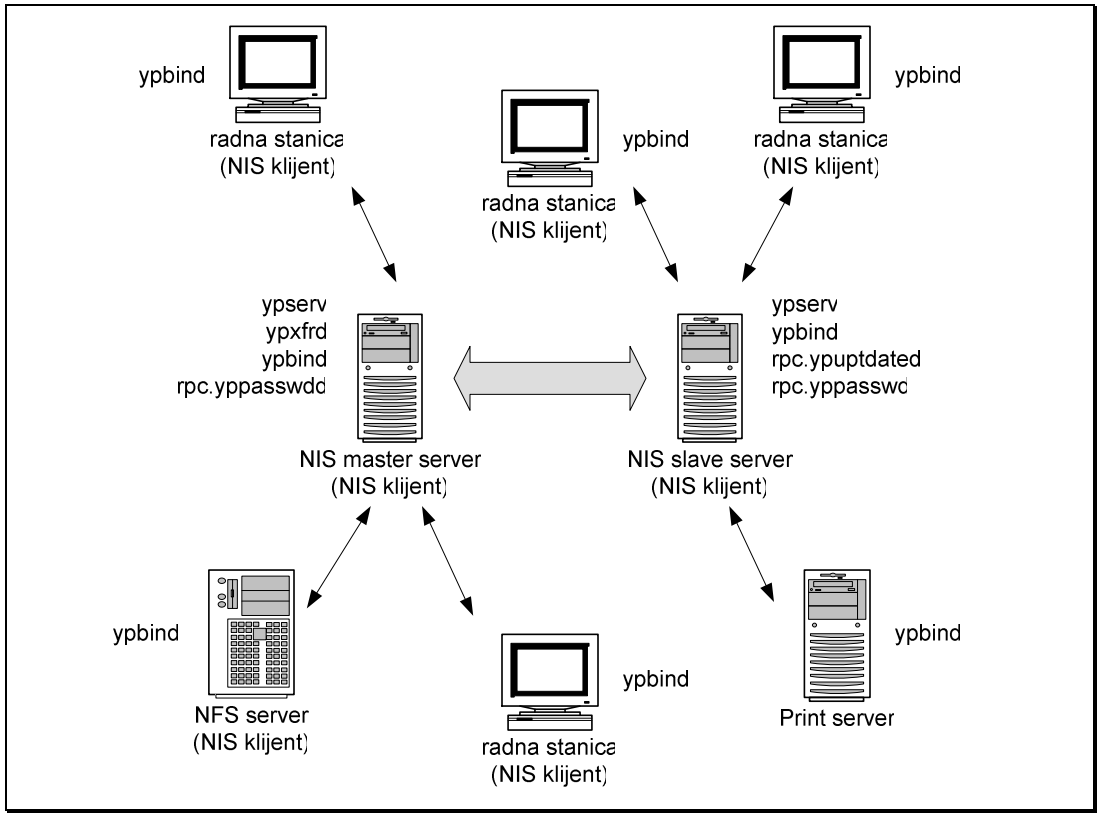
NIS (Network Information System) je UNIX mehanizam centralizovane autentifikacije i upravljanja konfiguracionim datotekama. Ukoliko se NIS ne koristi, svaki UNIX proces koji želi da pristupi konfiguracionim datotekama poput /etc/passwd mora da otvori datoteku sa lokalnog diska i iz nje pribavi odgovarajuće informacije. To znači da sve bitne konfiguracione datoteke moraju postojati na svim računarima u mreži koji koriste UNIX, odnosno Linux sistem. Takođe, ukoliko administrator želi da kreira novog korisnika ili promeni članstvo u grupama, moraće da modifikuje sadržaj datoteke /etc/passwd, odnosno /etc/group na svim računarima u mreži. NIS zamenjuje lokalne konfiguracione datoteke centralizovanom bazom podataka koja sadrži potpuno iste informacije.

NIS grupiše računare u NIS domene. Svi računari u jednom NIS domenu koriste centralizovanu bazu pomoću koje se vrši autentifikacija korisnika i koja obezbeđuje konfiguracione datoteke za sve računare u tom domenu. U okviru jednog domena jedan računar obavlja funkciju NIS master servera. Na tom računaru se nalaze sve konfiguracione datoteke (passwd, group, hosts) od kojih se gradi baza. Dodatno, u jednom domenu postoje i slave serveri čiji se sadržaj ažurira sa master serverima. Ovi serveri odgovaraju NIS klijentima, odnosno svim računarima u domenu koji vrše NIS lookup.

Prilikom podizanja operativnog sistema na klijent računaru u NIS domenu, pokreće se program ypbind (NIS lookup daemon) koji šalje broadcast upit tražeći domenski NIS server. Svi serveri u domenu odgovaraju na upit, a sistem čiji odgovor prvi stigne postaje server datom klijentu. Server koji je opterećen odgovoriće sa zakašnjenjem, dok serveri sa lakšim opterećenjem odgovaraju brže, tako da se na ovaj način vrši raspodela opterećenja među serverima.

Korisnik klijent računara može odrediti koji računar obavlja funkciju njegovog NIS servera komandom ypwhich, dok se računar koji obavlja funkciju master servera može odrediti pomoću komande ypwhich -m.

Napomena: svaki sistem u NIS domenu, uključujući i servere, je NIS klijent. Na primer, master serveru se nakon traženja servera u domenu može dodeliti slave server. To je normalno i ne predstavlja nikakav problem.



Slika 7.6 *NIS domen*

Komponente NIS sistema

NIS sistem čini nekoliko komponenti:

- baza podataka, odnosno skup DBM datoteka,
- NIS servisi (daemon),
- programi za administraciju baze.

Baza podataka se nalazi u /var/yp/domain direktorijumu na svakom serveru (domain je ime domena). Kreiranje NIS baze obuhvata sledeće postupke: kreiranje domena, konverziju datoteka u DMB format baze podataka, kreiranje alias tabele za komandu sendmail, generisanje informacija za RPC sistem, povezivanje korisnika sa mrežnim grupama i uparivanje imena računara sa MAC i IP adresama.

Na master serveru se takođe nalaze i datoteke pomoću kojih se može regenerisati baza (Makefile). Kreiranje i regenerisanje NIS baze vrši se pokretanjem komande make u direktorijumu /var/yp.

NIS servisi koji obezbeđuju rad NIS mehanizma su:

ypserv	NIS server (master ili slave),
ypbind	proces kojim se klijentu dodeljuje NIS server,

ypxfrd	proces na master serveru koji upravlja prenošenjem ažuriranih podataka na slave server,
rpc.ypupdated	proces na slave serveru koji upravlja prenošenjem ažuriranih podataka sa master servera,
rpc.yppasswdd	rešava zahteve za promenu lozinke korisnika sa udaljenih računara.

Programi koji se koriste za administraciju NIS baze su:

ypinit	kreira novi NIS domen,
makedbm	kreira bazu, odnosno konvertuje konfiguracione datoteke u DBM format baze podataka,
mkalias	kreira alias tabelu za komandu sendmail,
mknetid	generiše informacije za RPC sistem,
revnetgroup	generiše datoteku kojom se korisnici povezuju sa mrežnim grupama,
stdethers	generiše datoteku kojom se uparuju imena računara sa MAC adresama,
stdhosts	generiše datoteku kojom se uparuju imena računara sa IP adresama.

Programi koji se koriste za ažuriranje i distribuiranje NIS baze su:

yppoll	zahtev slave servera za ažuriranje baze sa master serverom,
yppush	prosleđuje tabele sa master servera na slave servere,
ypxfr	program koji vrši transfer tabela sa jednog servera na drugi,
ypset	zahtev da sistem bude server procesu ypbin na klijent računaru.

Sistemske datoteke koje ulaze u sastav NIS baze

NIS baza podataka se nalazi u direktorijumu /var/yp/domain na serveru, gde je domain ime domena. Pre kreiranja baze administrator sistema treba da odredit koje će konfiguracione datoteke ući u bazu. U podrazumevanom stanju, u sastav NIS baze ulaze sledeće datoteke:

passwd	svi korisnici sistema,
group	sve korisničke grupe,
hosts	parovi IP adresa i imena računara,
ethers	parovi MAC adresa i imena računara ,
networks	parovi IP adresa i imena mreža,
services	imena IP portova,
protocols	imena IP protokola,

netgroup	pripadnost korisnika mrežnim grupama.

Apache web server

Apache je Open Source projekat, zasnovan na NCSA httpd izvornom kodu. Apache web server je jedan od najpopularnijih i najčešće korišćenih web servera na Internetu. Po statistikama iz oktobra 2003. godine, 64% web-servera na Internetu je upravo neka od verzija Apache-a. Iako Microsoft ulaže dosta u razvoj svoje IIS tehnologije (Internet Information Server), IIS za sada ne predstavlja ravnopravnog konkurenta Apache-u. Navodimo razloge zbog kojih je Apache u prednosti nad konkurentskim web serverima:

- mogućnost izvršavanja na UNIX/BSD/Linux sistemima,
- stabilnost,
- solidne performanse.

Pre svega, prisustvo Windows operativnog sistema ne predstavlja preduslov za instalaciju Apache web servera. Apache je prvenstveno razvijan na različitim Unix/BSD/Linux platformama, a nakon toga je usledilo njegovo prenošenje na Windows okruženje. S druge strane, IIS je strogo vezan za Windows, i to za okruženja izgrađena na NT tehnologiji (New Technology), što mu itekako sužava krug primene. Bez obzira na mnoge napredne opcije samog IIS-a, Windows NT i na njemu bazirani serveri (Windows Server NT/2000/2003) nisu tako poželjni na Internetu kao UNIX/Linux serverski sistemi. Jedan razlog je cena samog softvera - Apache je za sada besplatan i može se pokrenuti na Linux sistemu, koji je takođe besplatan. Drugi razlog su veoma ozbiljni propusti vezani za sigurnost sistema, što postaje veoma aktuelno u poslednje vreme.

Sledeća velika prednost Apache web servera je njegova, u praksi proverena, stabilnost. Mnogi poznati web sajtovi, od kojih su neki zaista veliki, koriste Apache kao svoj primarni web server.

Sem robusnosti i sigurnosti, Apache se odlikuje i solidnim performansama. Naravno, tu tvrdnju treba uzeti sa rezervom, pošto dosta CGI/PERL skriptova, PHP modula, kao i pristupa MySQL bazama podataka može znatno da uspori odziv samog servera, ali ne i da naruši njegovu stabilnost.

Ove karakteristike Apache web servera su pre svega posledica modularne arhitekture. Apache se sastoji od manjeg operativnog jezgra preko koga je moguće učitati različite module i skriptove. Time je omogućeno povezivanje sa mnogim drugim softverskim elementima na samom serveru i u njegovom operativnom sistemu.

Instalacija

Sama procedura instalacije zavisi od operativnog sistema na kom će Apache biti instaliran, kao i od varijante preuzetog instalacionog paketa. Moguće je preuzeti izvorni kod za Linux, izvorni kod za Windows i Windows MSI instalacioni paket. Trenutno, raspoložive

su verzije 1.3.xx i 2.0.xx. Izvršni i izvorni kod Apache web servera, kao i sva potrebna dokumentacija nalaze se na adresi: http://httpd.apache.org.

Izvršena je instalacija Apache ver. 2.0.50 pod Linux okruženjem. U ovu svrhu je preuzeta arhiva sa izvornim kodom httpd-2.0.50.tar.gz. Dobijenu arhivu je najpre potrebno raspakovati:

```
# gzip –d httpd–2.0.50.tar.gz
# tar xvf httpd–2.0.50.tar
```

U tekućem direktorijumu će biti kreiran novi poddirektorijum na koji je potrebno preći kako bi se nastavilo sa prevođenjem koda i daljom instalacijom. Potrebno je izvršiti skript configure sa parametrom --prefix koji određuje gde će se Apache instalirati; ovaj parametar je bitan zbog podešavanja Apache-a (httpd.conf).

```
# ./configure --prefix=/usr/bin/apache
```

Samo prevođenje se obavlja komandom make iz direktorijuma gde je raspakovana izvorna arhiva. Nakon toga sledi instalacija (make install), a zatim i opciono konfigurisanje:

```
# make
# make install
# vi /usr/bin/apache/conf/httpd.conf
```

Nakon konfiguracije, server se može pokrenuti. Pokretanje i zaustavljanje servera se izvodi komandom apachectl.

```
# /usr/bin/apache/bin/apachectl start
# /usr/bin/apache/bin/apachectl stop
```

Nakon pokretanja Apache-a moguće je pomoću web-browsera (na primer, Konqueror ili Mozilla) pristupiti na podrazumevanu stranicu web-servera, koja se nalazi na adresi http://127.0.0.1/. Ovo je ujedno i potvrda uspešne instalacije.

Potrebne privilegije

Ako je Listen direktivom u konfiguracionoj datoteci httpd.conf izabran bilo koji port sa brojem manjim od 1024 (podrazumevana vrednost je 80), neophodno je da Apache ima root privilegije kako bi mogao da se poveže (bind) na privilegovani port. Nakon što je server pokrenut i nakon što je obavio par preliminarnih aktivnosti, kao što je otvaranje log datoteka, on će pokrenuti više podprocesa koji će opsluživati zahteve klijenata. Glavni httpd proces nastavlja da se izvršava kao privilegovan (root) dok će ostali procesi biti izvršavani kao manje privilegovani.

Pokretanje i zaustavljanje web servera - apachectl skript

Metod za pokretanje httpd servera koji se preporučuje je korišćenje apachectl skripta. Ovaj skript podešava neke promenjive okruženja koje su neophodne za pravilno funkcionisanje httpd servera u nekim okruženjima. Skript apachectl prosleđuje argumente date u komandnoj liniji prilikom njegovog pokretanja samom httpd-u, što omogucava

korišćenje bilo koje httpd opcije sa apachectl skriptom. Neke opcije čije je prisustvo u datom okruženju uvek potrebno moguce je specificirati i u samoj datoteci apachectl.

U slučaju neuspešnog pokretanja, Apache će prijaviti grešku na sistemskoj konzoli i/ili u log datoteci.

U slučaju potrebe za automatskim podizanjem web servera pri podizanju sistema, što je najčešći slučaj, potrebno je u sistemske datoteke tipa rc.local ili rc.N uvrstiti komandu za pokretanje servera. Sistemska datoteka se bira u zavisnosti od nivoa izvršavanja (runlevel) u kom se sistem pokreće. Na ovaj način, Apache će biti pokrenut nakon podizanja sistema sa root privilegijama. Zbog toga je potrebno obratiti pažnju na sigurnosna podešavanja (pogledati poglavlje o merama zaštite Apache web servera i sigurnosti UNIX i Linux sistema).

Skript apachectl je dizajniran kao standardni System V init script. Kao takav, on može da uzme parametre start, stop, restart i prevede ih u odgovarajuće signale za sam httpd.

Komunikacija sa httpd procesom

U slučaju potrebe, Apache je moguće zaustaviti ili ponovo pokrenuti slanjem signala httpd daemon procesu. Slanje signala je jedan od načina međuprocesne komunikacije na UNIX/Linux sistemima (pogledati poglavlje o administraciji procesa). Pregledanjem liste aktivnih procesa može se ustanoviti prisustvo većeg broja httpd daemon procesa, ali je signale potrebno slati samo jednom od njih - roditeljskom. Roditeljskom procesu se mogu slati tri signala: TERM, HUP i USR1.

Roditeljski proces kome je poslat TERM (stop) signal pokušava da obavi gašenje svoje dece procesa. Taj postupak može potrajati neko vreme. Nakon toga se gasi i sam roditeljski proces. Opsluživanje zahteva koje je bilo u toku se prekida, a svi novi zahtevi se odbacuju.

Roditeljski proces kome je poslat HUP (restart) signal ubija svoju decu procese, kao u slučaju TERM signala, ali ne prekida svoje izvršenje. On ponovo učitava konfiguracione datoteke i ponovo otvara sve log datoteke, a zatim pokreće novi set dece procesa, čije je izvršenje u skladu sa izmenjenom konfiguracijom, i nastavlja opsluživanje zahteva.

Roditeljski proces kome je poslat USR1 (graceful) signal šalje preporuku deci procesima da prekinu izvršavanje nakon opsluživanja tekućeg zahteva, odnosno odmah, u slučaju da nemaju zahtev za opsluživanje. Zatim roditeljski proces čita svoje konfiguracione datoteke i ponovo otvara log datoteke. Roditeljski proces zatim pokrece novu decu procese čije je izvršenje u skladu sa izmenjenom konfiguracijom.

Signali se roditeljskom httpd procesu mogu slati na dva načina:

- komandom kill, pri čemu se identifikator (PID) roditeljskog httpd procesa može odrediti na osnovu sadržaja datoteke httpd.pid,
- upotrebom apachectl skripta sa odgovarajucim parametrima.

Na primer, signal TERM se roditeljskom httpd procesu može poslati komandom kill na sledeći način:

```
# kill -TERM `cat /usr/local/apache2/logs/httpd.pid`
```

Navodimo i tri primera upotrebe apachectl skripta kojima se roditeljskom httpd procesu šalju TERM, HUP i USR1 signali, respektivno:

```
# apachectl -k stop
# apachectl -k restart
# apachectl -k graceful
```

Konfigurisanje Apache web servera

Apache web server se konfiguriše pomoću direktiva koje se navode u tekstualnim konfiguracionim datotekama. Najvažnija od njih je httpd.conf, a druge datoteke mogu biti dodate korištenjem direktive Include. Apache će reagovati na promene u ovim datotekama nakon ponovnog pokretanja servisa. Server takode učitava datoteku koja sadrži MIME tipove dokumenata čije je ime specificirano TypesConfig direktivom (podrazumevano ime je mime.types).

Sintaksa ovih datoteka je jednostavna. U jednoj liniji se može navesti samo jedna direktiva, a dugačka linija se može podeliti na dve ili više linije navođenjem backslash karaktera (\) na kraju linije. Svi komentari počinju znakom # na početku linije. Apache nije osetljiv na razliku između velikih i malih slova pri navođenju imena direktiva, ali potencijalna ograničenja postoje kod navođenja parametara direktivama.

Direktive koje se navode u glavnoj konfiguracionoj datoteci odnose se na ceo server. Ako postoji potreba da se neki parametri specifično postave, koriste se direktive u drugim opsezima dejstva (scope). Ograničavanje opsega dejstva direktiva se rešava korišćenjem sekcija direktiva.

<Directory> zatvara grupu direktiva koje se odnose samo na određeni direktorijum u sistemu datoteka i njegove poddirektorijume. Na primer:

```
<Directory /usr/local/httpd/htdocs>
Options Indexes FollowSymLinks
</Directory>
```

<DirectoryMatch> odnosi se na direktorijume i njihove poddirektorijume definisane regularnim izrazom.

```
<DirectoryMatch "^/www/.*/[0-9]{3}">
```

<Files> direktive u ovoj sekciji se odnose na datoteke čija imena odgovaraju zadatim.

```
<Files filename> ... </Files>
```

<FilesMatch> direktive u ovoj sekciji se odnose na datoteke čija imena odgovaraju zadatim regularnim izrazima.

```
<FilesMatch "\.(gif|jpe?g|png)$">
```

<Location> direktive u ovoj sekciji se odnose na URL-ove čija imena odgovaraju zadatim.

<LocationMatch> direktive u ovoj sekciji se odnose na URL-ove čija imena odgovaraju zadatim regularnim izrazima.

Ove sekcije ograničavaju dejstvo direktiva koje sadrže na određene lokacije u sistemu datoteka ili URL-ove. Moguće je gnežđenje ovih sekcija, čime se mogu dobiti precizno podešene strukture.

Apache omogućava administratorima da jedan računar koriste kao web server za više web sajtova istovremeno. U tom smislu, pomenućemo još jedno ograničenje dejstva direktiva, odnosno <VirtualHost> sekcije. Korišćenjem ove sekcije moguće je postaviti konfiguracione direktive za različite web sajtove na istom serveru. Detalji vezani za konfiguracione sekcije i direktive mehanizma virtuelnog hostinga su navedeni zajedno sa objašnjenjem dejstva odgovarajućih direktiva.

Apache takođe podržava decentralizovano održavanje konfiguracije, što se postiže upotrebom .htaccess datoteka. Direktive navedene u ovim datotekama odnose se na direktorijum u kome se one nalaze i sve njegove poddirektorijume. Ove datoteke se čitaju prilikom svakog opsluživanja zahteva, tako da su sve promene u njima validne prilikom opsluživanja sledećeg zahteva. Direktive AcessFileName i AllowOverride u glavnoj konfiguracionoj datoteci definišu ime i važnost direktiva u .htaccess datotekama.

Navodimo neke od značajnijih direktiva koje se koriste za konfigurisanje Apache web servera u konfiguracionoj datoteci httpd.conf (detaljna lista direktiva se nalazi na adresi http://httpd.apache.org/docs-2.0/mod/quickreference.html).

ServerRoot direktiva koja definiše koren stabla direktorijuma u kojima se drže Apache konfiguracione datoteke, kao i datoteke u kojima Apache upisuje poruke o greškama.

PidFile direktiva kojom se definiše datoteka u koju Apache upisuje svoj PID (Process ID) prilikom startovanja. Na primer:

```
PidFile logs/httpd.pid
```

Listen direktiva koja omogućava da se Apache poveže na specificiranu IP adresu servera i port, ukoliko su vrednosti različite od podrazumevanih. Na primer:

```
Listen 12.34.56.78:80
```

LoadModule direktiva koja omogućava učitavanje modula koji su prevedeni kao dinamički deljeni objekti - DSO (Dynamic Shared Object). Statički prevedeni moduli se ne učitavaju pomoću ove direktive. Detaljna lista modula se nalazi na adresi http://httpd.apache.org/docs-2.0/mod/. Na primer:

ServerAdmin definiše e-mail adresu administratora ili webmaster-a. Ta adresa se pojavljuje na nekim stranicama koje generiše sam web server (na primer, na stranicama koje prijavljuju neku grešku klijentu).

```
ServerAdmin webmaster@datasecuritysystems.com
```

ServerName — definiše ime i port samog web servera. Poželjno je koristiti ovu direktivu, čak i ako je DNS za dati server konfigurisan.

```
ServerName www.datasecurityssystems.com:80
```

DocumentRoot — definiše direktorijum u kome se nalaze HTML dokumenti kao i ostali elementi web sajta, odnosno web sajtova.

```
DocumentRoot /var/www/html
```

DirectoryIndex — definiše koje će datoteke Apache potražiti kad u zahtevu dobije ime direktorijuma umesto imena konkretne stranice. Na primer, ukoliko klijent u zatraži od servera stranicu http://www.datasecuritysystems/, web server u tom direktorijumu najpre potražiti datoteku index.html, a zatim index.htm.

```
DirectoryIndex  index.html  index.htm
```

AccessFileName — definiše ime datoteke koju će Apache konsultovati po pitanju prava pristupa direktorijumu u kome se ona nalazi. Ova datoteka definiše dodatna prava pristupa, koja nisu navedena u httpd.conf.

```
AccessFileName .htaccess
```

HostNameLookups — vrednostima On i Off se respektivno definiše da li se prati DNS ime klijenta koji je pristupio serveru (na primer www.crackheaven.org) ili samo njegova IP adresa (11.22.33.45). Praćenje DNS imena klijenata (vrednost On) ima negativan uticaj na performanse web servera.

ErrorLog — određuje lokaciju datoteka u kojima se beleže poruke o greškama za primarni web sajt. Ukoliko se koristi VirtualHost direktiva, u čijem sklopu nije navedena posebna ErrorLog direktiva, log grešaka vezanih za taj VirtualHost snima se u ovoj datoteci.

```
ErrorLog logs/error.log
```

CustomLog — definiše lokaciju i ime datoteke za beleženje pristupa web serveru.

```
CustomLog logs/access.log common
```

ServerTokens — konfiguriše HTTP zaglavlje koje web server vraća u odgovoru na zahtev. Full vraca informaciju o operativnom sistemu i kompajliranim modulima.

ServerSignature — ukoliko se postavi na On, dodaje liniju koja sadrži potpis servera, odnosno verziju servera i ime virtuelnog hosta na stranicama koje generiše server (na primer, na stranicama koje sadrže poruke o greškama, listing FTP direktorijuma, itd).

VirtualHost i

NameVirtualHost — virtuelni hostovi se koriste u svrhu održavanja višestrukih domena/imena hostova na postojecem web serveru. Većina konfiguracija koristi virtuelne hostove zasnovane na imenu, tako da server ne mora da vodi računa o IP adresama.

Navodimo i važne direktive koje utiču na performanse web-servera:

Timeout — određuje vreme u sekundama nakon čijeg isteka Apache odbacuje zahtev.

KeepAlive — ukoliko se postavi na vrednost On, omogućava klijentu da održava trajnu konekciju (više od jednog zahteva po konekciji). U slučaju da po jednoj konekciji Apache primi veliki broj zahteva web server će postati preopterećen. To se može izbeći korišćenjem dodatnih direktiva.

MaxKeepAliveRequests — definiše maksimalan broj zahteva u trajnoj konekciji. Vrednost 0 predstavalja neograničen broj zahteva.

KeepAliveTimeout — određuje vreme čekanja za sledeći zahtev od istog klijenta na istoj konekciji u sekundama.

MaxRequestsPerChild i

ThreadsPerChild — za razliku od Windows verzije, Apache za Linux koristi jedan roditeljski proces i veći broj dece procesa kojima opslužuje zahteve. Pri tome, svaki zahtev se obraduje posebnom niti (thread) dece procesa. Direktive MaxRequestsPerChild i ThreadsPerChild predstavljaju maksimalan broj zahteva koje jedan proces opslužuje (vrednost 0 znači da nema ograničenja) i konstantan broj niti u jednom procesu koji opslužuje zahteve.

User — specificira UID korisnika pod kojim će Apache server biti pokrenut. Poželjno je kreirati korisnika sa minimalnim pravima na sistemu koji ce služiti samo za pokretanje Apache web servera. Ne preporučuje se korišćenje regularnih korisnika, jer može doći do preklapanja zahteva za resursima.

Group — specificira GID pod kojim će Apache server biti pokrenut. Poželjno je kreirati grupu sa minimalnim pravima koja će služiti samo za pokretanje Apache web servera.

Postavljanje web sajta

Nakon uspešno okončanog procesa instalacije i konfigurisanja potrebno je postaviti na odgovarajuću lokaciju datoteke koje čine sadržaj web sajta. U ove datoteke najčešće spadaju hipertekst dokumenti (HTML), slike u JPEG i GIF formatu i dinamičke stranice sa aktivnim sadržajem (PHP).Web server nakon pokretanja omogućava klijentima (programima za pregledanje web stranica) da preuzmu te datoteke.

Mere zaštite

Problemi vezani za sigurnost su postali deo svakodnevice. Pitanje sigurnosti je naročito dobilo na značaju zahvaljujući ubrzanom širenju Internet mreže i povecanjem njene dostupnosti običnim korisnicima. Uporedo sa ovim razvijala se i informatička svest kod pojedinaca, koji su potpomognuti informacijama sa samog Interneta, uz podršku raznih "hakerskih" grupa, stekli mogućnost da zahvaljujuci propustima u serverskom softveru i operativnom sistemu gotovo anonimno ugrožavaju udaljene računare. Web sajtovi su jedna od omiljenih meta napadača, s obzirom da obično imaju veliku bazu korisnika, a samim tim pružaju mogućnost sticanja publiciteta na mreži. Postoje i mnogo ozbiljnije posledice napada na web sajtove: zloupotreba tuđeg imena, neovlašćena upotreba važnih informacija, krađa značajnih podataka (kao što su brojevi kreditnih kartica članova nekog kluba). Naročito su u opasnosti sajtovi sa dinamičkom sadržinom (CGI, PHP) koja na odredeni način vrši interakciju sa korisničkim računarom. Iz navedenog se vidi da su web serveri jedna od kritičnih tačaka problema sigurnosti na Internetu.

Neki problemi potiču od samog operativnog sistema koji predstavlja okruženje na kojem se Apache izvršava. U opštem slucaju, Apache je pokrenut od strane root korisnika, a zahteve opsužuje kao korisnik koji je definisan User direktivom (na primer, korisnik www). Kao što je slučaj sa svakom komandom koju izvršava root, potrebno je sprečiti modifikaciju sadržaja programa od strane neprivilegovanih korisnika. Dozvolu za upis u direktorijume, poddirektorijume i datoteke sme imati samo superuser root. Na primer, ako je za ServerRoot izabran /usr/local/apache onda se preporučuje da superuser pre instalacije samog Apache paketa kreira direktorijume na sledeći način (pretpostavla se da samo root korisnik ima pravo modifikacije sadržaja direktorijuma /, /usr i /usr/local):

```
# mkdir /usr/local/apache
# cd /usr/local/apache
# mkdir bin conf logs
# chown 0 . bin conf logs
# chgrp 0 . bin conf logs
# chmod 755 . bin conf logs
```

Nakon instalacije httpd izvršne datoteke, potrebno je preduzeti slične mere zaštite:

```
# cp httpd /usr/local/apache/bin
# chown 0 /usr/local/apache/bin/httpd
# chgrp 0 /usr/local/apache/bin/httpd
# chmod 511 /usr/local/apache/bin/httpd
```

Takođe, može se kreirati poddirektorijum htdocs, u kom i ostali korisnici mogu vršiti izmene, s obzirom da root nikad ne izvršava datoteke sa te lokacije, niti ih na toj lokaciji kreira.

Prethodno navedeni koraci zabranjuju neprivilegovanim korisnicima da menjaju datoteke u gore navedenim direktorijuma. Ukoliko se ovi koraci preskoče, mogući su sledeći scenariji zloupotrebe:

- zamena httpd izvršne datoteke drugom datotekom, koja može da izvede zlonamerne akcije prilikom pokretanja;

- dozvola upisa u log poddirektorijum omogućava nekom neprivilegovanom korisniku da log datoteku zameni linkom na neku sistemsku datoteku, koju superuser može nepažnjom da obriše;
- ako su same log datoteke omogucene za upis, moguce je u njih uneti lažne podatke, sakrivajuci time tragove nekih drugih akcija.

Korišćenje .htaccess datoteka predstavlja dodatnu zaštitnu meru. Pri tome, potrebno je zabraniti korišćenje svih .htaccess datoteka, osim onih čije je korišćenje eksplicitno dozvoljeno.

```
<Directory />
AllowOverride None
</Directory>
```

Sledeca sekcija ce onemoguciti navigaciju korisnika kroz strukturu direktorijuma u sistemu datoteka:

```
<Directory />
Order Deny,Allow
Deny from all
</Directory>
```

Pored ovih osnovnih, ali veoma bitnih mera zaštite, potrebno je stalno pratiti rad web servera pregledanjem log datoteka. Iako ove datoteke prikazuju podatke samo o onome što se već dogodilo, njihovim pregledanjem može se steći predstava o najčešće izvođenim tipovima napada na web server i ideja o protivmerama zaštite koje treba poduzeti.

Na primer, neka je u access.log datoteci prisutna linija slična sledećoj:

```
www.crackheaven.com - - [1/Apr/2004:10:29:59 +0100] "GET /.htpasswd
HTTP/1.1"
```

To znači da postoji sigurnosni problem sa .ht* datotekama. Neovlašćeni korisnici ne smeju preuzeti ove datoteke posredstvom web servera, tako da se kao protivmera u httpd.conf unosi sledeće:

```
<Files ~ "^\.ht">
Order allow,deny
Deny from all
<Files>
```

Nakon toga će pokušaj preuzimanja ovih datoteka prouzrokovati upis linije slične sledećoj u access.log datoteci:

```
[Thu Apr 1 11:01:39 2004] [error] [client www.crackheaven.com]
client denied by server configuration:
/usr/local/apache/htdocs/.htpasswd
```

chroot-jail

Najekstremnija sigurnosna mera podrazumeva smeštanje Apache-a u takozvani chroot zatvor (chroot-jail). Time se ograničava područje sistema datoteka koje je dostupno Apache web serveru. Ovim premeštanjem se kreira nova struktura direktorijuma koja

sadrži samo Apache, njemu potrebne datoteke i minimalan broj drugih programa. Bilo kakav sigurnosni propust u samom Apache-u može da ugrozi samo ovu strukturu, bez naročitog uticaja na ostatak sistema. Apache smešten u chroot-jail više ne predstavlja slabu tačku pogodnu za ugrožavanje serverskog sistema i softvera.

Kreiranje chroot-jail okoline za Apache izvodi se u nekoliko koraka:

- dodavanje novog korisnika (www, UID=80) i grupe (www, GID=80);
- kreiranje direktorijumske strukture;

```
# mkdir /chroot/httpd
# mkdir /chroot/httpd/dev
# mkdir /chroot/httpd/lib
# mkdir /chroot/httpd/etc
# mkdir /chroot/httpd/home
# mkdir /chroot/httpd/tmp
# chmod 777 /chroot/httpd/tmp/
# chmod +t /chroot/httpd/tmp/
# mkdir -p /chroot/httpd/usr/sbin
# mkdir -p /chroot/httpd/var/run
# mkdir -p /chroot/httpd/var/log
```

- prebacivanje konfiguracione i izvršne datoteke u novu chroot strukturu i kreiranje specijalnih datoteka /dev/null i /dev/urandom koje su potrebne za normalno funkcionisanje sistema;

```
# mv /etc/httpd /chroot/httpd/etc/
# mv /home/httpd /chroot/httpd/home/
# mv /var/log/httpd /chroot/httpd/var/log/
# mv /usr/sbin/httpd /chroot/httpd/usr/sbin/
# mknod /chroot/httpd/dev/null c 1 3
# chmod 666 /chroot/httpd/dev/null
# mknod /chroot/httpd/dev/urandom c 1 9
```

- pronalaženje deljenih biblioteka koje koristi httpd:

```
# ldd /chroot/httpd/usr/sbin/httpd
libpam.so.0 => /lib/libpam.so.0 (0x4001b000)
libdl.so.2 => /lib/libdl.so.2 (0x40023000)
libz.so.1 => /usr/lib/libz.so.1 (0x40026000)
...
...
libttf.so.2 => /usr/lib/libttf.so.2 (0x40223000)
libjpeg.so.62 => /usr/lib/libjpeg.so.62 (0x4024a000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

- kopiranje potrebnih bibilioteka na chroot lokacije;

```
# cp /lib/libpam.so.0 /chroot/httpd/lib/
# cp /lib/libdl.so.2 /chroot/httpd/lib/
# cp /usr/lib/libz.so.1 /chroot/httpd/usr/lib/
...
...
# cp /usr/lib/libttf.so.2 /chroot/httpd/usr/lib/
# cp /usr/lib/libjpeg.so.62 /chroot/httpd/usr/lib/
```

```
# cp /lib/ld-linux.so.2 /chroot/httpd/lib/
# strip -R .comment /chroot/httpd/usr/lib/*
```

- kopiranje passwd i group datoteka u kojima se ostavljaju samo www korisnik i www grupa;

```
# cp /etc/passwd /chroot/httpd/etc/
# cp /etc/group /chroot/httpd/etc/
# vi /chroot/httpd/etc/passwd
# vi /chroot/httpd/etc/group
# less /chroot/httpd/etc/passwd
www:x:80:80:Apache Server:/home/httpd:/bin/false
# less /chroot/httpd/etc/group
www:x:80:
```

- kopiranje datoteka /etc/resolv.conf, /etc/nsswitch.conf, /etc/localtime i /etc/hosts;

```
# cp /etc/resolv.conf /chroot/httpd/etc/
# cp /etc/nsswitch.conf /chroot/httpd/etc/
# cp /etc/localtime /chroot/httpd/etc/
# cp /etc/hosts /chroot/httpd/etc/
```

- promenu syslog i httpd skriptova;

```
# vi /etc/rc.d/init.d/syslog
~~daemon syslogd -m 0~~
daemon syslogd -m 0 -a /chroot/httpd/dev/log
# vi /etc/rc.d/init.d/httpd
~~daemon httpd~~
/usr/sbin/chroot /chroot/httpd/ /usr/sbin/httpd
~~rm -f /var/run/httpd.pid~~
rm -f /chroot/httpd/var/run/httpd.pid
```

Nakon toga je potrebno proveriti rad Apache servera u chroot-jail okolini:

```
# /etc/rc.d/init.d/syslog restart
# /etc/rc.d/init.d/httpd start
```

Praćenje rada i održavanje servera

Kao što je već rečeno, analizom sadržaja log datoteka moguće je proveriti samo akcije koje je web servera već izvršio prilikom opsluživanja zahteva. Rad servera se može pratiti i u realnom vremenu, ali su za to potrebni neki dodatni elementi - na primer modul mod_status, koji obezbeđuje pregledanje aktivnih procesa i niti i aktivnost koju trenutno obavljaju. Ovaj modul se učitava pomoću sledećih direktiva navedenih u datoteci httpd.conf:

```
LoadModule status_module libexec/httpd/mod_status.so
<Location /server-status>
   SetHandler server-status
   Order deny,allow
   Deny from all
   Allow from .mclsp.pri
</Location>
```

pri čemu je .mclsp.pri domen sa kog se prati pristup.

Održavanje se svodi na primenu svih aktuelnih zakrpa (patch), osvežavanje izvršnih datoteka servera i modula (update).

8

ŠTAMPAČI

UNIX, kao i svi mrežni operativni sistemi, korisnicima omogućava štampanje dokumenata na lokalnim i udaljenim štampačima. To znači da korisnik može da koristi štampač koji je pomoću paralelnog ili USB kabla povezan na njegov računar, kao i štampače koji su povezani na drugi umreženi računar ili na neki mrežni uređaj. Da bi korisnik uopšte mogao da štampa na računaru mora biti instaliran i pokrenut odgovarajući servis (daemon) koji upravlja zahtevima za štampu. Za razliku od Microsoft Windows ili Mac OS operativnih sistema, UNIX nema standardni interfejs i sistem za podršku štampača, što dovodi do problema pri razvoju drajvera za štampače. Različite vrste UNIX sistema koriste različite servise i skupove komandi kojima se upravlja redom za štampu, a dva najznačajnija rešenja koja se danas koriste deo su System V i BSD UNIX sistema. Ovi sistemi za štampu omogućavaju štampanje teksta na linijskim štampačima i štampanje teksta i grafike na PostScript štampačima. Kasnijim proširenjima uvedena je podrška za veći broj štampača i formata datoteka koje se mogu štampati. Linux sistemi za štampu (lprNG - line printer new generation) i CUPS (common UNIX printing system) objedinjuju System V i BSD interfejse.

Proces štampanja

Proces štampanja pod UNIX sistemom. Štampači, redovi za štampu i print serveri. Uvod u CUPS.

Kada korisnik pošalje zahtev za štampu, zahtev se najpre smešta u red za štampu. Red za štampu je specijalni direktorijum koji se nalazi na hard disku radne stanice ili servera. Zahtev čeka u redu sve dok se štampač ne oslobodi, nakon čega se štampa. Svakom štampaču se dodeljuje red za štampu na računaru na koji je fizički povezan.

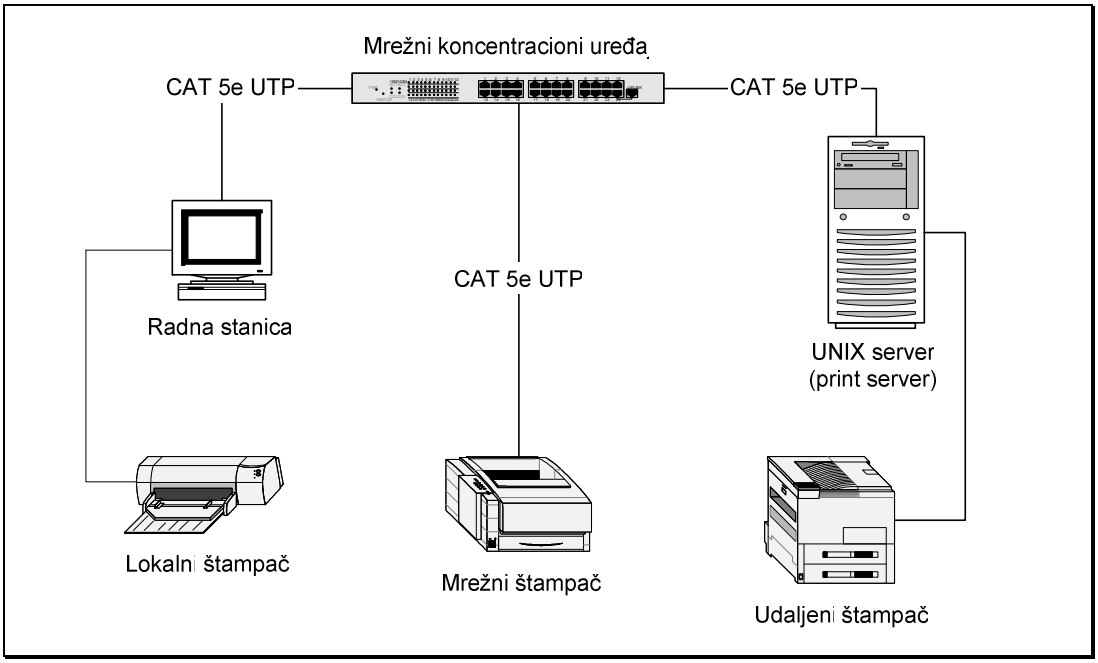
Komponente UNIX okruženja za štampu

Osnovne komponente UNIX okruženja za štampu su:

- štampači
- redovi za štampu
- serveri za štampu

Štampač

Štampač je hardverski uređaj za stampu. Štampač je deljivi resurs - može ga koristiti jedan korisnik ili grupa korisnika. U odnosu na računar sa kog se šalje zahtev za štampu štampač može biti lokalni ili udaljeni - remote. Lokalni štampači su svi štampači povezani na paralelni ili serijski port računara sa kog se šalje zahtev za štampu. Udaljeni štampači su svi štampači povezani na drugi server ili radnu stanicu koji su dostupni sa računara sa kog se šalje zahtev za štampu. U mrežne štampače ugrađeni su NIC adapteri, odnosno mrežne kartice koje omogućavaju direktno vezivanje štampača na mrežne uređaje (Hub ili Switch), pri čemu se veza najčešće ostvaruje UTP kablovima kategorije 5 i 5e. Slika 7.1 ilustruje lokalne, udaljene i mrežne štampače i način na koji se oni vezuju za računare, odnosno mrežne uređaje.



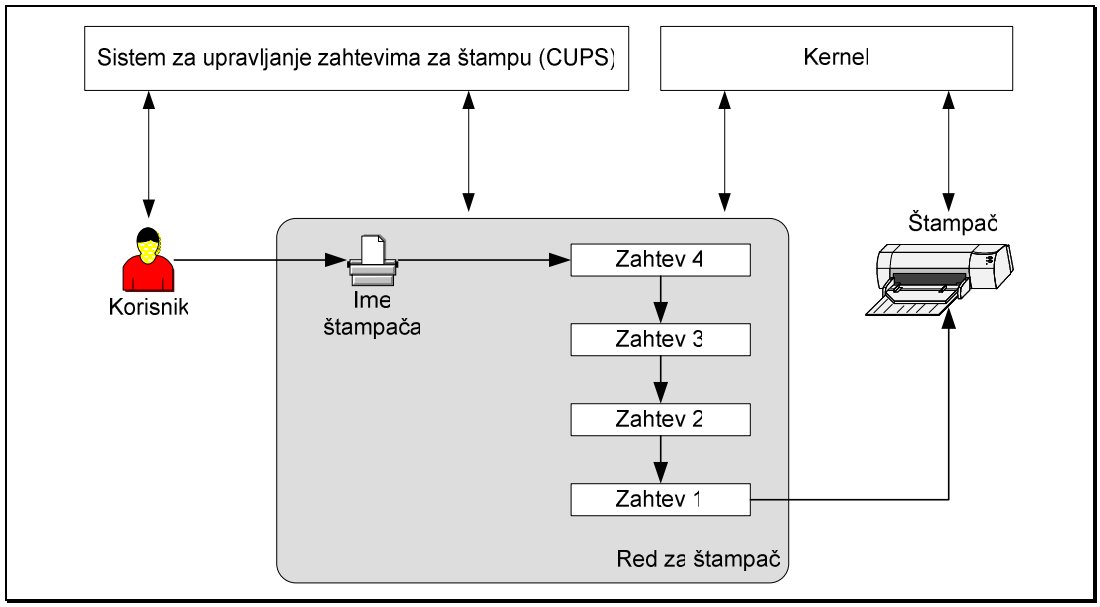
Slika 7.1 *Vrste štampača i načini povezivanja na računare i mrežne uređaje*

Red za štampač

Red za štampač (Printer Queue) je specijalan direktorijum ili datoteka smeštena na hard disku radne stanice ili mrežnog servera. Pošto štampači obično nemaju hard disk za skladištenje dokumenata, svi zahtevi se najpre smeštaju u red za štampač. Ako je štampač slobodan, zahtev se odmah štampa. Ako je štampač zauzet, zahtev čeka u redu sve dok se štampač ne oslobodi. Na UNIX sistemima, svaki štampač ima svoj red koji se nalazi na

disku radne stanice ili mrežnog servera na koji je štampač povezan. Redovi za štampač se obično nalaze u direktorijumu /var/spool kao datoteke /var/spool/lp0 i /var/spool/lp2.

Ime štampača je simboličko ime reda za štampač koje dodeljuje sistem administrator. Ime štampača logički predstavlja hardverski uređaj - korisnici šalju zahteve za štampu koristeći to ime. Ime kao što je hplaser1 opisuje određeni tip štampača (HP LaserJet), a broj 1 ukazuje da je to prvi štampač koji je instaliran na sistemu. Ime štampača može da opiše i njegovu lokaciju u preduzeću, odnosno grupu korisnika kojim je dozvoljen pristup štampaču. Na primer, broj1 opisuje prvi stampač koji se nalazi u odeljenju "projektovanje". Slika 7.2 ilustruje funkcionisanje reda za štampač.



Slika 7.2 *Princip funkcionisanja reda za štampač*

Server za štampu

Server za štampu je računar na kome se nalazi red za štampu. Na serveru za štampu aktiviran je print servis (daemon) koji prihvata dolazeće zahteve, raspoređuje ih u red i šalje na štampu kad je štampač slobodan. Zahtevi za štampu nalaze se u redu sve dok se ne odštampaju, a nakon toga ih print servis briše iz reda.

Server za štampu može da bude radna stanica ili mrežni server. Štampači koji su priključeni na radnu stanicu najčešće koriste isti mrežni server kao server za štampu, pri čemu se red sa svim zahtevima čuva lokalno. Štampači koji su priključeni na mrežni server (ili povezani direktno na mrežni koncentracioni uređaj) najčešće koriste isti mrežni server kao server za štampu. U tom slučaju se zahtevi upućeni sa drugih računara procesiraju centralno. Serveri za štampu mogu da upravljaju većim brojem štampača i redova za štampu.

Sistem administratori postavljaju inicijalno okruženje za štampu tako što instaliraju štampače, definišu redove za štampu i odgovarajuću podršku u vidu servisa. Nakon toga, administrator rukovodi okolinom za štampanje: određuje dostupnost štampača, vrši

kontrolu pristupa štampaču (određuje skup korisnika koji mogu da koriste štampač) i upravlja redovima za štampu.

Common UNIX Printing System (CUPS™)

CUPS je konstruisan radi otklanjanja problema sa štampom. Osnovna ideja je kreiranje sistema za štampu koji može da se koristi na svim varijantama UNIX sistema i opsluži sve korisničke zahteve za štampu. Easy Software Products razvili su CUPS sistem za štampu kao unapređenje standardnih UNIX rešenja za štampu. CUPS korisnicima sistema nudi System V i BSD komandne linijske interfejse, što ga čini kompatibilnim sa velikim brojem UNIX sistema.

CUPS koristi Internet Printing Protocol (IPP) kao standardni protokol za upravljanje poslovima i karakteristikama štampe (veličina papira, rezolucija štampača), a uz redukovanu funkcionalnost podržava i lpd servis za štampu (Line Printer Daemon). IPP sloj se nalazi na vrhu HTTP protokola u okviru TCP/IP skupa protokola, što omogućava administratoru da upravlja štampačima i poslovima za štampu sa udaljene lokacije, koristeći web browser. Dodatno, IPP obezbeđuje kontrolu pristupa - autentifikaciju, i šifrovanje zahteva za štampu, što ga čini sigurnim i fleksibilnim rešenjem.

Svaka datoteka ili skup datoteka koje su poslate na štampu čine jedan posao. CUPS podržava koncept filtriranja: svaki posao se najpre filtrira, odnosno konvertuje u format koji štampač prepoznaje i tako obrađen šalje štampaču, čime je omogućeno jednostavno štampanje datoteka različitih formata. CUPS nudi filtre za štampu raznih grafičkih i tekstualnih datoteka, kao što su PDF datoteke. Dodatno, pomoću RIP filtra (Raster Image Processor), PostScrit datoteke se mogu konvertovati u bitmapirane slike koje se mogu odštampati na bilo kom štampaču.

Backend obavlja najvažniji zadatak - šalje filtrirane poslove za štampu na štampač. CUPS backend šalje poslove štampaču putem paralelnog, serijskog ili USB porta, kao i preko mreže, koristeći IPP, i uz izvesna ograničenja JetDirect (AppSocket) i Line Printer Daemon (LPD) protokole.

CUPS podržava kreiranje klase štampača: poslovi koji se šalju klasi štampača prosleđuju se prvom slobodnom štampaču koji pripada toj klasi. Ova mogućnost podseća na Printer Pool opciju Microsoft Windows sistema.

Drajveri za štampače su najčešće realizovani u vidu PPD datoteka (PostScript Printer Definition) koje opisuju mogućnosti konkretnog štampača, poput podržane veličine stranica. U osnovnom CUPS paketu nalaze se drajveri i PPD datoteke za Hewlett-Packard LaserJet i DeskJet i EPSON 9-pin, 24-pin, Stylus Color i Stylus Photo štampače, čime je omogućeno štampanje na širokom spektru štampača. Otisak generisan na ovaj način na modelima drugih proizvođača često nije u skladu sa mogućnostima štampača, tako da se u osnovnom paketu nalazi i demonstracija pisanja drajvera za konkretan štampač. Dodatno, drajveri za veliki broj štampača sa pratećim filtrima dostupni su i mogu se preuzeti sa adrese www.easysw.cp,/printpro (Easy Software Products).

CUPS je licenciran GNU General Public i GNU Library General Public licencama.

Korišćenje CUPS sistema za štampu

Štampanje i administracija reda za štampu iz komandnog interpretera. Podešavanje karakteristika štampača i dokumenata. Postavljanje podrazumevanog štampača.

U ovom poglavlju opisane su procedure štampanja i administracije reda za štampu iz komandne linije. CUPS objedinjuje standardne System V i BSD komandne interfejse, tako da se za štampanje i administraciju mogu koristiti komande koje pripadaju i jednom i drugom sistemu (na primer lp i lpr). Dodatno, korisnici mogu podesiti karakteristike štampača (veličina papira, margine), teksta (broj karaktera i linija po inču, broj kolona) i slike (osvetljenje i veličina slike) iz komande linije.

Štampanje i administracija reda za štampu

Slanje zahteva na štampu

CUPS obezbeđuje System V (`lp`) i Berkeley (`lpr`) komande za štampu. Koristeći sledeće komande, korisnici mogu odštampati dokumenat na podrazumevanom štampaču u sistemu:

```
$ lp filename
```

ili

```
$ lpr filename
```

Pomoću CUPS sistema korisnici mogu odštampati datoteke raznih formata (tekstualne, PostScript i slikovne datoteke) bez dodatnih filtra, čime je omogućeno štampanje iz aplikacije ili sa komandne linije.

Sa većine umreženih računara korisnicima je dostupno nekoliko štampača. Ti štampači mogu biti priključeni na paralelni, serijski ili usb port lokalnog sistema, ili dostupni preko mreže.

Pre slanja dokumenta na štampu korisnik treba da odredi koji su mu štampači dostupni i koji se štampač koristi kao podrazumevani (default). Komanda lpstat -p prikazuje listu dostupnih štampača:

```
$ lpstat –p
```

Komanda lpstat –d obaveštava korisnika o podrazumevanom (default) štampaču ili klasi.

```
$ lpstat –d
system default destination: hplj1005
```

Nakon toga, zahtev se šalje na štampu komandama lp -d ili lpr -P, čime se specificira odredišni štampač.

```
$ lp –d printer filename
$ lpr –P printer filename
```

Većina datoteka može se verodostojno odštampati ukoliko se koriste podrazumevani parametri štampe. U nekim slučajevima (na primer, ako se koristi format papira koji nije podrazumevan), korisnik će morati da promeni parametre štampe prilikom slanja dokumenta na štampu. To se može učiniti navođenjem opcije -o komandama lp ili lpr, pri čemu same opcije zavise od konkretnog štampača.

```
$ lp –o landscape –o scaling=75 –o media=A4 balthazar.jpg
$ lpr –o landscape –o scaling=75 –o media=A4 balthazar.jpg
```

Podrazumevano, CUPS štampa jednu kopiju dokumenta. Korisnici po želji mogu odštampati veći broj kopija istog dokumenta, navođenjem parametra -n, odnosno -# komandama lp i lpr.

```
$ lp –n noofcopies filename
$ lpr –#noofcopies filename
```

Provera statusa štampača

Koristeći komande lp i lpr korisnici šalju zahtev u red za štampu. Na većim mrežama, sa centralizovanim štampačima, u redovima za štampu često se nalazi veliki broj poslova.

System V komanda lpstat (line printer status) i BSD komanda lpq (line printer queue) na BSD sistemu prikazuju status, odnosno stanje reda za štampu. Komande prikazuju status reda za podrazumevani štampač. Status reda konkretnog štampača može se videti pomoću komande lpstat -d.

```
$ lpstat ENTER
Printer-1 johndoe 4427776
Printer-2 johndoe 15786
Printer-3 johndoe 372842
```

Poslovi su izlistani po redu po kojem ce biti štampani. Ukoliko se navede parametar -p, komanda lpstat prikazuje aktivnu datoteku i aktivni štampač.

```
$ lpstat –p
printer DeskJet now printing DeskJet-1.
```

Program lpq formira statusni izveštaj na osnovu informacija dobijenih od programa lpd (line printer deamon) ili cupsd (CUPS daemon). Komanda lpq bez argumenata na ekranu prikazuje izveštaj o podrazumevanom štampaču i broju poslova u redu za taj štampač. Dugi format izveštaja (opcija -l) uključuje dodatne informacije o poslovima u redu poput imena korisnika koji je poslao zahtev za štampu, imena računara sa kog je zahtev poslat, opisnih informacija o poslu i veličine posla u bajtovima. S obzirom na postojanje prioriteta redovi za štampu se prazne po modifikovanom FIFO principu.

U argumente komande lpq spadaju:.

-P printer specifira se štampač čiji se red ispituje,

-L prikazuje izveštaj u dugom formatu,

jobid ... all specificira se skup poslova od interesa za analizu.

Primer ilustruje korišćenje lpq komande:

```
$ lpq -Phplj1005@tulip
printer is ready and printing
Rank     Owner/ID      Class     Job  Files          Size      Time
active   nm@tulip+466  A         466  /etc/passwd    188
     15:39:00
```

Pošto CUPS koristi Internet Printing Protocol status reda za štampu može se proveriti pomoću bilo kog web browsera (Mozilla, Netscape) na stranici: http://localhost:631. Na ovoj stranici može se proveriti status štampača, klasa i poslova u redu.

Brisanje poslova iz reda

Korisnici žele da uklone pogrešno poslate dokumente iz reda za štampu (naročito ukoliko su dokumenti veliki, a štampanje se naplaćuje po stranici). Svaki korisnik može da ukloni svoje poslove iz reda, ali ne i poslove drugih korisnika, dok superuser može da ukloni bilo koji posao iz reda. Poslovi se brišu iz reda komandama lprm (System V) i cancel (BSD). Pre brisanja, posla potrebno je odrediti broj posla pomoću komande lpstat.

```
$ cancel job-id ENTER
$ lprm job-id ENTER
```

Sledeći primer ilustruje određivanje broja posla komandom lpstat i brisanje zahteva iz reda komandom lprm.

```
$ lpq -Phplj1005@tulip
printer is ready and printing
Rank     Owner/ID      Class     Job  Files          Size      Time
active   nm@tulip+466  A         466  /etc/passwd    188
     15:39:00
$ lprm job 466
```

Na ovaj način se iz reda uklanja samo jedan posao. Komanda lprm omogućava da se iz reda uklone svi poslovi koje je poslao određeni korisnik (parametar -user), odnosno svi poslovi koji se nalaze u redu za određeni štampač (parametar -P). Za ove administrativne postupke najčešće su potrebne privilegije superusera.

Primer redom ilustruje uklanjanje svih poslova koje je započeo korisnik jsmith sa štampača lp0 i uklanjanje svih poslova iz reda sa štampača lp1.

```
$ lprm -Plp0
$ lprm -Plp0 -user john
```

Direktno štampanje

Direktno štampanje je izvodljivo na lokalnim štampačima putem redirekcije izlaza. Direktnim štampanjem zaobilazi se red za štampu, što se ne preporučuje ukoliko je štampač dostupan i drugim korisnicima preko mreže. Dodatno, korisnik koji štampa direktno ne može da koristi pogodnosti CUPS sistema kao što su filtri.

```
$ cat file >/dev/lp0
```

Podešavanje karakteristika štampača i dokumenata

Ovo poglavlje opisuje standardne opcije štampača koje su dostupne pri štampanju datoteka iz komandne linije komandama lp i lpr.

Opšte karakteristike

U karakteristike štampača koje se primenjuju prilikom štampanja svih tipova datoteka spadaju:

- veličina papira,
- orijentacija,
- jednostrana ili dvostrana štampa.

Veličina papira navodi se pomoću opcije -o media=size komande lp ili lpr.

```
$ lpr -o media=size,options filename
```

Pri tome, veličina papira (size) može biti:

Letter	US Letter (216x279mm)
Legal	US Legal (216x356mm)
A4	ISO A4 (210x297mm)
COM10	US #10 Envelope (241x105mm)
DL	ISO DL Envelope (220x110mm)

Dodatno se kao parametri opcije -o media mogu navesti i transparency (čime se specificira štampanje na transparentnoj foliji) i izvor papira - paper tray (Upper, Lower, MultiPurpose, LargeCapacity).

Ukoliko se drugačije ne navede prilikom štampe se koristi portrait (vertikalna) orijentacija stranice. Navođenjem parametra -o landscape, stranica se rotira za 90 stepeni radi štampe u horizontalnoj orijentaciji.

```
$ lpr -o landscape filename
```

Navođenjem opcije -o sides=value specificira se jednostrana štampa (podrazumevana vrednost) ili dvostrana štampa (ukoliko je štampač podržava). U dozvoljene vrednosti value spadaju: one-side-short-edge, two-side-long-edge i two-side-short-edge.

Štampanje naslovne stranice (banner page)

Korišćenje opcije -o job-sheets specificira se štampanje naslovne i poslednje stranice dokumenta. Naslovna stranica obično uključuje ime dokumenta i ime korisnika koji je dokument poslao na štampu.

```
$ lpr -o job-sheets=start,end filename
```

Sadržaj vrha i dna naslovne i poslednje stranice specificiran je parametrima start i end (end nije obavezan parametar). Oznake mogu biti: classified, confidential, secret, topsecret i unclassified, a ako se navede standard, naslovna stranica se štampa bez oznake.

Opšte karakteristike dokumenata

U karakteristike dokumenata koje se primenjuju prilikom štampanja svih tipova datoteka spadaju:

- opseg stranica,
- štampanje parnih i neparnih stranica,
- štampanje stranica u rastućem (od prve ka poslednjoj) ili opadajućem redu (od poslednje ka prvoj),
- broj stranica dokumenta po stranici papira,
- štampanje lika u ogledalu,
- podešavanje osvetljenja,
- Gamma korekcija.

U podrazumevanom stanju štampa se ceo dokument, i parne i neparne stranice, redom od prve ka poslednjoj, pri čemu se na jednoj strani papira štampa jedna stranica dokumenta.

Opseg stranica može se promeniti opcijom -o page-ranges. Pri tom, opseg se može specificirati kao jedna ili više stranica ili kolekcija podopsega. Stranice će uvek biti štampane u rastućem redu, bez obzira na redosled stranica u page-ranges opciji. Primeri ilustruju zadavanje opsega stranica za štampu:

```
$ lp -o page-ranges=1 filename
$ lp -o page-ranges=1-4 filename
$ lp -o page-ranges=1-4,7,9-12 filename
```

Opcijom -o page-set korisnik može da naznači da li želi da štampa samo parne (even) ili neparne (odd) stranice. Podrazumevano, štampaju se sve stranice.

```
$ lp -o page-set=odd filename
$ lp -o page-set=even filename
```

Redosled štampanja stranica u podrazumevanom stanju je rastući (normal), a može se izmeniti opcijom -o output-order na sledeći način:

```
$ lp -o outputorder=normal filename
$ lp -o outputorder=reverse filename
```

Korisnik dodatno može da štampa veći broj stranica dokumenta (1,2,4,9,16) na jednoj stranici papira opcijom -o number-up:

```
$ lp -o number-up=1 filename
$ lp -o number-up=2 filename
$ lp -o number-up=4 filename
```

Ukoliko se štampa na preslikačkoj foliji ili papiru poželjno je koristiti opciju -o mirror, čime se štampa "lik u ogledalu" stranice:

```
$ lp -o mirror filename
```

Osvetljenje štampe može se podesiti koristeći -o brightness=level opciju. Vrednosti veće od 100 osvetliće otisak, dok vrednosti manje od 100 zatamnjuju otisak. Podrazumevana vrednost je 100. Gama korekcija se vrši parametrom -o gamma=value. Vrednosti veće od 1000 posvetliće otisak, dok vrednosti manje od 1000 zatamnjuju otisak.

```
$ lp -o brightness=120 filename
$ lp -o gamma=1700 filename
```

Karakteristike tekstualnih dokumenata

Sledeće opcije važe pri štampanju tekstualnih datoteka:

- broj karaktera po inču (podrazumevana vrednost je 10)

```
$ lp -o cpi=10 filename
$ lp -o cpi=12 filename
$ lp -o cpi=17 filename
```

- broj linija po inču (podrazumevana vrednost je 6)

```
$ lp -o lpi=6 filename
$ lp -o cpi=8 filename
```

- broj kolona na stranici (podrazumevana vrednost je 1)

```
$ lp -o columns=2 filename
$ lp -o columns=3 filename
```

- margine stranice (vrednosti margina value navode se u tačkama, pri čemu je jedna tačka jednaka 1/72 inča, odnosno 0.35mm)

```
$ lp -o page-left=value filename
$ lp -o page-right=value filename
$ lp -o page-top=value filename
$ lp -o page-bottom=value filename
```

- štampanje zaglavlja koje uključuje broj stranice, oznaku posla (najčešće ime datoteke) i datum na vrhu svake stranice. Dodatno, ključne reči C i C++ jezika su označene, a komentari se štampaju u kurzivu.

```
$ lp -o prettyprint filename
```

Karakteristike grafičkih dokumenata

Sledeće opcije važe pri štampanju grafičkih datoteka (slika):

- pozicioniranje slike na sredinu stranice (center), vrh (top), dno (bottom), levu (left) ili desnu (right) stranu papira. Dodatno, slika se može pozicionirati u neki od uglova (top-left, top-right, bottom-left, bottom-right)

```
$ lp –o position=value filename
```

- skaliranje slike (vrednosti percentage opcija scaling i natural-scaling se navode u procentima) i podešavanje broja tačaka po inču (ppi - pixel per inch)

```
$ lp –o scaling=percentage filename
$ lp –o natural-scaling=percentage filename
$ lp –o ppi=value filename
```

Slanje dokumenta na štampu bez filtriranja

Ukoliko na sistemu postoji odgovarajući filtar datoteka se pre štampe filtrira, odnosno priprema za štampu. Ukoliko postoji eksplicitna potreba datoteke se mogu poslati na štampač direktno, bez filtriranja:

```
$ lp –o raw filename
$ lpr –l filename
```

Podrazumevana podešavanja štampača

Ovo poglavlje opisuje postupak podešavanja podrazumevanog štampača i opcija štampača.

Sistem administrator inicijalno postavlja podrazumevani štampač za sve korisnike. Korisnici naknadno mogu promeniti podrazumevani štampač komandom lpoptions -d.

```
$ lpoptions –d printername
```

Štampač može da bude lokalni (na primer deskjet) ili udaljeni (laserjet@printserver).

Za svaki štampač može se podesiti podrazumevani skup opcija (default) koji će se koristiti prilikom štampanja dokumenata. To je mnogo jednostavnije od navođenja velikog broja opcija prilikom svakog štampanja.

Podrazumevane opcije štampača podešavaju se komandom lpoptions. Opcije se navode pomoću parametra -o, kao kod komandi lp i lpr. Na primer, sledeća komanda specificira štampanje na A4 papiru sa posvetljenjem otiska od 20%.

```
$ lpoptions –o media=A4 –o brightness=120
```

Komande lp i lpr koriste podrazumevana podešavanja, kao da su opcije navedene samim komandama (lpr -o media=A4 -o brightness=120).

Napomena: svaki korisnik može podesiti lični skup podrazumevanih opcija, kao i podrazumevani štampač. Koristeći komandu lpoptions, superuser root podešava podrazumevane opcije za sve korisnike. Podrazumevane opcije se ne mogu podesiti za root nalog.

Ukoliko se ne specificira štampač podešavanja izvršena komandom lpoptions odnose se na podrazumevani štampač. Drugi štampač se mora eksplicitno specificirati parametrom -p:

```
$ lpoptions –p laserjet –o prettyprint –o media=Legal
```

Predhodna dva primera ilustruju postupke postavljanja opcija za podrazumevani i specificirani štampač. Postavljene opcije se naknadno mogu ukloniti koristeci parametar –r.

```
$ lpoptions –p laserjet –r prettyprint
```

Komandom lpoptions može se prikazati trenutni skup podrazumevanih opcija štampača:

```
$ lpoptions
media=A4 brightness=120
$ lpoptions –p laserjet
media=Legal
```

Instance štampača

CUPS omogućava korisnicima da postave nekoliko skupova podrazumevanih podešavanja za isti štampač. Svaki skup se definiše pomoću slash karaktera (/).

```
$ lpoptions –p laserjet/image –o media=A4 –o brightness=120 –o
position=center
$ lpoptions –p laserjet/text –o media=Legal –o prettyprint –o
cpi=12 –o lpi=8 –o columns=2
```

Komande lp i lpr takođe razumeju ovu notaciju:

```
$ lp –d laserjet/image sličica.jpg
$ lpr –P laserjet/text helloworld.C
```

Nepotrebne instance štampača mogu se ukloniti komandom lpoptions -x printer/instance:

```
$ lpoptions –x laserjet/image
```

Napomena: komanda lpoptions -x uklanja instancu. Red za štampu uklonjene instance se ne briše dok se ne isprazni ili dok ga superuser ne obriše komandom lpadmin.

Administracija CUPS sistema za štampu

Instalacija CUPS sistema. Upravljanje štampačima, klasama štampača i administracija reda za štampu

Administracija CUPS sistema za štampanje obuhvata postupke instalacije CUPS sistema i štampača, upravljanje štampačima, klasama štampača i administraciju redova za štampu. Ovi administrativni zadaci ubrajaju se u domen aktivnosti superusera.

Instaliranje CUPS sistema

Easy Software Products distribuira binarnu verziju CUPS sistema u sledećim formatima:

- TAR format sa instalacionim i deinstalacionim skriptovima (prenosiva distribucija),
- RPM paket,
- DEB paket.

CUPS u prenosivom tar.gz formatu dostupan je za sve UNIX platforme, dok su distribucije u RPM i DEB paketima dostupne jedino za Linux.

Napomena: instalacija CUPS-a uništiće instalaciju postojećeg sistema za štampanje. Ukoliko postoji problem sa CUPS softverom, isti se može ukloniti pomoću priložene deinstalacione skripte, nakon čega se stari sistem (na primer lprNG) može reinstalirati sa instalacionog diska UNIX distribucije.

Instalacija CUPS softvera distribucije zahteva da se na sistem najpre prijavi superuser.

Ukoliko se sistem instalira iz TAR distribucije, arhivu najpre treba raspakovati u neki privremeni direktorijum (na primer /cups-install), nakon čega se pokreće instalacioni skript:

```
# ./cups.install
```

U toku instalacije skript zahteva odgovore na par jednostavnih pitanja, nakon čega se inicijalizuje red za štampu i pokreće odgovarajući servis.

Instalacija RPM distribucije pokreće se sledećim komandama:

```
# rpm -e lpr
# rpm -i cups-1.1-linux-M.m.n-intel.rpm
```

Nakon instalacije inicijalizuje se red za štampu i pokreće odgovarajući servis.

Instalacija DEB paketa pokreće se sledećim komandama:

```
# dpkg -i cups-1.1-linux-M.m.n-intel.deb
```

Nakon instalacije inicijalizuje se red za štampu i pokreće odgovarajući servis.

Upravljanje štampačima

Ime svakog štampača može da sadrži najviše 127 slova, brojeva i underscore karaktera. U imenu štampača ne smeju se naći specijalni karakteri poput "/" i "@" niti razmaknice. Imena štampača nisu osetljiva na mala i velika slova (case-sensitive): "PRINTER", "Printer", i "printer" se smatraju istim imenom.

Štampači su povezani na paralelni ili USB port računara (predstavljeni specijalnim datotekama /dev/lp1 i /dev/usb/lp0), ili na mrežni koncentracioni uređaj. CUPS koristi koncept jedinstvenih identifikatora resursa (URI), koji su uopštenija forma jedinstvenih lokatora resursa (URL) korišćenih u Web browserima.

Potpuna lista podržanih uređaja (portova i mrežnih protokola) koji se mogu koristiti za štampanje može se dobiti pomoću komande lpinfo -v (opcija -v naznačava prikazivanje liste dostupnih uređaja).

```
# lpinfo -v
network socket
network http
network ipp
network lpd
direct parallel:/dev/lp1
serial serial:/dev/ttyS1?baud=115200
serial serial:/dev/ttyS2?baud=115200
direct usb:/dev/usb/lp0
network smb
```

Rezultat tipa tabele u kojoj svaka linija predstavlja jedan uređaj. Prva kolona predstavlja vrstu uređaja a iza nje sledi ime URI-ja ili specijalne datoteke kojom je uređaj predstavljen. Ukoliko je uređaj za štampu regularna datoteka, koristi se URI: file:/directory/filename. Ukoliko se štampanje vrši preko mreže, uređaj se predstavlja pomoću oznake protokola, imena servera za štampu i štampača.

Instalacija prvog štampača

Štampač se na CUPS sistem može dodati na dva načina: komandom lpadmin (/usr/sbin/lpadmin) i Web interfejsom za administraciju štampača, koji se nalazi na adresi http://localhost:631/admin. Komandom lpadmin se iz komandne linije može izvršiti većina administrativnih zadataka vezanih za štampače. Štampač se dodaje na sledeći način:

```
# /usr/sbin/lpadmin -p printer -E -v device -m ppd
```

Sledeći primeri ilustruju redom:

- dodavanje matričnog štampača koji je povezan na paralelni port

```
# /usr/sbin/lpadmin -p DotMatrix -E -v parallel:/dev/lp1 -m
epson9.ppd
```

- dodavanje HP DeskJet štampača koji je povezan na USB port računara

```
# /usr/sbin/lpadmin -p DeskJet -E -v usb:/dev/usb/lp0 -m
deskjet.ppd
```

- dodavanje HP LaserJet štampača koji koristi JetDirect mrežni interfejs na IP adresi 172.16.32.44.

```
# /usr/sbin/lpadmin -p LaserJet -E -v socket://172.16.32.44 -m
laserjet.ppd
```

Napomena: epson9.ppd, deskjet.ppd i laserjet.ppd su odgovarajuće PPD datoteke za epson 9-pin, HP DeskJet i HP LaserJet štampače, koje su uključene u osnovni CUPS paket.

Korišćenje CUPS Web servera je jednostavnije od komandne linije: potrebno je u Web browseru otvoriti stranicu http://localhost:631/admin i pritisnuti na dugme Add Printer,

čime se pokreće postupak dodavanja štampača. Dalje je umesto nagađanja imena URI uređaja i PPD datoteke koje je potrebno navesti programu lpadmin, dovoljno pritisnuti mišem na odgovarajuću listu i uneti neke jednostavne informacije.

Dodavanje novih i modifikacija instaliranih štampača

Naredba lpadmin–p se koristi za dodavanje novih i modifikaciju karakteristika postojećih štampača.

```
# /usr/sbin/lpadmin –p printer options
```

Kao mogući argumenti (options) mogu se navesti:

-c class	dodaje navedeni štampač u klasu štampača class. Ako navedena klasa ne postoji, biće kreirana;
-i interface	kopira navedeni interfejs skript u štampač. Interfejs skriptove koriste System V štampački drajveri. Interfejs skripte onemogućavaju filtriranje i koriste se samo ako ne postoje drugi drajveri za štampač;
-m model	specificira standardni drajver koji je najčešće PPD datoteka. PPD datoteke se nalaze u /usr/share/cups/model/ direktorijumu. Lista svih dostupnih modela se može prikazati komandom lpinfo –m;
-r class	uklanja navedeni štampač iz class klase štampača. Ako navedena klasa ostane prazna, biće izbrisana;
-v device-uri	dodeljuje zadatke komunikacije sa štampačem novom uređaju (URI-ju ili sitemskoj datoteci). Ako se određeni zadatak izvršava na navedenom štampaču, biće prekinut i restartovan preko novog uređaja;
-D info	daje tekstualni opis štampača, npr, "Laser1";
-E	omogućava rad štampača (enable) i prihvata zadatak (accept);
-L location	daje tekstualnu informaciju o lokaciji štampača, npr. "Moe`s Tavern";
-P ppd-file	specifira lokalnu PPD datoteku kao drajver za štampač.

Brisanje štampača

Štampač se briše komandom lpadmin–x na sledeći način:

```
# /usr/sbin/lpadmin –x printer
```

Postavljanje primarnog štampača

Primarni štampač se postavlja komandom lpadmin –d. Primarni štampač se može zaobići korišćenjem naredbe lpoptions.

```
# /usr/sbin/lpadmin -d printer
```

Pokretanje i zaustavljanje štampača

Komande enable i disable pokreću i zaustavljaju rad štampača::

```
# /usr/bin/enable printer
# /usr/bin/disable printer
```

Štampač čiji je rad zaustavljen može da primi poslove štampanja, ali ih neće izvršiti sve dok se ne pokrene ponovo. Zaustavljanje rada štampača je korisno ako štampač loše radi i treba ga privremeno zaustaviti radi servisiranja. Bilo koji poslovi koji su na čekanju biće odštampani nakon što se štampač ponovo pokrene.

Prihvatanje i odbijanje poslova štampanja

Kao što je rečeno, zaustavljen štampač i dalje prima poslove. Štampač takođe može da odbije nove poslove nakon što završi tekuće, što je korisno ukoliko se na primer, na servisiranje štampača čeka nekoliko dana. Odbijanjem poslova sprečava se nagomilavanje zahteva za štampu u redu. Komande accept i reject prihvataju i odbijaju poslove za određeni štampač:

```
# /usr/sbin/accept printer
# /usr/sbin/reject printer
```

Podešavanje kvota na štampaču

CUPS podržava koncept kvota po pitanju broja i veličine stranica za svaki štampač. Kvote su povezane sa svakim korisnikom pojedinačno, ali jedan skup ograničenja važi za sve korisnike određenog štampača. Na primer, mogu se postaviti kvote od pet stranica dnevno za svakog korisnika koji štampa na nekom skupom štampaču, ali se ne mogu postaviti kvote za sve korisnike osim za korisnika jsmith.

Parametri job-k-limit, job-page-limit, i job-quota-peiod određuju da li će i kako kvote biti nametnute štampaču. Parametar job-quota-period određuje vremenski interval za praćenje kvota. Interval je izražen u sekundama, tako da dan iznosi 86.400, nedelja 604.800, a mesec 2.592.000 sekundi. Opcija job-k-limit određuje ograničenje veličine posla u kilobajtima, a job-page-limit ograničenje broja štampanih stranica. Da bi kvote bile nametnute, period i bar jedno od ograničenja moraju biti postavljeni na vrednost različitu od nule. Dodatno, mogu se kombinovati sve tri opcije u jednoj komandnoj liniji.

```
# /usr/sbin/lpadmin -p printer -o job-quota-period=604800 -o job-k-
limit=1024
# /usr/sbin/lpadmin -p printer -o job-quota-period=604800 -o job-
page-limit=100
```

Kontrola pristupa štampaču

Komandom lpadmin -u određuje se koji korisnik sme da dobije pristup štampaču. Osnovna konfiguracija omogućuje svim korisnicima pristup štampaču:

```
# /usr/sbin/lpadmin -p printer -u allow:all
```

CUPS podržava liste dozvole i zabrane pristupa. Zajedno sa listom korisnika, potrebno je specificirati da li ti korisnici imaju ili nemaju pristup štampaču.

```
# /usr/sbin/lpadmin -p printer -u allow:nmacek,jsmith,bora
```

Ova komanda dozvoljava korisnicima nmacek, jsmith i bora da štampaju na navedenom štampaču, ali niko drugi ne može. Sledeća komanda ima suprotan efekat: svi korisnici osim korisnika josh, nick, mark, trey i dave moći će da štampaju na navedenom štampaču.

```
# /usr/sbin/lpadmin -p printer -u deny:josh,nick,trey,mark,dave
```

Pristup štampaču može se kontrolisati i putem UNIX grupa, postavljajući karakter "@" ispred imena svake grupe. Komanda:

```
# /usr/sbin/lpadmin -p printer -u allow:jsmith,@printerusers
```

dozvoljava korisniku jsmith i svim članovima grupe printerusers da štampaju na gore navedenom štampaču.

NAPOMENA: naredbe allow i deny nisu kumulativne. To znači da se svaki put prilikom izmene kontrole pristupa mora navesti kompletna lista korisnika kojima se pristup dozvoljava ili zabranjuje. Takođe, CUPS podržava rad sa samo jednom listom korisnika - listom koja ili dozvoljava ili zabranjuje korisnicima štampanje. Ako se postave obe liste, samo druga lista ostaje validna.

Klase štampača

CUPS podržava koncept klasa štampača (printer classes). Poslovi poslati klasi štampaju se na prvom slobodnom štampaču te klase. Klase same po sebi mogu biti članovi nekih drugih klasa, tako da je moguće definisati velike skupove štampača kojima se omogućava štampanje u svakom trenutku, sa vrlo malim čekanjem u redu za štampu.

Upravljanje klasama štampača

Za administraciju klasa štampača koristi se komanda lpadmin. Sledeće tri komande redom ilustruju:

- dodavanje štampača u klasu (ukoliko klasa ne postoji, biće kreirana)

```
# /usr/sbin/lpadmin -p printer -c class
```

- uklanjanje štampača iz klase

```
# /usr/sbin/lpadmin -p printer -r class
```

- brisanje klase štampača

```
# /usr/sbin/lpadmin -x class
```

Implicitne klase

CUPS takođe podržava koncept implicitnih klasa (implicit classes). Implicitne klase funkcionišu poput štampačkih klasa, ali se kreiraju same ovisno o dostupnosti određenih štampača i njihovih klasa. Implicitne klase omogućavaju postavljanje više štampačkih servera sa identičnim postavkama za štampanje, tako da klijent bude u mogućnosti da pošalje svoj posao prvom slobodnom serveru. Ako se jedan ili više servera sruše zadaci štampanja biće prosleđeni onom serveru koji radi, omogućavajući tako sigurno štampanje.

Kao šta je ranije pomenuto, implicitne klase se automatski kreiraju na osnovu dostupnih štampača i klasa. Ova funkcija se može onemogućiti postavljanjem vrednosti off direktive ImplicitClasses u cupsd.conf datoteci.

Konfigurisanje klijenata

Klijent je bilo koji računar sa kog korisnici šalju zahteve za štampu ka print serveru. Klijent može da bude i server ukoliko on direktno komunicira sa nekim od svojih štampača. CUPS podržava više metoda konfiguracije klijentskih računara.

Najkomplikovaniji način konfiguracije klijenata je pojedinačno dodavanje svakog štampača na kom se sa tog računara može štampati:

```
# lpadmin -p printer -E -v ipp://srvname/printers/printername
```

Parametri srvname i printername su ime (hostname) ili IP adresa servera za štampu i ime štampača koji je povezan na taj server. Ovaj način konfigurisanja nije preporučljiv ukoliko postoji veliki broj klijenata (administrator ne treba da unosi dnevno 2000 istih komandi ukoliko isti posao može da odradi pomoću shell programa ili na neki drugi način). Ovako konfigurisani klijenti imaju lokalne redove za štampače.

CUPS se može konfigurisati tako da radi bez lokalnog reda za štampač, i u tom slučaju se svi poslovi za štampu šalju centralizovanom print serveru. Ako se taj server sruši, štampanje je onemogućeno. Ova mogućnost CUPS-a aktivira se dodavanjem linije:

```
ServerName srvname
```

u datoteku /etc/cups/client.conf (ukoliko datoteka ne postoji potrebno je najpre kreirati, a zatim u nju upisati liniju). Parametar srvname može biti ime ili IP adresa primarnog servera za štampu.

Primarni print server takođe se može odrediti za svakog pojedinačnog korisnika. U tom slučaju, potrebno je dodati liniju ServerName srvname u datoteku ~/.cupsrc.

Automatsko konfigurisanje klijenata

CUPS podržava automatsko konfigurisanje klijenata na istoj podmreži. Print serveri u određenim intervalima šalju informacije o dostupnim štampačima i klasama štampača koje se automatski ažuriraju na klijentima. Ovaj način konfiguracije je najjednostavniji: svaki klijent će videti listu dostupnih štampača. Kompletno konfigurisanje automatske detekcije štampača vrši se direktivama BrowseAddress, BrowseInterval i BrowseTimeout u datoteci /etc/cups/cupsd.conf. Podrazumevano, informacije o štampačima šalju se svakih 30 sekundi. Iako ove štampačke informacije ne zauzimaju mnogo prostora (obično oko 80 bajtova po štampaču), u slučaju velikog broja servera i štampača ovo slanje može opteretiti mrežu, tako da se interval može uvećati direktivom BrowseInterval.

U podrazumevanom stanju, klijenti vide štampače sa iste podmreže. Podmreže (subnet) se upotrebljavaju za minimiziranje količine prometa koja se odvija u pozadini. Dodatno, klijentima se može omogućiti da vide štampače unutar drugih podmreža višestrukim direktivama BrowsePoll u datoteci /etc/cups/cupsd.conf. Na ovaj način se inicira provera statusa - polovanje (polling) većeg broja print servera koji se mogu nalaziti na različitim podmrežama.

Količina poll upita može se dodatno ograničiti: dovoljno je da jedan klijent poluje štampače na drugim podmrežama, a zatim direktivom BrowseRelay emituje informacije o štampačima svim klijentima koji se nalaze na istoj podmreži.

CUPS konfiguracione datoteke

CUPS se konfiguriše pomoću tekstualnih datoteka koje se nalaze u direktorijumu /etc/cups. U nastavku teksta dat je spisak značajnih konfiguracionih datoteka:

cupsd.conf	konfiguracija CUPS servisa (*/usr/sbin/cupsd*). Nakon izmene konfiguracione datoteke potrebno je ponovo pokrenuti CUPS servis inicijalizacionim skriptovima (/etc/init.d/cups) ili slanjem HUP signala odgovarajućem daemon procesu;

```
# /etc/init.d/cups restart
```

printers.conf	informacije o dostupnim lokalnim štampačima. Udaljeni štampači se ne navode u ovoj datoteci. Datoteka printers.conf se modifikuje komandom lpadmin ili preko Web interfejsa;
classes.conf	sadrži informacije o svakoj lokalnoj štampačkoj klasi. Udaljene i implicitne klase koje se automatski prijavljuju ne navode se u ovoj datoteci. Datoteka classes.conf modifikuje se komandom lpadmin ili preko Web interfejsa;
client.conf	klijent konfiguraciona datoteka. CUPS-ova klijentska aplikacija (lp, lpr) koristi /etc/cups/client.conf kao primarnu konfiguracionu datoteku. Klijentska aplikacija takođe proverava da li u home direktorijumukorisnika postoji datoteka .cupsrc;
mime.convs	lista standardnih konverzionih filtera;

mime.types	lista standardnih formata datoteka.

Štampanje sa ostalih sistema

CUPS je zasnovan na IPP protokolu, tako da svaki sistem koji podržava IPP može automatski da šalje poslove CUPS serveru ili prima poslove CUPS klijenata.

Podrška za LPD klijente

CUPS pruža podršku uz ograničenu funkcionalnost LPD baziranim klijentima. Korisnici koji rade na sistemima na kojima je instaliran LPD (Line Printer Daemon) mogu štampati datoteke na udaljenom CUPS serveru, izlistati status poslova na čekanju i ukloniti nepoželjan posao iz reda. LPD ne podržava automatsku konfiguraciju klijenata tako da se svaki klijent mora ručno konfigurisati. Dodatno, LPD ne podržava veliki broj CUPS-karakterističnih opcija štampača i filtriranje.

Program cups-lpd prihvata zahteve za štampu LPD klijenata preko inetd ili xinetd wrapper programa. Na sistemima na kojima se koristi inetd podrška za LPD klijente se uključuje dodavanjem sledeće linije u /etc/inetd.conf konfiguracionu datoteku:

```
printer stream tcp nowait lp /usr/lib/cups/daemon/cups-lpd cups-lpd
```

Putanja cups-lpd programa može da varira sa konkretnom instalacijom CUPS-a. Nakon toga potrebno je restartovati cupsd slanjem HUP signala ili /etc/init.d/cups skriptom.

```
# killall -HUP inetd
```

Ukoliko je na sistemu aktivan xinetd wrapper daemon potrebno je kreirati datoteku /etc/xinetd.d/printer i u nju upisati sledeće linije:

```
service printer
{
socket_type = stream
protocol = tcp
wait = no
user = lp
server = /usr/lib/cups/daemon/cups-lpd
}
```

Pogram xinetd automatski čita novu konfiguracionu datoteku i omogućava podršku za štampanje LPD klijentima.

Program cups-lpd ne vrši nikakvu kontrolu pristupa baziranu na postavkama u cupsd.conf, hosts.allow ili hosts.deny datotekama koje koristi TCP wrapper. Pokretanjem cups-lpd na serveru omogućava se svakom LPD klijentu iz lokalne mreže (ili Interneta, ukoliko je rutiranje loše konfigurisano) da štampa preko vašeg servera.

Štampanje na LPD serverima

CUPS pruža lpd backend za štampanje na LPD serverima i štampačima - dovoljno je da se upotrebi URI uređaj lpd://srvname/printername, gde je parametar srvname ime ili IP adresa print servera, a parametar printername ime štampača.

CUPS i Windows

CUPS podrška za Windows klijente realizuje se pomoću programskog paketa SAMBA (www.samba.org), počev od verzija SAMBA 2.0.6. U datoteku smb.conf potrebno je upisati sledeće linije:

```
printing = cups
printcap name = cups
```

Nakon toga korisnici MS Windows sistema moći će da štampaju na CUPS serveru.

CUPS klijent može da štampa na Windows serverima korišćenjem LPD protokola CUPS sistema i TCP/IP Printing servisa na Windows sistemu. Drugi način je korišćenje Microsoft Server Message Block protokola (SMB), za koji podršku obezbeđuje programski paket SAMBA.

9

ARHIVIRANJE I BACKUP

Gubitak podataka koji su danima unošeni, formirani, modifikovani i održavani predstavlja veoma nezahvalnu situaciju za svakog korisnika računara (dodatno, izaziva stresne, frustrirajuće situacije). Ozbiljne greške koje dovode do gubitaka informacija u datotekama, ili do gubitaka celih datoteka, mogu se javiti čak i na savremenim, sofisticiranim sistemima datoteka koji raspolažu velikim brojem opcija za uvećavanje pouzdanosti. Najčešći uzroci gubitka podataka su:

- ljudski faktor, kao što je, na primer, nehotično brisanje datoteke,
- razni oblici neregularnog rušenja operativnog sistema, poput nestanka struje, koji dovode sisteme datoteka u nekonzistentno stanje,
- fizičko oštećenje medijuma, koje za posledicu najčeće ima bespovratan gubitak podataka.

Kao što se vidi, broj potencijalnih uzroka gubitka podataka je veliki, tako da je za ozbiljne korisnike neophodno oformiti rezervnu kopiju značajnih podataka. U daljem tekstu se uvode dva pojma vezana za kreiranje rezervnih kopija podataka - arhiva (archive) i rezervna kopija podataka (backup).

Strategije kreiranja rezervnih kopija podataka

Pojam arhive. Razlike između arhiva i sistema datoteka. Rezervne kopije podataka (backup). Puni i inkrementalni backup. Prosta šema i višeslojne kopije.

Iako su na prvi pogled slični postupci koji se mogu izvršiti istim komandama, backup i arhiviranje se razlikuju u tome ko ih izvršava, s kojim ciljem i koje datoteke proces obuhvata. Arhive kreiraju korisnici sistema s ciljem da sačuvaju lične datoteke, odnosno da naprave presek situacije u značajnim vremenskim trenucima. Arhiviranje obuhvata korisničke podatke, odnosno datoteke značajne za korisnike. Za razliku od arhiva, backup

uključuje čitave sisteme datoteka ili fundamentalne delove operativnog sistema (na primer, sistemske direktorijume), a kreiraju ga sistem administratori s ciljem očuvanja integriteta sistema ili njegovih funkcionalnih celina.

Arhive

Arhiviranje je proces kreiranja kopije značajnih datoteka na drugom medijumu u značajnim trenucima vremena. Arhiviranje je preventivna mera - u slučaju havarije sistema, korisnici sa arhivnim kopijama značajnih datoteka osobođeni su frustracije gubitka podataka i vremena koje bi utrošili za njihovo novo kreiranje. Dodatno, ukoliko su značajne datoteke arhivirane, uklanjanje suvišnih datoteka sa diska je relativno lak i brz proces - svi korisnički podaci mogu se izbrisati, nakon čega se korisni jednostavno prenose sa arhiva u sistem datoteka. Arhive se mogu koristiti i za organizaciju softvera i dokumenata za javni i ograničeni pristup, najčešće putem Interneta. Zahvaljujući javno dostupnim programskim paketima i defacto standardizovanim formatima arhiva, ova procedura je jednostavna i kao takvu je mogu koristiti svi korisnici računara.

Poređenje arhiva i sistema datoteka

Jedina sličnost između sistema datoteka i arhiva je u njihovom sadržaju - obe strukture sadrže iste vrste objekata, odnosno datoteke i direktorijume. Posmatrano sa svih ostalih aspekata (struktura podataka, broj i vrsta operacija nad objektima, metod pristupa) sistemi datoteka i arhive su različiti.

Pre svega sistem datoteka je najbrža, najkvalitetnija i najkompleksnija struktura podataka jednog operativnog sistema, koja podržava veliki broj operacija nad svojim objektima. Arhiva je relativno prosta struktura podataka koja obično podržava dve fundametalne operacije - dodavanje datoteka u arhivu i povratak (extraction) datoteka iz arhive. Složene operacije nad datotekama nikad se ne vrše u arhivi, nego u sistemu datoteka. Arhiva obavlja jednostavnu ali korisnu funkciju - služi da vrati izgubljene podatke u sistem datoteka. Za razliku od arhive sistem datoteka mora sadržati dinamički izmenljive kontrolne podatke (metadata) i obavezno se mora kreirati pre upotrebe, što u slučaju arhiva nije neophodno. Dodatno, pristup podacima u sistemu datoteka je direktan: nove datoteke mogu se dodati na bilo koje mesto, pod uslovom da ima dovoljno prostora, a postojeće datoteke mogu se lako obrisati. Pristup podacima u arhivi je strogo sekvencijalan: nove datoteke mogu se dodati isključivo na kraj arhive, a postojeće je teško ili nemoguće obrisati.

Backup

Pod terminom kopije podataka podrazumeva se postupak memorisanja čitavog sistema datoteka ili fundamentalnih direktorijima i datoteka. Backup treba da obuhvati podatke koji obezbeđuju integritet celog sistema ili neke funkcionalne celine, kao što je na primer baza podataka ili sistem za elektronsku poštu.

Backup obično obuhvata veliku količinu podataka, tako da je kreiranje iste relativno dug proces, i kao takav se zbog nedostatka motivacije ne radi se često (izuzetak predstavlja relativno kratak period nakon havarije sistema, u kom ljudi shvataju značaj kopije podataka, ali ga i brzo zaboravljaju). U svakom slučaju, imati bilo koju kopiju datoteke je mnogo bolje nego nemati nikakvu. Dobri sistem administratori automatizuju backup, odnosno deklarišu koje podatke treba arhivirati i vreme kreiranja backup kopije, nakon čega planer poslova (cron) pokreće odgovarajući program za backup. Po pravilu, optimalna strategija predstavlja kombinaciju punog (full) i inkrementalnog (incremental) backupa, čime se obezbeđuje potpun integritet podataka, a vreme potrebno za kreiranje kopije i broj upotrebljenih medijuma minimizira.

Prosta šema

Ova šema na jednostavan način kombinuje dve gore navedene metode: najpre se napravi kopija svih podataka (full backup), koja je vremenski intenzivna i zahteva po pravilu veliki broj medijuma, a zatim se u više navrata metodom inkrementalnog backupa arhiviraju samo datoteke koje su promenjene u periodu od zadnjeg arhiviranja. Poređenje datoteka se po pravilu vrši upoređivanjem vremena zadnje modifikacije datoteka i njihovih kopija u arhivi, odnosno vremena zadnje modifikacije arhive. Inkrementalna metoda po pravilu relativno kratko traje i zahteva manji broj medijuma.

Višeslojna kopija

Prosta metoda je pogodna za ličnu upotrebu i servere sa manjom količinom podataka. Za velike servere, odnosno velike količine podataka, metoda višeslojne kopije (multilevel backup) je mnogo pogodnija. Prosta metoda sa dva nivoa kopije može se generalizovati uvođenjem dodatnih inkrementalnih nivoa, pri čemu pun backup predstavlja nivo 0, a različiti nivoi inkrementalnih kopija nivoe 1, 2, 3 itd (niži nivo podrazumeva veću količinu podataka). Na nivou svake inkrementalne kopije, kopiraju se samo one datoteke koje su promenjene u periodu od poslednjeg kreiranja kopije istog ili višeg nivoa, čime se postiže duža istorija arhiviranja. Tipičan primer je sedmodnevna kopija - ponedeljkom se kreira puna kopija (level 0), a zatim se svaki dan vrši inkrementalni backup onih datoteka koje su promenjene u toku tog ili prethodnog dana (što zavisi od vremena u koje je backup zakazan). Duža istorija arhiviranja je pogodna za upotrebu u sistemima za rad sa bazama podataka i transakcijskim sistemima u kojima je potrebno sistem vratiti na neko prethodno stanje radi otkrivanja i uklanjanja grešaka. Vreme restauracije sistema datoteka ili baze podataka se pomoću višeslojne kopije može svesti na minimum.

Koje podatke treba uvrstiti u backup ?

Po pravilu, treba praviti kopiju podataka koji su unikatni, značajni, ili se ne mogu rekonstruisati, što obuhvata:

- korisničke podatke koje korisnici ne mogu sami arhivirati,
- aplikacije koje nije jednostavno reinstalirati,
- baze podataka,

- sistemske konfiguracione datoteke (najčešće se nalaze se na /etc direktorijumu, ali mogu biti i na drugim sistemskim direktorijumima, kao što su instalacioni direktorijum neke aplikacije ili home direktorijumi korisnika). Dodatno, sve korisnike treba obučiti za rad sa programom za arhiviranje i objasniti im značaj arhiviranja podataka.

Backup ne treba da obuhvati :

- softver koji se može lako reinstalirati, osim ako sadrži složene konfiguracione datoteke,
- datoteke sa /proc direktorijuma, zato što te kvazi-datoteke kernel generiše automatski (naročito ne treba arhivirati datoteku /proc/kcore, koja predstavlja sliku RAM memorije),
- velike log datoteke koje opisuju statistiku aktivnosti sistema,
- spool direktorijume.

Komprimovane kopije podataka

Kopije podataka u velikim sistemima zahtevaju veliki broj medijuma i dosta vremena za kreiranje i održavanje. Broj medijuma, odnosno veličina kopije može se smanjiti ukoliko se podaci prvo komprimuju, a zatim arhiviraju. Kritične komponente koje utiču na brzinu kompresije su procesorska snaga, koja je u većini slučajeva dovoljna, i algoritmi za kompresiju, koji su takođe sve bolji, što opravdava korišćenje kompresije. Dodatno, u neke programe za arhiviranje ugrađena je podrška za kompresiju (na primer, –z opcija GNU tar komande aktivira kompresiju arhive gzip programom).

Komprimovana kopija podataka može se realizovati na nekoliko načina koji se razlikuju po procesorskoj zahtevnosti, veličini arhive, i pouzdanosti.. U nastavku teksta opisane su dve osnovne metode,

Prva metoda - kompletna komprimovana kopija (koju, na primer koristi program tar), realizuje se najpre kreiranjem kompletne kopije, a zatim komprimovanjem iste u celini i upisivanjem na medijume. Osnovni nedostatak ove metode je pouzdanost: ako program za kompresiju ne koristi algoritme za korekciju grešaka, podaci koji se u arhivi nalaze ne mogu se rekonstruisati ukoliko se nalaze iza bita koji je pogrešan. Ovi algoritmi takođe nisu svemogući - u slučaju većeg oštećenja arhive rekonstrukcija podataka nije moguća, a sve datoteke koje se nalaze iza greške koja ne može da se koriguje postaju neupotrebljive.

Druga metoda, koju koristi program afio, predstavlja kopiju komprimovanih datoteka, gde se najpre svaka datoteka zasebno komprimuje, a zatim se kreira kopija tih datoteka koja se upisuje na medijume. Pouzdanost ove metode je znatno veća - svaka datoteka se komprimuje nezavisno, tako da ukoliko dođe do greške u kopiji samo će određene datoteke biti neupotrebljive, dok su ostale raspoložive.

Linux programi za backup i arhiviranje

tar (Tape Archiver). cpio (Copy in and out). Sintaksa, režimi rada i primeri upotrebe. Kompresija datoteka.

Deo svakog ozbiljnog operativnog sistema su programi za arhiviranje i kreiranje rezervnih kopija podataka. Tradicionalni UNIX programi koji se koriste u te svrhe su: tar (tape archiver) i cpio (copy in and out), koji postoje na svim UNIX sistemima, a osim njih se mogu koristiti razni besplatni ili komercijalni softverski paketi različitih prozvođača (KBackup, Amanda). Rezervne kopije mogu se kreirati na raznim uređajima i odgovarajućim medijumima, poput izmenljivih magnetnih diskova, optičkih diskova, CD-RW i DVD-RAM uređaja, magnetnih traka, itd. Umrežavanje uvodi novi koncept arhiviranja zasnovan na činjenici da se podaci mogu preko mreže arhivirati na različitim računarima, odnosno da diskovi različitih računara mogu predstavljati medijume za čuvanje kopija podataka. Izbor programskog paketa, odnosno komande za arhiviranje zavisi od izabranog koncepta arhiviranja i uređaja, odnosno medijuma koji se u te svrhe koristi.

Pomoću programa tar i cpio može se izvršiti arhiviranje kako pojedinačnih, tako i grupe datoteka. Obe komande opslužuju veliki broj raznovrsnih uređaja, odnosno sve uređaje koji se u te svrhe mogu iskoristiti, a koje podržava kernel. Relativno loš rad sa specijalnim datotekama, poput node datoteka, linkova i pipe datoteka je negativna osobina ovih komandi - na nekim UNIX sistemima može se desiti da se nakon normalnog početka arhiviranja specijalne datoteke ne kopiraju ili da se komanda "zaglavi", što je posebno karakteristično za tar arhiver. Podrška za arhiviranje specijalnih datoteka je doprinos Linux programera - koristeći tar i cpio na Linux sistemima se bez problema može napraviti kvalitetan backup ili arhiva koja uključuje i specijalne datoteke.

Pomoću komande dump može se napraviti kopija čitavih sistema datoteka. Dump pristupa sistemu datoteka direktno, zaobilazeći VFS, što ima svoje prednosti: sistem datoteka ne mora biti montiran, odnosno aktiviran, a i sam proces je brži. Glavni nedostatak dump komande je ograničenje na jedan tip sistema datoteka, preciznije na konkretni UNIX. Na primer dump Linux sistema može da radi samo sa ext2/ext3 sistemima datoteka, dok su arhive koje kreiraju tar ili cpio čitljive na bilo kom UNIX sistemu. Komanda dump direktno podržava nivoe arhiviranja (backup levels), što se u slučaju tar ili cpio komande može postići samo u sprezi sa drugim komandama.

tar (tape archiver)

Komanda tar je jedna od najjednostavnijih i najčešće korišćenih komandi za arhiviranje podataka. Ime tar (tape archiver) potiče od nekada de-facto standardnog uređaja za arhiviranje - magnetne trake. Iako je u osnovi kreiran uglavnom za rad sa magnetnom trakom tar uspešno radi i sa ostalim uređajima poput disketa, kao i sa arhivama u okviru sistema datoteka (ukoliko je disk privremeni medijum za smeštaj arhive). Na Linux sistemima prisutna je napredna verzija GNU tar, u koju je uključen veliki broj novih mogućnosti koje nisu prisutne u klasičnom UNIX tar arhiveru. Među njima svakako treba

istaći mogućnost dodavanja datoteka na kraj postojeće arhive, inkrementalno, odnosno uslovno arhiviranje, komparaciju datoteka u arhivi i sistema datoteka i kompresiju arhive pomoću gzip ili compress programa. Jedna od najzahvalnijih dodatnih funkcija je ignorisanje grešaka prilikom čitanja arhive. Na starijim UNIX sistemima tar prekida rad kada naiđe na grešku prilikom čitanja arhive, čineći sve datoteke koje se u arhivi nalaze nakon tog mesta neupotrebljivim. Na Linux sistemu, tar će nastaviti s radom, tako da ostatak arhive može da se raspakuje.

Sintaksa i argumenti komande tar

Sintaksa komande tar je:

```
$ tar komanda [opcije] ime_arhive lista_datoteka
```

Jedan komandni argument je obavezan, a opcioni argumenti se navode ukoliko su potrebni (najčešće jesu). Kao i za većinu Linux komandi, tar koristi dva oblika komandnih argumenata: kraći, u vidu jednog slova kome prethodi znak -, i duži oblik sa punim nazivom, koji počinje sa --. Ime arhive može biti ime datoteke, ukoliko se arhiva kreira u vidu datoteke, ili nod za uređaj, ukoliko se datoteka kreira na izmenljivom medijumu. Lista datoteka se zadaje u jednostavnom obliku file1, file2, ... fileN, dir1, dir2, ... dirN, pri čemu je upotreba džoker karaktera dozvoljena.

Najvažniji komandni argumenti programa tar su:

-c kreiranje nove arhive,

-t listanje sadržaja arhive,

-x ekstrakcija datoteka iz arhive (extract).

U GNU tar programima, karakterističnim za Linux sisteme, postoje dodatne komande:

-u uslovno ažuriranje (update). Datoteka se dodaje na kraj arhive, samo ako je novijeg datuma u odnosu na postojeću kopiju datoteke u arhivi;

-r bezuslovno ažuriranje (append). Datoteka se dodaje na kraj arhive;

--delete brisanje datoteka iz arhive. Datoteka se u arhivi označi kao da je obrisana i dalje se ne koristi prilikom listanja ili ekstrakcije, iako fizički postoji u arhivi. Datoteka se ne može obrisati iz arhive koja se nalazi na magnetnim trakama;

-d poređenje, odnosno pronalaženje razlika između arhive i sistema datoteka;

-A konkatenacija, odnosno spajanje dve arhive u jednu.

Dodatno, često se koriste i opcioni argumenti koji preciznije definišu rad tar arhivera:

-f arch deklariše datoteku ili uređaj na kome treba kreirati arhivu, odnosno na kome se nalazi arhiva,

-h zadaje se arhiviranje originala a ne simboličkog linka (dereference),

-k	obezbeđuje čuvanje postojećih datoteka, prilikom ekstrakcije iz arhive ne uništavaju se postojeće datoteke,
-l	arhiviranje datoteka sa lokalnog sistema datoteka,
-M	kreira arhivu na više medijuma, obavezan za velike arhive koje prevazilaze kapacitet jednog medijuma (multi-volume archive),
-p	čuva prava pristupa datoteka,
-P	obezbeđuje apsolutnu putanju, odnosno ne briše vodeći /, koji se u podrazumevanom stanju uklanja iz putanje,
--remove-files	briše datoteke nakon dodavanja u arhivu,
--same-owner	kreira raspakovane datoteke sa istim vlasničkim odnosima,
-v	opširno opisuje status prilikom izvršenja tar komande,
-w	interaktivni rad (od korisnika se zahteva potvrda za svaku akciju),
--verify	provera ispravnosti arhive nakon upisa u nju,
- I filelist1	arhiviraju se datoteke pobrojane u datoteci FILE,
-X filelist2	datoteke pobrojane u datoteci FILE se ne arhiviraju,
-Z	uključuje kompresiju arhive pomoću programa compress,
-z	uključuje kompresiju arhive pomoću programa gzip,
--ignore-failed-read	u slučaju oštećenja arhive, tar komanda prestaje sa radom. Ukoliko je ova opcija navedena, arhiviranje se nastavlja nakon oštećenja.

Kreiranje arhive

Komanda tar prilikom kreiranja arhive zahteva dve informacije - ime arhive, koje predstavlja fizički uređaj ili datoteku na disku, i listu datoteka koje će biti uključene u tu arhivu, u obliku file1, file2, ... fileN, dir1, dir2, ... dirN. Veći broj datoteka se može specificirati upotrebom džoker karaktera ili navođenjem imena direktorijuma, čime se kompletan sadržaj tog direktorijuma uključuje u arhivu. Na primer, sledeće dve komande:

```
$ tar cvf /dev/rst0 myfile
$ tar cvf myfile.tar myfile
```

arhiviraju istu datoteku (myfile). Prva komanda upisuje arhivu na magnetnu traku, a druga komanda arhivu upisuje u datoteku pod imenom myfile.tar. Argumenat cfv ima sledeće značenje: c je kreiranje nove arhive, v obezbeđuje poruke o izvršenju komande na ekranu, a f specificira da je prvi sledeći argumenat ime tar arhive, odnosno uređaja. Preporučuje se da, ukoliko je arhiva realizovana u formi datoteke, ima ekstenziju .tar - to nije obavezno, ali je poželjno, jer veoma lepo opisuje tip datoteke.

Ako se datoteka koja sadrži tar arhivu šalje preko mreže, koristeći na primer ftp servise, poželjno je da se najpre obavi kompresija. Time se štedi prostor na disku i smanjuje

količina podataka koja se prenosi po mreži. Na Linux sistemima postoje dva programa koji se mogu koristiti u te svrhe, a koji se direktno mogu pozvati iz komande tar: compress i gzip. Ukoliko se kompresija koristi, arhivama je potrebno dati ekstenzije .tar.Z i .tar.gz,

Sledeći primer ilustruje kreiranje komprimovane tar arhive pomoću gzip programa:

```
$ tar cvf myfile.tar myfile
$ gzip myfile.tar
```

što je ekvivalentno sledećoj komandi:

```
$ tar cvfz myfile.tar.Z myfile
$ ls myfile*
myfile
myfile.tar.gz
```

Komanda tar na većini UNIX sistema omogućava korisnicima da pomoću opcija –I filelist1 (inlcude) i -X filelist2 (exlclude) specificiraju datoteke filelist1 i filelist2 u kojima su pobrojane datoteke koje treba uključiti ili izbaciti iz liste datoteka za arhiviranje. Sledeći primer ilustruje izbacivanje datoteka pobrojanih u exclude.list iz liste prilikom kreiranja arhive. Najpre se napravi tekstualna datoteka, exclude.list, po pravilu jedna linija - ime jedne datoteke.

```
$ cat exclude.list
exclude.list
file1
file2
tempdata
```

Zatim se tar komandi, pomoću argumenta X naglasi da iz liste izbaci sve datoteke koje su pobrojane u datoteci exclude.list. Poželjno je da ime datoteke sa listom za izbacivanje (u prethodnom primeru exclude.list) bude pobrojano u samoj datoteci, zato što ona ne treba da se arhivira.

```
$ tar cvfX mydata.tar exclude.list *
```

U ovom primeru arhiviraju se sve datoteke osim datoteka exclude.list, file1, file2 i tempdata, koje su pobrojane u datoteci exclude.list. Slično se u drugoj tekstualnoj datoteci može specificirati lista datoteka koje treba arhivirati, kao što je prikazano sledećim primerom:.

```
$ tar cvfX mydata.tar exclude.list –I include.list
```

U ovom primeru arhiviraju se sve datoteke čija su imena navedena u datoteci include.list, izuzimajući one datoteke čija su imena navedena u datoteci exclude.list. Ime arhive je mydata.tar.

Svaka opcija ima poziciju svog argumenta, f je prvi i specificira ime arhive (mydata.tar), X je drugi i specificira exclude.list, dok je I treći i specificira include.list. Imena datoteka za uključivanje ili izbacivanje iz liste su proizvoljna.

Pomoću tar arhivera u sprezi sa komandom find mogu se arhivirati datoteke na bazi raznih kriterijuma, kao što su imena, vreme poslednje modifikacije datoteka, veličina, vreme

poslednjeg pristupa i tako dalje. Sledeći primer ilustruje kreiranje arhive koju čine datoteke novije od lastmod.dat, pomoću komandi tar i find.

```
$ find –newer lastmod –print >> include.list
$ tar cvf mydata.tar –I include.list
```

Arhive koje kreira tar uključuju osim podataka i atribute datoteka kao što su vlasnički odnosi, prava pristupa i vremena koja opisuju pristupe i modifikacije. Svi atributi se verno prenose u tar arhivu. Kada se datoteke raspakuju iz arhive, svi atributi se verno prenose iz arhive u novoformiranu datoteku. Vlasnički i grupni atribut se memoriše u arhivi na osnovu numeričkih vrednosti UID i GID, što može dovesti do neprijatne situacije ukoliko se arhiva raspakuje na drugom UNIX sistemu - datoteka može dobiti pogrešnog vlasnika ili ostati bez vlasnika. Probleme ove vrste rešava superuser koji nelegalno vlasništvo može promeniti komandom chown.

Listanje sadržaja arhive i ekstrakcija datoteka

Listanje sadržaja arhive, bez ekstrakcije, postiže se upotrebom t komande, kao što je prikazano u sledećem primeru. Prva komanda, tar tf prikazuje imena datoteka koje se nalaze u arhivi. Druga komanda zadata je sa v flegom, tako da osim imena datoteka prikazuje i ostale atribute. Osim pregleda sadržaja arhive, listanje se koristi za određivanje punog imena datoteke koje je neophodno za ekstrakciju pojedinačnih datoteka iz arhive.

```
$ tar tf myfiles.tar
1.txt
2.txt
acme/
acme/1.txt
$ tar tvf myfiles.tar
-rw-rw-r-- nm/nm            20 2003-10-24 15:42:06 1.txt
-rw-rw-r-- nm/nm            25 2003-10-24 15:42:08 2.txt
drwxrwxr-x nm/nm             0 2003-10-24 11:08:17 acme/
-rwxrwxr-x nm/nm           125 2003-10-24 11:07:50 acme/1.txt
```

Za ekstrakciju datoteka iz arhive koristi se x komanda kao što je prikazano u sledećem primeru. Ukoliko se ne navede lista datoteka, vrši se ekstrakcija cele arhive. Navođenjem liste, mogu se iz arhive izvući pojedinačne datoteke, grupe datoteka i direktorijumi.

```
$ tar xvf myfiles.tar
```

Primeri korišćenja tar komande

Primer 1 (ažuriranje arhive i bezuslovno dodavanje datoteka - update i append). Na home direktorijumu treba napraviti tri datoteke (a.a, b.b i c.c) i upisati neki tekst u njih. Zatim se ove tri datoteke arhiviraju u simulacionu datoteku proba.tar. Nakon kreiranja arhive, izlistati sadržaj arhive pomoću tar tvf komande. Zatim se sa postojećom arhivom demonstrira komanda za ažuriranje u i komanda za dodavanje na kraj r. Najpre će se isprobati –u (update) komanda u dva slučaja: u prvom slučaju ukoliko su sve datoteke (a.a, b.b i c.c) nepromenjene, a u drugom slučaju ukoliko se promeni sadržaj samo jedne datoteke, na primer a.a. Potom će se isprobati r komanda, dodavanjem nepromenjene

datoteke b.b komandom –r (append). Obratititi pažnju gde se u arhivi nalaze nove datoteke.

Priprema datoteka i kreiranje arhive:

```
$ cd
$ ls -l / > a.a
$ cp /etc/passwd b.b
$ cp /etc/hosts c.c
$ tar cvf proba.tar a.a b.b c.c
a.a
b.b
c.c
```

Listanje sadržaja arhive:

```
$ tar tvf proba.tar
-rw-rw-r-- nm/nm          1376 2003-10-24 16:13:06 a.a
-rw-r--r-- nm/nm         12857 2003-10-24 16:12:53 b.b
-rw-r--r-- nm/nm           188 2003-10-24 16:12:59 c.c
```

Pokušaj ažuriranja arhive sa nepromenjenom datotekom:

```
$ tar uvf proba.tar a.a
```

Ukoliko se arhiva ažurira sa identičnom datotekom transfer se ne izvršava - u arhivi nije ništa promenjeno.

Pokušaj ažuriranja arhive datotekom sa promenjenim sadržajem:

```
$ cp /etc/fstab a.a
$ tar uvf proba.tar a.a
a.a
$ tar tvf proba.tar
# a.a sa starim sadržajem:
-rw-rw-r-- nm/nm          1376 2003-10-24 16:13:06 a.a
-rw-r--r-- nm/nm         12857 2003-10-24 16:12:53 b.b
-rw-r--r-- nm/nm           188 2003-10-24 16:12:59 c.c
# a.a sa novim sadržajem:
-rw-rw-r-- nm/nm           957 2003-10-24 16:22:14 a.a
```

Demonstracija bezuslovnog dodavanja datoteke u arhivu:

```
$ tar rvf proba.tar b.b
b.b
$ tar tvf proba.tar
-rw-rw-r-- nm/nm          1376 2003-10-24 16:13:06 a.a
# stara datoteka b.b:
-rw-r--r-- nm/nm         12857 2003-10-24 16:12:53 b.b
-rw-r--r-- nm/nm           188 2003-10-24 16:12:59 c.c
-rw-rw-r-- nm/nm           957 2003-10-24 16:22:14 a.a
# datoteka b.b koja je bezuslovno dodata:
-rw-r--r-- nm/nm         12857 2003-10-24 16:12:53 b.b
```

Demonstracija ekstrakcije datoteke iz arhive:

```
$ rm c.c
```

```
$ tar xvf proba.tar c.c
c.c
$ ls -l
-rw-rw-r--    1 nm       nm            957 okt 24 16:22 a.a
-rw-r--r--    1 nm       nm          12857 okt 24 16:12 b.b
-rw-r--r--    1 nm       nm            188 okt 24 16:12 c.c
-rw-rw-r--    1 nm       nm          40960 okt 24 16:30 proba.tar
```

Zaključak:

- datoteka se uvek dodaje isključivo na kraj arhive
- komanda -u dodaje datoteke pod uslovom da su izmenjene, dok -r uvek dodaje datoteke, bez obzira na to da li su izmenjene ili ne.

Primer 2 (ekstrakcija datoteka iz arhive). Prilikom ekstrakcije, ime datoteke mora da se navede onako kako je navedeno u arhivi, uključujući i putanju ukoliko je ima, što je demonstrirano sledećim primerom. Datoteka se zove a, ali se u arhivi nalazi sa putanjom svoje roditeljske grane dir1. Primer ilustruje bezuspešan pokušaj ekstrakcije datoteke a po imenu, i uspešan pokušaj ekstrakcije iste datoteke sa navođenjem putanje pre imena.

Priprema datoteka, kreiranje arhive i uklanjanje originalnih datoteka:

```
$ mkdir dir1
$ cp a.a dir1/a
$ cp b.b dir1/b
$ cp c.c dir1/c
$ tar cvf proba1.tar dir1/a dir1/b dir1/c
dir1/a
dir1/b
dir1/c
$ rm dir1/*
```

Neuspešna ekstrakcija:

```
$ tar xvf proba1.tar a
tar: a: Not found in archive
```

Uspešna ekstrakcija:

```
$ tar xvf proba1.tar dir1/a
dir1/a
$ ls -l dir1
-rw-rw-r--    1 nm       nm            957 okt 24 17:09 a
```

Primer 3 (arhiviranje celog direktorijuma). Za arhiviranje celog direktorijuma dovoljno je navesti ime direktorijuma u listi datoteka tar komande, a tar će arhivirati čitavo stablo po dubini sa svim pripadajućim datotekama. Za potrebe ovog primera potrebno je na home direktorijumu napraviti direktorijum dir1 i u njemu tri datoteke (a.a, b.b i c.c). Arhivirati ceo direktorijum dir1 u simulacionu datoteku arh1.tar. Obrisati dir1 sa home direktorijuma, a potom raspakovati arhivu arh1. Kreirati dir2 na home direktorijumu, preći na dir2 i u njemu raspakovati arhivu arh1. Pogledati gde su datoteke raspakovane.

Priprema direktorijuma i datoteka:

```
$ cd
$ mkdir dir1
$ cp /etc/fstab dir1/a.a
$ cp /etc/passwd dir1/b.b
$ cp /etc/hosts dir1/c.c
```

Arhiviranje celog direktorijuma :

```
$ tar cvf arh1 dir1
dir1/
dir1/a.a
dir1/b.b
dir1/c.c
```

Listanje sadržaja arhive:

```
$ tar tvf arh1
drwxrwxr-x nm/nm             0 2003-10-24 17:27:46 dir1/
-rw-r--r-- nm/nm           957 2003-10-24 17:27:25 dir1/a.a
-rw-r--r-- nm/nm         12857 2003-10-24 17:27:34 dir1/b.b
-rw-r--r-- nm/nm           188 2003-10-24 17:27:46 dir1/c.c
```

Brisanje originala:

```
$ rm dir1/*
```

Ekstrakcija celog direktorijuma:

```
$ tar xvf arh1 dir1
dir1/
dir1/a.a
dir1/b.b
dir1/c.c
```

Ekstrakcija celog direktorijuma na drugo mesto (dir2)

```
$ mk dir2
$ cd dir2
$ tar xvf arh1 dir1
dir1/
dir1/a.a
dir1/b.b
dir1/c.c
```

Pregled raspakovanih datoteka:

```
$ cd ..
$ ls -l
.:
arh1  dir1  dir2
./dir1:
a.a  b.b  c.c
./dir2:
dir1
./dir2/dir1:
a.a  b.b  c.c
```

Zaključak:

- prilikom ekstrakcije arhive treba obratiti pažnju na tekuću poziciju, odnosno tekući direktorijum. Direktorijumi koje arhiva sadrži biće kreirani u njemu, ukoliko tamo već ne postoje;
- u novonastale (i/ili postojeće) direktorijume biće raspakovane odgovarajuće datoteke.

Primer 4 (višemedijumska arhiva). Komanda tar zahteva da se flegom M eksplicitno naglasi uporeba većeg broja medijuma ukoliko je količina podataka velika. Dodatno se može specificirati i veličina medijuma upotrebom flega L.

```
# arhiva na disketama
$ tar cvfM /dev/fd0H1440 *
# arhiva to disketama – veličine medijuma 1.2MB)
$ tar cvfML /dev/fd0 1200 *
# extrakcija datoteka iz višemedijumske arhive
$ tar xvfM /dev/fd0H1440 *
```

Arhiviranje i kompresija

Prilikom skladištenja podataka na masovne memorijske medijume, datoteke određenih formata konzumiraju više mesta nego što je to potrebno. To se najbolje može pokazati na primeru tekstualnih datoteka koje se skladište po principu jedan ASCII karakter po jednom bajtu memorije. ASCII karakter se sastoji od sedam bitova, a kako većina masovnih memorijskih uređaja kao najmanju jedinicu upisa specificira bajt, gubitak memorije od 12,5% je očigledan. Daljim proučavanjem konačnih polja jezičkih simbola, može se zaključiti da se karakteri mogu kodirati u manjim grupama bitova, zavisno od frekvencije upotrebe. Takvim tehnikama kodiranja tekstualne datoteke se mogu komprimovati za čitavih 50%, što predstavlja značajno smanjenje veličine datoteke koje za sobom povlači efikasnije i brže skladištenje na memorijski medijum.

compress

Postoji nekoliko standardnih UNIX komandi koje su povezane sa kompresijom, a to su: compress, uncompress, zcat i zless:

compress	komprimuje datoteku (vrši kompresiju) u novu datoteku sa ekstenzijom .Z,
uncompress	vrši dekompresije datoteke, odnosno restaurira originalnu datoteku iz komprimovane datoteke,
zcat	privremeno dekompresuje datoteku i prikazuje njen sadržaj na standardnom izlazu pomoću programa cat,
zless	privremeno dekompresuje datoteku i prikazuje njen sadržaj na standardnom izlazu pomoću programa less (less preprocessor).

Sintakse ovih komandi su:

```
$ compress [ –fv ] file(s)
$ uncompress [ –v ] [ file(s) ]
$ zcat [ file(s)]
$ zless [ file(s)]
```

Od opcija pominjemo samo najznačajnije:

-f	(force) komprimovaće datoteku čak i u slučaju da se kompresijom dobija veća datoteka,
-v	prikazaće procenat redukcije za svaku komprimovanu datoteku.

gzip (GNU ZIP)

Program gzip vrši kompresiju datoteka pomoću Lempel-Ziv algoritma (LZ77), koji je takođe korišćen i u programu PKZip. Koeficijent kompresije zavisi od veličine, vrste i samog sadržaja datoteke. Na primer, veličina tekstualnih datoteka, kao što su izvorni kodovi ili tekst na engleskom jeziku, može se redukovati za 60-70%. Ovaj metod kompresije je znatno bolji od korišćenja LZW algoritma, Huffman-ovog kodiranja i adaptivnog Huffman-ovog kodiranja. Kompresija se izvodi čak i u slučajevima kada je komprimovana datoteka neznatno veća od originala. Najgori slučaj širenja datoteke je nekoliko bajtova za gzip zaglavlje, i pet bitova po bloku podataka od 32 kilobajta (razmera širenja iznosi 0.015% u slučaju velikih datoteka). Broj sistemskih blokova koje zauzima komprimovana datoteka skoro nikad nije veći od broja blokova koje zauzima original.

Nakon kompresije na disku će biti kreirana komprimovana datoteka sa ekstenzijom .gz. Komprimovanje čuva vlasničke odnose, prava pristupa i vreme poslednje modifikacije. Program gzip je sposoban da vrši kompresiju, ali ne i arhiviranje. Svaka datoteka se mora posebno komprimovati. Ukoliko je potrebno kreirati komprimovanu arhivu u kojoj se nalazi veći broj datoteka, gzip se koristi u sprezi sa programom tar. Kompimovane datoteke se mogu vratiti u originalni oblik pomoću programa gunzip, odnosno gzip -d. Program gunzip takođe vrši dekompresiju datoteka koje su kreirane programom compress, pri čemu je detekcija ulaznog formata automatska. Sadržaj komprimovanih tekstualnih datoteka se može pregledati pomoću programa zcat i zless bez prethodno obavljene dekompresije.

Sledeći primer ilustruje jednostavnu upotrebu programa gzip:

```
$ ls –l bigfile
-rw-r-----  1  root  root  17561600  jun 14 10:31  bigfile
$ gzip bigfile
$ ls –l bigfile.gz
-rw-r-----  1  root  root   6782471  jun 14 10:31  bigfile.gz
$ gunzip bigfile
$ ls –l bigfile
-rw-r-----  1  root  root  17561600  jun 14 10:31  bigfile
```

U nastavku teksta date su sintakse komandi gzip i gunzip i objašnjenje značajnijih opcija:

```
$ gzip [–d] [–lrtv] [–S suffix] [ filename ...  ]
$ gunzip [ –lrtv] [–S suffix] [ filename ...  ]
```

Opcije su:

-d	ukoliko se navede, gzip će pokušati da dekomprimuje datoteku filename;
-l	za svaku komprimovanu datoteku prikazuje sledeća polja: compressed size (veličinu kompresovane datoteke), uncompressed size (veličinu originalne datoteke), stepen kompresije, uncompressed name (ime originalne datoteke). Za datoteke koje nisu u gzip formatu navodi se -1 u polju uncompressed size;
-r	ukoliko uz ovu opciju kao filename navede ime direktorijuma, gzip će kompresovati sve datoteke koje se u tom direktorijumu nalaze. Komprimovane datoteke se smeštaju u direktorijum u kom se nalaze originali;
-S .suf	umesto -gz koristi se sufiks naznačen parametrom .suf. Svi sufiksi su dozvoljeni, ali ovu opciju radi preglednosti datoteka treba izbegavati;
-t	proverava se integritet kompresovane datoteke;
-v	gzip i gunzip prikazuju ime i procenat kompresije svake obrađene datoteke;
-#	određuje stepen kompresije. -1 (ili --fast) je najbrža kompresija sa najmanjim stepenom kompresije. -9 ili (--best) je najsporija kompresija sa najvećim stepenom kompresije. Podrazumevani stepen kompresije je 6. Kompresija velikih datoteka je procesorski, odnosno vremenski zahtevna procedura. Odredićemo vreme potrebno za kompresiju i stepen kompresije datoteke veličine 25MB korišćenjem opcija -1, -9 i -6:

```
$ ls -l file*
-rw-r--r--   1   root   root   25221120   jun 14   11:30   file1
-rw-r--r--   1   root   root   25221120   jun 14   11:30   file6
-rw-r--r--   1   root   root   25221120   jun 14   11:30   file9
$ time gzip -1 file1
real    0m23.007s
user    0m17.050s
sys     0m2.390s
$ time gzip -6 file6
real    0m29.551s
user    0m23.770s
sys     0m2.470s
$ time gzip -9 file9
real    0m49.836s
user    0m44.040s
sys     0m2.330s
$ gunzip -l file*
         compressed        uncompressed  ratio uncompressed_name
           15020033            25221120  40.4% file1
           14395450            25221120  42.9% file6
           14369780            25221120  43.0% file9
```

```
          43785263           75663360  42.1% (totals)
```

Primer sprege tar arhivera sa gzip i compress programima

Za potrebe ovog primera na home direktorijumu treba napraviti tri datoteke (a.a, b.b i c.c). Arhivirati ih u simulacionu datoteku proba.tar bez kompresije, potom u simulacionu datoteku proba.tar.Z uz korišćenje programa compress, i na kraju u simulacionu datoteku proba.tar.gz uz korišćenje programa gzip. Uporediti veličine novonastalih arhiva. Obrisati datoteke a.a, b.b i c.c i raspakovati jednu kompresovanu arhivu.

Priprema datoteka:

```
$ cd
$ cp /etc/fstab a.a
$ cp /etc/passwd b.b
$ cp /etc/hosts c.c
```

Kreiranje arhiva:

```
$ tar cf proba.tar a.a b.b c.c        # bez kompresije
$ tar cfZ proba.tar.Z a.a b.b c.c     # compress
$ tar cfz proba.tar.gz a.a b.b c.c    # gzip
$ ls -l
-rw-rw-r--    1 nm       nm      20480 okt 24 18:06 proba.tar
-rw-rw-r--    1 nm       nm       5399 okt 24 18:06 proba.tar.Z
-rw-rw-r--    1 nm       nm       4226 okt 24 18:06 proba.tar.gz
```

Ekstrakcija datoteka iz komprimovane arhive:

```
$ rm a.a b.b c.c
$ tar xvf proba.tar.Z             # pogrešno
tar: This does not look like a tar archive
$ tar xvfZ proba.tar.Z            # ispravno
$ tar xvf proba.tar.gz            # pogrešno
tar: This does not look like a tar archive
$ tar xvfz proba.tar.gz           # ispravno
```

Zaključak:

- kompresija smanjuje veličinu arhive, gzip je najbolji
- prilikom ekstrakcije datoteka iz arhive mora da se navede tip kompresije (Z – compress, z – gzip) koji je korišćen za kreiranje arhive. Ukoliko se to ne navede, tar smatra arhivu standardnom (nekompresovanom) i vraća poruku o grešci.

cpio (copy in and out)

cpio je jedna od najmoćnijih komandi za prenos datoteka na svim UNIX sistemima. Služi za arhiviranje datoteka u cpio formate arhiva, ali se može upotrebiti za prenos čitavih stabala datoteka sa jednog mesta na drugo. Komanda cpio ima tri različita načina rada. U izlaznom načinu rada (copy-out) komanda cpio arhivira datoteke, odnosno kopira ih u arhivu. U ulaznom režimu rada (copy-in) komanda cpio vrši ekstrakciju datoteke iz arhive,

a u posebnom obliku (copy-test) lista sadržaj cpio arhive. U prolaznom režimu rada (copy-pass) komanda cpio kopira datoteke sa jednog direktorijuma na drugi, kombinujući ulazni i izlazni režim rada bez realnog korišćenja arhive. Na Linux sistemima postoji GNU cpio program koji podržava i tar format za arhiviranje, čime se postiže kompatibilnost sa tar programom. Na UNIX sistemima tar format se ne može koristiti za specijalne datoteke (blok ili karakter uređaji), nego se koristi binarni cpio format koji je kompatibilan sa starijim cpio programima. Prilikom ekstrakcije iz arhive cpio komanda automatski prepoznaje koja je vrsta arhive prisutna i prilagođava se njoj.

Režimi rada i sintaksa cpio komande

Svaki režim rada ima specifičnu sintaksu. Za arhivske režime rada karakteristična je upotreba znakova > i < koji simbolišu smer ka arhivi i smer iz arhive. Lista datoteka nažalost nije jednostavna kao kod tar komande - lista se zadaje putem tekstualne datoteke, u kojoj je u svakoj liniji navedena jedna datoteka. Komanda cpio može se koristiti u sprezi sa UNIX komandama (find, ls), koje će kreirati listu datoteka za cpio komandu. Jedan argument (o, i, p) kojim se specificira režim rada je obavezan, a drugi argumenti su opcioni.

Sintaksa za izlazni režim rada koji kreira arhivu (copy-out mode) je:

```
$ cpio –o[opcije] >arhiva <lista
```

Treba primetiti da znak > ide uz ime arhive, što znači da se datoteke prenose u arhivu, a da znak < ide uz datoteku u kojoj se nalazi lista.

Sintaksa za ulazni režim rada u kom se vrši ekstrakcija datoteka iz arhive (copy-in mode) je:

```
$ cpio –i[opcije] <arhiva
```

Treba primetiti da znak < ide uz ime arhive, što znači da se datoteke prenose iz arhive, a da znaka < nema jer lista u ovom slučaju nije potrebna. Ukoliko se navedu pojedinačne datoteke onda će samo one biti raspakovane iz arhive. Specijalan oblik cpio naredbe (copy-test) koji lista sadržaj arhive je:

```
$ cpio –it[opcije] <archiva
```

Sintaksa za prolazni režim rada u kome se datoteke pobrojane u datoteci lista kopiraju u odredišni direktorijum (copy-pass mode), je :

```
$ cpio –p[opcije] /odredišni_direktorijum <lista
```

Osnovno što treba primetiti je da se umesto imena arhive pojavljuje ime odredišnog direktorijuma u UNIX stablu. Takođe, znak < ide uz listu datoteka.

Opcioni argumenti

-a vreme poslednjeg pristupa datotekama, koje komanda cpio čita radi ažuriranja, se resetuje;

-A	datoteke se dodaju na kraj arhive (opcija funkcioniše jedino u izlaznom režimu rada);
-B	veličina ulazno-izlaznog bloka postavlja se na 512x10=5120 bajtova. U protivnom, komanda radi sa blokom veličine 512 bajtova, što negativno utiče na performanse;
-c	opcija je identična sa "-H newc", pri arhiviranju se koristi novi prenosivi format (SVR4 portable format);
-C IO-SIZE	veličina ulazno-izlaznog bloka postavlja se na IO-SIZE bajtova;
-d	direktorijumi se kreiraju uvek kada je to potrebno;
-F file	zadaje se ime arhive, što je ekvivaletno sa >file;
-m	vreme poslednje modifikacije se čuva prilikom kreiranja novih datoteka;
-M MESS	poruka MESS se prikazuje kada se dostigne kraj medijuma i zahteva od korisnika da ubaci novi medijum. Ako argument MESS sadrži niz %d, tada se niz zamenjuje postojećim brojem medijuma koji treba ubaciti, a brojanje počinje od 1;
-t	copy-test mode, odnosno listanje sadržaja arhive (radi samo u ulaznom režimu);
-u	bezuslovno kopiranje. Koristi se u izlaznom i prolaznom režimu. Sa ovom opcijom do transfera dolazi bez obzira koja je datoteka novija;
-v	obezbeđuje prikazivanje statusa, odnosno prikazuje datoteke koje se kopiraju. Kada se koristi sa –t opcijom obezbeđuje prikaz sličan prikazu ls –l komande.

Specificiranje formata arhive

Pomoću –H ARC_FORMAT opcije specificira se format za arhiviranje. Podrazumevani format u izlaznom režimu rada (copy-out) je stari binarni format (bin), dok u ulaznom režimu (copy-in) komanda cpio automatski prepoznaje format arhive. Važeći formati su: bin (stari binarni format), newc (novi prenosivi (SRV4) format koji podržava sisteme datoteka sa više od 65536 i-node čvorova), crc (novi prenosivi (SRV4) format sa proverom ispravnosti zapisa), tar (stari tar format), ustar (POSIX.1 tar format). Takođe, cpio prepoznaje GNU tar arhive.

Kako se koristi cpio

U izlaznom režimu (copy-out mode), cpio kreira arhivu i kopira datoteke u nju. Sledeći primer ilustruje izlazni režim rada:

```
$ cpio –ocvB >arh1 <lista1
```

Komanda cpio izvršava se u izlaznom režimu, što obezbeđuje argument –o, arhiva se kreira u vidu datoteke pod imenom arh1, a zatim se u arhivu kopiraju sve datoteke navedene u listi, odnosno u datoteci lista1, pri čemu se na ekranu prikazuju poruke o

izvršenju komande (fleg v). Koristi se podrazumevani bin format, a veličina I/O bloka je 5120 bajtova (fleg B).

U ulaznom režimu rada (copy-in mode) program cpio lista arhivu ili kopira jednu ili više datoteka iz arhive u sistem datoteka. Sledeći primer ilustruje ekstrakciju svih datoteka iz arhive arh1:

```
$ cpio –iv <arh1
```

Komanda cpio se izvršava u ulaznom režimu, vrši se ekstrakcija svih datoteka iz arhive arh1, a na ekranu se prikazuju poruke o izvršenju komande. Sledeći primer ilustruje test režim, odnosno listanje sadržaja arhive što se obezbeđuje argumentima –it:

```
$ cpio –itv <arh1
```

Sledeći oblik se preporučuje kao potpuni oblik za pozivanje komande cpio u ulaznom režimu:

```
$ cpio –ic(t)vd(u)mB <arhiva
```

Flegovi t i u se zadaju po želji, a ostali argumenti se gotovo redovno koriste.

U prolaznom režimu rada (copy-pass mode) cpio kopira datoteke u odredišni direktorijum, pri čemu se direktorijumi kopiraju rekurzivno, odnosno po dubini. Sledeći primer ilustruje prolazni režim rada komande cpio:

```
$ cpio –pdumv /home/jsmith/new <lista2
```

U ovom primeru datoteke koje su pobrojane u datoteci lista2 kopiraju se na direktorijum /home/jsmith/new, kopiranje je bezuslovno (fleg u), komanda prikazuje poruke o izvršenju na ekranu (fleg v), a vreme poslednje modifikacije datoteka se čuva (fleg m).

Korišćenje pipe mehanizma - cpio u sprezi sa komandama ls i find

Kao što je već naglašeno lista datoteka za cpio se ne može navesti direktno u komandnoj liniji (kao kod tar komande - file1, file2, ... dir1, dir2), već isključivo u vidu tekstualne datoteke. Lista se može navesti u komandnoj liniji isključivo ako se cpio koristi u pipe sprezi sa komandama ls i find, gde ls i find kreiraju listu, a cpio na bazi zadate liste obavlja arhiviranje ili transfer.

```
# arhiviranje svih datoteka sa tekuće grane u arh1
$ ls | cpio –ocvB >arh1
# arhivira ceo direktorijum dir1 u arh1
$ find dir1 –print | cpio –ocvB >arh2
# alternativni oblik prethodne komande
$ find dir1 –cpio –ocvB >arh2
```

Primeri korićenja komande cpio

Primer 1 (Izlazni i ulazni režim rada - kreiranje arhive, listanje i ekstrakcija). Na home direktorijumu treba napraviti tri datoteke (a.a, b.b i c.c) i upisati neki tekst u njih. Zatim se kreira lista datoteke lista1 koja će sadržati te tri datoteke, a potom se na bazi liste obavi

arhiviranje u datoteku proba1. Nakon toga treba obrisati datoteke a.a, b.b, c.c i lista1. Arhivu treba testirati, a zatim je raspakovati.

Priprema datoteka:

```
$ cd
$ cp /etc/fstab a.a
$ cp /etc/passwd b.b
$ cp /etc/hosts c.c
```

Kreiranje liste:

```
$ cat >lista
/a.a
/b.b
/c.c
<CTRL-D>
```

Izlazni režim, odnosno kreiranje arhive:

```
$ cpio -ocvB >proba1 <lista1
a.a
b.b
c.c
3 blocks
```

Testiranje arhive:

```
$ cpio -itvB <proba1
-rw-r--r--   1 nm       nm            957 okt 25 18:18 a.a
-rw-r--r--   1 nm       nm          12857 okt 25 18:19 b.b
-rw-r--r--   1 nm       nm            188 okt 25 18:19 c.c
3 blocks
```

Ekstrakcija kompletne arhive:

```
$ rm a.a b.b c.c lista1
$ cpio -ivB <proba1
a.a
b.b
c.c
3 blocks
```

Primer 2 (korišćenje cpio u pipe sprezi sa komandama ls i find). Na home direktorijumu treba napraviti tri datoteke (a.a, b.b i c.c) i upisati neki tekst u njih. Datoteke treba arhivirati u arhivu proba2 pomoću pipeline mehanizma komande ls i cpio, a zatim u arhivu proba3 pomoću pipeline mehanizma komandi find i cpio.

Priprema datoteka:

```
$ cd
$ cp /etc/fstab a.a
$ cp /etc/passwd b.b
$ cp /etc/hosts c.c
```

Kreiranje arhive (pipeline ls i cpio):

```
$ ls | cpio -ocvB >proba1
a.a
b.b
c.c
3 blocks
```

Kreiranje arhive (pipeline find i cpio):

```
$ cd
$ find . -print | cpio -ocvB >proba3
a.a
b.b
c.c
3 blocks
```

Primer 3 (kopiranje direktorijuma u copy-pass režimu). Na home direktorijumu treba kreirati dva direktorijuma dir1 i dir2, a u direktorijumu dir1 poddirektorijum dir1/dir11. Zatim kreirati datoteke dir1/a.a i dir1/dir11/b.b. Kopirati u copy-pass režimu sadržaj direktorijuma dir1 u dir2 i rekurzivno izlistati sadržaj direktorijuma dir2.

Kreiranje direktorijuma i datoteka:

```
$ cd
$ mkdir -p dir1 dir2 dir1/dir11
$ cp /etc/fstab dir1/a.a
$ cp /etc/hosts dir1/dir11/b.b
$ cd dir1
```

Prolazni režim rada:

```
$ find * -print | cpio -pdum ../dir2
../dir2/a.a
../dir2/dir11
../dir2/dir11/b.b
27 blocks
```

Listanje dir2:

```
$ cd ../dir2
$ ls -R
.:
a.a  dir11
./dir11:
b.b
$ cd ..
```

Primer 4 (kopiranje jednog sistema datoteka na drugi). Primer demonstrira prenos celog sistema datoteka na drugi, pri čemu ukoliko postoji rekurzija izvorišnog i odredišnog direktorijuma u aktivnom UNIX stablu, treba koristiti –mount opciju, čime se sprečava pojavljivanje beskonačnih petlji (na primer ako je reč o kopiranju root sistema datoteka).

```
$ cd /usr
$ find . -mount -print | cpio -pduvm /copy_of_usr
```

dump i restore

Komanda dump se najčešće koristi za kreiranje kopija podataka (backup). Obično se koristi za kopiranje čitavog sistema datoteka, bilo u punom ili inkrementalnom obliku. Pored celih sistema datoteka, dump komanda može da arhivira pojedinačne datoteke, grupe datoteka i direktorijume. Komanda dump kreira kopije podataka u formatu koji uključuje informacije o direktorijumima, ali i notacije kada je sistem datoteka bio poslednji put montiran.

Komanda dump može pristupati i lokalnim i udaljenim uređajima, poput traka. Ako se koristi komanda tipa remote dump ime uređaja magnetne trake treba da ima prefiks u vidu imena udaljenog računara na kome se traka nalazi (na primer, nicotine:/dev/rst0). Na nekim UNIX sistemima postoji posebna komanda rdump, koja obavlja rad sa udaljenim uređajima.

Izlaz komande dump obezbeđuje dosta informacija o sistemu datoteka koji se arhivira, uključujući datum poslednjeg arhiviranja, nivo inkrementalnosti, ime sistema datoteka i mount-point direktorijum, zatim analizira sve direktorijume i datoteke koji će biti arhivirani i na osnovu toga procenjuje broj medijuma koji će biti potreban za backup.

Komanda dump koristi backup nivoe da ukaže da li je reč o punom ili inkrementalnom arhiviranju. Nivo 0 za komandu dump je puni backup, dok nivoi od 1do 9 predstavljaju inkrementalne kopije.

Komanda dump čuva u datoteci /etc/dumpdates informacije o datumu i nivou kopije koja je poslednja urađena, kao i svim uspešnim prethodno kreiranim kopijama u hronološkom redosledu. Neuspele kopije se ne upisuju u ovu datoteku.

Takozvane dump arhive mogu da se kreiraju ili na disku ili na nekom jeftinijem medijumu, kao što su trake ili optički diskovi. Kao i tar, dump kreira arhive u specifičnom formatu koji može kasnije da se iskoristi za ekstrakciju datoteka. Dobra osobina disk baziranih arhiva je u tome što su one uvek raspoložive, i proces za arhiviranje može da se automatizuje potpuno bez ljudske intervencije. Kod kreiranja arhiva na izmenljivim medijumima postoje procedure koje se ne mogu automatizovati, poput zamene medijuma i labelisanja, koje mogu dovesti do izvesnih grešaka prouzrokovanih ljudskim faktorom.

Komanda dump se može koristiti i za arhiviranje pojedinačnih datoteka i direktorijuma. U tom režimu, dump komanda radi uvek na nivou 0, odnonso obavlja se bezuslovno arhiviranje, bez obzira da li je datoteka već arhivirana ili nije.

Dump komandu je najbolje koristiti na sistemu datoteka koji nije aktiviran, odnosno na sistemu datoteka u kom nema otvorenih datoteka. To je poželjno zbog toga što neke promene u metadata području mogu biti nesaglasne sa stanjem u dump arhivi, tako da je krajnji rezultat loša arhiva iz koje ne mogu da se raspakuju sve datoteke.

Restauracija čitavog sistema datoteka iz arhive

Čitavi sistemi datoteka koji su prethodno arhivirani komandom dump, mogu se restaurirati komandom restore. Prilikom ekstrakcije pojedinačnih datoteka, interaktivni režim rada restore komande je pogodan za korišćenje. U tom režimu restore komanda ima interne

komande kao što su cd, ls tako da mogu da se vide datoteke koje su u dump arhivi, a pomoću komandi add select files i extract biraju se i ekstrahuju željene datoteke.

10

ADMINISTRACIJA PROCESA

Poznavanje procesa je neophodno za razumevanje suštine rada UNIX operativnih sistema. Svi programi se izvršavaju u vidu procesa kojima upravlja kernel. U ovom poglavlju najpre su date osnovne teorijske napomene koje se tiču procesa, a zatim su objašnjeni osnovni postupci iz oblasti administracije procesa.

Osnovne tehnike upravljanja procesima

Uvod u procese.Izvršavanje programa i kreiranje procesa. Prikazivanje procesa. Signali.

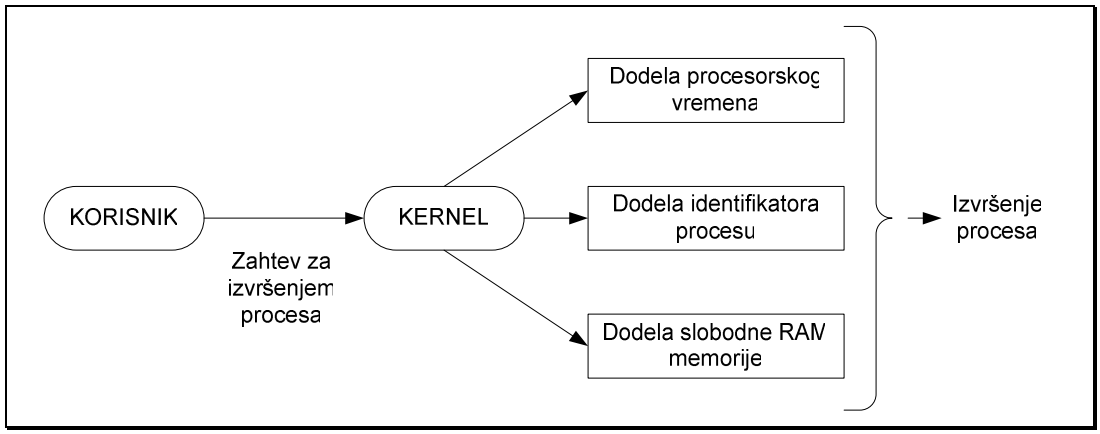
Upravljanje zadacima i poslovima se na UNIX sistemima vrši putem procesa. Procese mogu da pokreću korisnici ili sam operativni sistem.

Većina zadataka koji se izvršavaju pod UNIX-om pokreće neki proces, koji kasnije može pokrenuti podproces, čime se stvara hijerarhijska struktura proces-proces roditelj, slična hijerarhijskoj strukturi sistema datoteka. Novi proces se kreira ukoliko korisnik želi da koristi vi editor ili da pošalje datoteku na štampu pomoću komande lp - program koji je pokrenut kreira proces, čiji je proces roditelj komandni interpreter. Proces roditelj, u ovom slučaju komandni interpreter, obezbeđuje okolinu (environment) koja je neophodna za normalno izvršenje procesa, i prelazi u suspendovano stanje (WAIT) za vreme izvršenja podprocesa. Kada podproces normalno ili nasilno završi svoj rad (na primer, korisnik prekine njegovo izvršenje) proces roditelj preuzima kontrolu i komandni prompt se vraća korisniku. Ukoliko se podproces smesti u pozadinu (background), komandni interpreter ne čeka kraj izvršenja procesa u suspendovanom stanju. Komandni prompt se odmah vraća, a korisnik može zadati drugu komandu i na taj način pokrenuti drugi proces koji će se izvršavati paralelno ili kvazi-paralelno sa prethodnim. UNIX na taj način obezbeđuje višeprocesni rad. Jednostavne interne komande, poput cd, izvršava komandni interpreter - za njihovo izvršenje se ne kreira novi proces.

Svaki program koji korisnik pokrene kreira proces. Procesima se dodeljuju jednoznačni numerički identifikatori (PID, Process Identification Number), koje sistem dalje koristi za identifikaciju i praćenje procesa do kraja njegovog izvršenja.

Prva dva procesa koja se kreiraju pri podizanju System V sistema su sched (planer poslova, PID=0, ne postoji na Linux sistemima) i init (inicijalizacija, PID=1). Procesi sched i init su zaduženi za upravljanje ostalim procesima. Proces init je proces roditelj procesa login koji proverava unešeno korisničko ime i lozinku, a zatim pokreće proces shell, odnosno komandni interpreter. Init čeka u suspendovanom stanju da korisnik završi svoj rad u komandnom interpreteru i da se odjavi sa sistema, nakon čega opet preuzima kontrolu i pokreće novi login proces.

Kernel operativnog sistema upravlja pokretanjem i uništenjem procesa i dodelom resursa sistema procesima, poput procesorskog vremena i operativne memorije.



Slika 10.1 *Pokretanje procesa*

Zavisno od mesta u hijerarhiji procesa, načina izvršenja i trenutne funkcionalnosti, procesi mogu pripadati sledećim kategorijama:

- Daemon - procesi koje je pokrenuo kernel i koji se izvršavaju u pozadini. Na primer, lpd (line printer scheduler) se najčešće pokreće pri podizanju sistema s namenom da prihvata poslove za štampu i da njima upravlja. Ukoliko nema zahteva za štampu, lpd se izvršava ali je neaktivan. Kada neko pošalje zahtev za štampu lpd postaje aktivan dok se zahtev ne odštampa.
- Proces roditelj (Parent) - proces koji kreira podproces je proces roditelj. Proces roditelj obezbeđuje okolinu koja je neophodna za normalno izvršenje procesa, i može preći u suspendovano stanje za vreme izvršenja podprocesa. Osim procesa init svi procesi imaju svoje roditelje. Proces init pokreće login i njemu je proces roditelj. Login je podproces procesa init.
- Podproces (Child) - proces koji pokreće proces roditelj. Na primer, ukoliko korisnik u komandnom interpreteru bash zada komandu vi, onda je proces koji vi kreira podproces komandnog interpretera.
- Orphan - ukoliko korisnik pokrene komandu iz terminala u grafičkom okruženju i zatvori prozor pre nego što komanda završi svoj rad, proces koji je komanda kreirala postaje siroče (orphan), odnosno proces bez roditelja. Ovakvi procesi mogu nastati u svim situacijama kada se nasilno prekine izvršenje proces roditelja. Da bi se

održala hijerarhijska struktura procesa init kao glavni proces "usvaja" sve siročiće a zatim prekida njihovo izvršenje.

- Zombie (Defunct) - proces koji izgubi vezu s proces roditeljem ostaje izgubljen u sistemu, a jedini resurs koji troši jeste jedno mesto u tabeli procesa. Ovakav proces se ne može uništiti standardnim metodama. Defunct procesi ne utiču na pad performansi računara i uklanjaju se prilikom sledećeg podizanja operativnog sistema.

Kreiranje procesa i izvršenje programa

Prilikom pokretanja programa sistem kreira posebno okruženje koje je neophodno za izvršenje programa. Linux razdvaja operacije kreiranja novog procesa (fork) i izvršavanja programa u resursima novostvorenog procesa (exec).

Pod Linux operativnim sistemom svaki proces je u potpunosti opisan svojim identitetom (PID, akreditivi), okolinom koju nasleđuje od roditelja (argumenti i promenljive) i kontekstom, odnosno stanjem procesa u datom trenutku vremena (memorija koju proces koristi, tekući direktorijum, otvorene datoteke).

Kada korisnik zada komandu, na primer ls, izvršava se sistemski poziv fork koji kreira novi proces. fork izvršava sledeće operacije:

- alokaciju slobodnog mesta u tabeli procesa (tabela procesa je lista aktivnih procesa pomoću koje se ostvaruje kvazi-paralelno izvršenje većeg broja procesa),
- dodelu jedinstvenog identifikatora procesa (PID),
- kopiranje konteksta proces roditelja,
- slanje identifikatora proces roditelju, čime se ostvaruje direktna kontrola podprocesa.

Nakon toga Linux izvršava program ls u novokreiranom kontekstu. Komandni interpreter izvršava sistemski poziv exec programa ls, čime se program shell i prateći podaci menjaju programom ls i adekvatnim podacima. Nakon toga ls izvršava svoj zadatak, odnosno lista sadržaj tekućeg direktorijuma.

Dobijanje informacija o procesima

Prikazivanje procesa (komanda ps)

Komanda ps (process status) prikazuje na ekranu listu aktivnih procesa, odnosno PID procesa i ime komande kojom je proces iniciran. Na taj način se može dobiti PID procesa koji je blokiran ili dugo traje, te ga treba zaustaviti ili uništiti (komanda kill zahteva PID kao argument). Ako se komanda pokrene nekoliko puta može se jednostavnim upoređivanjem utrošenog procesorskog vremena utvrditi da li je proces aktivan ili ne. Ukoliko procesorsko vreme raste, proces je aktivan. Ukoliko se vreme ne menja, proces je najverovatnije završio svoj rad. Dodatno, mogu se ustanoviti roditeljski odnosi između procesa.

Sintaksa komande ps je:

```
$ ps [-options]
```

ps bez argumenata prikazuje: PID, tip terminala (TTY), utrošeno procesorsko vreme (TIME), i ime komande koja je inicirala proces (CMD) za sve procese koji se izvršavaju u tekućem shell kontekstu (ili Terminal prozoru, ukoliko se radi u X Windows sistemu).

```
$ ps
  PID TTY          TIME CMD
  739 pts/0    00:00:00 bash
  781 pts/0    00:00:00 ps
```

Tri osnovna argumenta komande ps su -e (every process), -f (full listing), i -u (user).

ps -e — ps prikazuje PID, TTY, TIME i CMD svih procesa na sistemu .

ps -f — ps prikazuje dodatne informacije o svim procesima koji su pokrenuti iz tekućeg shell konteksta ili Terminal prozora. U dodatne informacije spadaju: ID korisnika koji je pokrenuo komandu koja je inicirala proces (UID), identifikator proces roditelja (PPID), prioritet procesa (C) i vreme kada je proces počeo sa izvršenjem (STIME).

ps -u UID — ps prikazuje PID, TTY, TIME i CMD svih procesa koje je inicirao korisnik čiji je UID naveden kao parametar.

Komanda ps -ef kombinuje argumente -e i -f, odnosno prikazuje detaljne informacije o svim procesima na sistemu. Zbog velikog broja procesa na sistemu preporučuje se upotreba ps -ef komande u pipe sprezi sa komandom less.

U polju TTY procesa koje je inicirao korisnik koji je lokalno prijavljen na sistem i koji ne koristi grafičko okruženje stajaće tty#. U polju TTY procesa koje je inicirao korisnik koji radi u grafičkom okruženju ili je prijavljen na sistem preko mreže, stoji pts#. Pseudoterminal (pts) je ime uređaja dodeljeno remote login sesijama i prozorima. Svaki prozor koji korisnik otvori nakon prijavljivanja na sistem dobija novi pts#.

Kao što je ranije rečeno, da bi korisnik uništio neki proces neophodno je da poznaje njegov PID. Na većini sistema izvršava se veliki broj procesa, tako da je listing komande ps -ef dugačak. Ukoliko je ime komande kojom je proces iniciran poznato, PID se lako može odrediti korišćenjem ps u pipe sprezi sa komandom grep.

```
$ ps -ef | grep tuxracer
```

Komanda ps -ef prikazuje detaljne informacije o svakom procesu, uključujući i identifikator proces roditelja PPID. PPID se koristi u slučajevima kada nije dovoljno uništiti proces u kom se izvršava blokirana aplikacija, već i njen proces roditelj. Ukoliko se proces roditelj uništi prvi, svi podprocesi se automatski uništavaju.

Dodatni argumenti komande ps zavise od same distribucije UNIX, odnosno Linux sistema. Veoma koristan argument komande ps, prisutan između ostalih na Red Hat, SuSE i Debian Linux sistemima je --forest kojim se zahteva prikazivanje hijerarhijskog stabla procesa.

```
$ ps -e --forest
...
```

```
  792 ?        00:00:00 xinetd
 6239 ?        00:00:00  \_ in.telnetd
 6240 ?        00:00:00      \_ login
 6241 pts/0    00:00:00          \_ bash
 6328 pts/0    00:00:00              \_ ps
 6329 pts/0    00:00:00              \_ bash
  816 ?        00:00:02 lpd
...
 1108 tty5     00:00:00 mingetty
 1109 tty6     00:00:00 mingetty
18820 ?        00:00:00 login
18824 tty2     00:00:00  \_ bash
```

Određivanje vremena potrebnog za izvršenje procesa (komanda time)

Komandom time se određuje vreme potrebno za izvršenje komande, odnosno procesa. Komanda time na ekranu prikazuje tri vremena: realno (real), sistemsko (system) i korisničko (user). Realno vreme obuhvata interval od zadavanja komande do potpunog izvršenja i povratka komandnog prompta, uključujući i vreme čekanja na ulaz, izlaz i ostale događaje. Korisničko vreme je količina procesorskog vremena utrošena na samo izvršenje procesa. Sistemsko vreme je vreme koje je kernel utrošio na opsluživanje procesa.

```
# time ls -lR /etc | sort > /dev/null
real    0m4.332s
user    0m0.580s
sys     0m0.110s
```

Slanje signala i uništenje procesa

Signali

Zavisno od implementacije u svakom UNIX sistemu je definisano 30 do 40 signala, od kojih je svaki predstavljen imenom i brojem. Slanje signala je metod komunikacije sa procesima - signali se mogu posmatrati kao kratke poruke specijalnog značenja koje se šalju procesima, koje procesi dalje prihvataju ili ignorišu.

Signali se koriste za uništenje, privremeni prekid i nastavak izvršenja procesa. Na primer, kombinacija tastera <Ctrl-C> može da uništi proces koji više nije pod kontrolom. Kada korisnik pritisne <Ctrl-C> aktivnom procesu se šalje signal prekida INT, nakon čega se uništavaju aktivni proces i svi podprocesi koje je on inicirao.

Signali se mogu klasifikovati u dve osnovne kategorije:

- signali za kontrolu procesa, koji se mogu koristiti bez obzira na trenutni komandni interpreter,
- signali za kontrolu posla, koji se mogu koristiti samo ako komandni interpreter podržava kontrolu posla.

U signale za kontrolu procesa spadaju:

TERM	Uništenje procesa (terminate) - proces može da ignoriše ovaj signal
KILL	Neopozivo uništenje procesa - proces ne može da ignoriše ili blokira ovaj signal
HUP	Uništenje procesa koji se izvršavaju u pozadini prilikom odjavljivanja korisnika sa sistema (hang up)
INT	Interaktivni signal prekida (interrupt), koji generiše INTR kontrolni karakter
QUIT	Interaktivno uništenje procesa, koje generiše QUIT kontrolni karakter

Podrazumevana akcija koja se izvršava kao posledica signala za kontrolu procesa je uništenje procesa. Procesi mogu da ignorišu sve signale osim signala KILL, koji se koristi kao poslednja mera pri uništenju procesa. Ne može se tačno odrediti koji proces ignoriše koje signale.

U signale za kontrolu posla spadaju:

STOP	Zaustavljanje procesa - proces ne može da ignoriše ili blokira ovaj signal
CONT	Nastavi izvršenje zaustavljenog procesa - proces ne može da ignoriše ili blokira ovaj signal
TSTP	Interaktivno zaustavljanje procesa, koje generiše SUSP kontrolni karakter
TTIN	Posao u pozadini pokušava da izvrši akciju čitanja - grupa procesa je suspendovana
TTOU	Posao u pozadini pokušava da izvrši akciju upisa - grupa procesa je suspendovana

Podrazumevana akcija koja se izvršava kao posledica signala za kontrolu posla (izuzev signala CONT) je zaustavljanje, odnosno suspenzija procesa. Procesi mogu ignorisati sve signale osim signala STOP i CONT, tako da korisnik uvek može da zaustavi proces i nastavi izvršenje procesa.

Osim signala KILL i STOP, proces može da "uhvati" (catch) signal, odnosno da izvrši neku drugu akciju umesto podrazumevane. Proces koji hvata signal može da odluči koju će akciju da izvrši kao posledicu datog signala. Na primer, proces koji primi TERM signal može regularno da završi svoj rad (da najpre završi obradu podataka i upiše rezultate na disk, a zatim da prekine izvršenje). Ukoliko proces ne ignoriše ili ne hvata signal, izvršava se podrazumevana akcija.

Uništenje procesa

Pre ili kasnije korisnici će imati potrebu da prekinu izvršenje nekog programa. Razlozi za takvu akciju mogu biti pokretanje pogrešnog programa, zadavanje prave komande u pogrešno vreme, ili gubitak kontrole nad programom.

Izvršenje procesa koji radi u prvom planu (foreground) najlakše se može prekinuti slanjem signala prekida (INT). Podrazumevana kombinacija tastera kojom se signal INT šalje je <Ctrl-C>, što se može redefinisati komandom stty int karakter. Aktuelna kombinacija se može videti u izlazu komande stty -a.

```
$ stty -a
...
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol =
<undef>;
...
```

Proces može ignorisati INT signal, tako da ovaj način uništenja procesa ne uspeva u svim slučajevima. Alternativni način uništenja je slanje signala QUIT (signal za interaktivno uništenje procesa). Podrazumevana kombinacija tastera kojom se signal QUIT šalje je <Ctrl-\>, što se može redefinisati komandom stty quit karakter. Proces može ignorisati i ovaj signal. Ukoliko komandni interpreter podržava kontrolu posla, proces se može suspendovati, a zatim uništiti slanjem signala pomoću komande kill. Ukoliko se proces ne može uništiti ni na jedan prethodno pomenut način, korisnik mora da se prijavi na sistem sa druge virtuelne konzole, terminal prozora ili preko mreže, a zatim iskoristi komande ps i kill za uništenje procesa.

Procesi koji se izvršavaju u pozadini mogu se uništiti iz tekućeg komandnog interpretera.

Komanda kill

Komanda kill se koristi za slanje signala procesima. Najčešće se upotrebljava za uništenje procesa, a dodatno za zaustavljanje i nastavak izvršenja suspendovanog procesa. Signale procesima mogu da šalju vlasnici i superuser, što znači da samo onaj korisnik koji je pokrenuo komandu kojom je proces iniciran ima pravo da taj proces zaustavi ili uništi, dok root može da zaustavi ili uništi bilo koji proces na sistemu.

Pre slanja signala neophodno je identifikovati proces komandama ps ili jobs. Komanda ps je prisutna na svim UNIX sistemima, dok je komanda jobs dostupna samo u komandnim interpreterima koji podržavaju kontrolu posla. Uništenje procesa izvršava se u tri koraka:

- određivanje PID procesa koji je potrebno uništiti komandom ps. Napomena: ukoliko se umesto PID dobavi PPID procesa, biće uništeni svi procesi čiji je PPID proces roditelj;
- slanje signala procesu komandom kill. Komanda zahteva PID procesa kao argument, a dodatno se može navesti i signal koji je potrebno poslati. Signal se može navesti pomoću broja ili imena - preporučuje se navođenje putem imena, jer numeričke vrednosti variraju od sistema do sistema. Ukoliko se signal ne navede, podrazumeva se TERM;
- provera liste procesa, odnosno utvrđivanje da li je signal uništio proces ili ne. Nekoliko signala mogu se iskoristiti za uništenje procesa, od kojih neke procesi mogu ignorisati, pa je u nekim slučajevima potrebno više puta zadati komandu kill, odnosno poslati nekoliko različitih signala procesima. Na primer, ukoliko korisnik želi da uništi proces čiji je PID 2345, potrebno je da zada komandu kill -HUP 2345, i na taj način pošalje signal HUP procesu. Ukoliko se proces ne uništi korisnik može poslati signal TERM, a u krajnjem slučaju i signal KILL (kill -KILL 2345), koji sigurno i neopozivo uništava proces.

Ukoliko neki proces koji se izvršava u pozadini konzumira mnogo procesorskog vremena, a korisnik želi da izvrši neku akciju bez uništenja procesa, potrebno je da pošalje signal

STOP procesu (kill -STOP PID). Signal STOP privremeno zaustavlja izvršenje procesa, čime se procesor oslobađa, tako će se sledeća komanda koju korisnik zada brže izvršiti. Nakon toga, procesu se može poslati signal CONT (kill -CONT pid) čime se nastavlja njegovo izvršenje.

Sintaksa komande kill je:

```
$ kill [-s signal ] PID
```

Dodatno, komanda kill sa parametrom -l prikazuje listu signala:

```
$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     17) SIGCHLD
18) SIGCONT     19) SIGSTOP     20) SIGTSTP     21) SIGTTIN
22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO
30) SIGPWR      31) SIGSYS      32) SIGRTMIN    33) SIGRTMIN+1
34) SIGRTMIN+2  35) SIGRTMIN+3  36) SIGRTMIN+4  37) SIGRTMIN+5
38) SIGRTMIN+6  39) SIGRTMIN+7  40) SIGRTMIN+8  41) SIGRTMIN+9
42) SIGRTMIN+10 43) SIGRTMIN+11 44) SIGRTMIN+12 45) SIGRTMIN+13
46) SIGRTMIN+14 47) SIGRTMIN+15 48) SIGRTMAX-15 49) SIGRTMAX-14
50) SIGRTMAX-13 51) SIGRTMAX-12 52) SIGRTMAX-11 53) SIGRTMAX-10
54) SIGRTMAX-9  55) SIGRTMAX-8  56) SIGRTMAX-7  57) SIGRTMAX-6
58) SIGRTMAX-5  59) SIGRTMAX-4  60) SIGRTMAX-3  61) SIGRTMAX-2
62) SIGRTMAX-1  63) SIGRTMAX
```

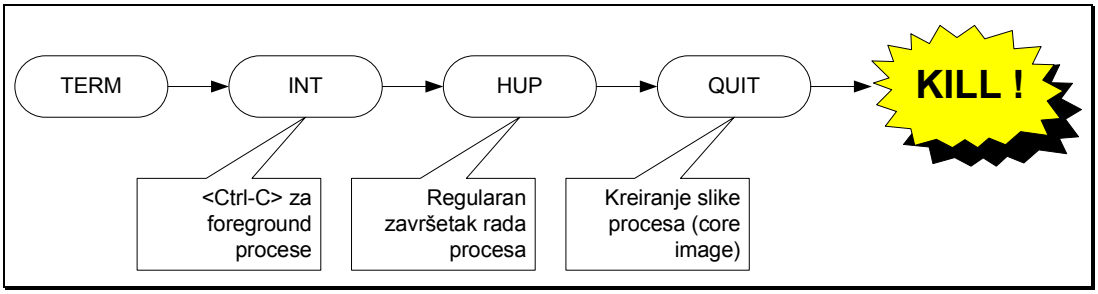
Dodatno, korisnik može koristiti komandu killall, koja funkcioniše kao i komanda kill, s tim što kao argument ne zahteva PID procesa, već njegovo ime:

```
$ killall -HUP inetd
```

Koji signal treba poslati procesu ?

Signal KILL je siguran način za uništenje procesa. Međutim, signal KILL ne vrši čisto uništenje procesa i kao takav se koristi kao poslednja alternativa. Proces ne može da "uhvati" signal KILL, tako da ne može da izvrši neku akciju pre uništenja, odnosno ne može regularno da završi svoj rad.

Proceduru uništenje procesa treba započeti signalom TERM, a zatim nastaviti signalom INT, ukoliko TERM ne funkcioniše. Ukoliko kontrolni karakteri koji izazivaju prekid ne funkcionišu, proces će verovatno ignorisati i INT signal. Signal koji većina procesa "hvata", nakon čega čisto regularno završava svoj rad je HUP - tako da uništenje procesa treba nastaviti slanjem tog signala. Ukoliko je korisnicima potrebna slika procesa (na primer, radi otkrivanja grešaka u projektovanju) šalje se QUIT signal. Ukoliko je proces i dalje živ, može se uništiti signalom KILL (kill -s 9 PID). Procesu shell se ne sme slati signal kill.



Slika 10.2 *Redosled primene signala za uništenje procesa*

Odjavljivanje sa sistema i procesi koji se izvršavaju u pozadini

Izvršavanje procesa u pozadini je posao u kome korisnik direktno ne učestvuje. Kao posledica toga može se desiti da korisnici zaborave na procese ili izgube predstavu o procesima koje su pokrenuli u pozadini. Ukoliko se korisnik odjavi sa sistema, UNIX će poslati signal HUP svim procesima koje je inicirao shell, odnosno uništiće sve procese koji su nastali kao posledica izvršenja korisničkih komandi. To znači da se nakon odjavljivanja korisnika prekida izvršenje svih programa koje je korisnik pokrenuo.

Neki programi se dugo izvršavaju tako da korisnici ne mogu čekati kraj njihovog izvršenja da bi se odjavili sa sistema. Ukoliko korisnik želi da pokrene program koji će nastaviti izvršenje i nakon odjavljivanja korisnika sa sistema potrebno je da obezbedi imunitet programa na HUP signal. Za to se koristi komanda nohup (no hang-up), čija je sintaksa:

```
$ nohup cmd args
```

gde su cmd i args komanda kojom se program pokreće i adekvatni argumenti. Program se dalje izvršava normalno, ali je imun na neke signale, uključujući HUP. Nakon odjavljivanja korisnika sa sistema, program nastavlja svoje izvršenje, ali više ne koristi terminal kao standardni izlaz.

Primer pokretanja nohup procesa je:

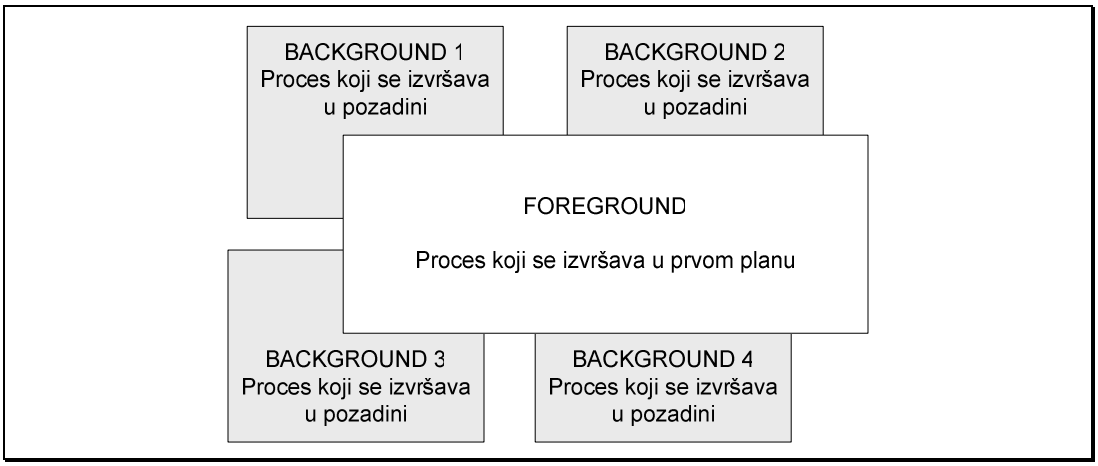
```
$ nohup sort lista1 > lista2 &
```

Nakon zadavanja ove komande korisnik može da se odjavi sa sistema, a proces koji je iniciran komandom sort će nastaviti svoje izvršenje. Nakon ponovnog prijavljivanja na sistem korisnik može da pogleda rezultat sortiranja u datoteci lista2.

Poslovi i prioriteti

Procesi koji se izvršavaju u pozadini i prioriteti. Grupe procesa i kontrola posla. Zakazivanje i periodično izvršavanje komandi.

UNIX je višeprocesni operativni sistem, odnosno ima mogućnost da izvršava veći broj zadataka istovremeno. Dok korisnik radi u tekst procesoru, u pozadini se izvršava veći broj procesa koji omogućavaju ispravno funkcionisanje sistema.



Slika 10.3 *Foreground i background procesi*

Procesi koji se izvršavaju u pozadini i prioriteti

Procesi koji se izvršavaju u prvom planu (foreground)

Komunikacija na relaciji komandni interpreter - korisnik najčešće se odvija po principu izvršavanja jednog zadatka u jednom momentu. Na primer, ako korisnik zada komandu sort bigfile, shell pokreće program i prikazuje izlaz na ekranu. Korisnik čeka da program završi s radom i da shell preuzme kontrolu, odnosno da se na ekranu pojavi komandni prompt, pre nego što zada drugu komandu. Izvršenje procesa u prvom planu (foreground processing) je podrazumevan način izvršenja procesa za sve komandne interpretere.

Procesi koji zahtevaju interakciju sa korisnikom moraju se izvršavati u prvom planu. Nemoguće je napisati pismo ili neki drugi dokument ako tekst procesor radi u pozadini i ako nema interakciju sa korisnikom. Procesi koji ne zahtevaju interakciju sa korisnikom (na primer, programi koji ulazne podatke čitaju iz datoteke ili ih dobijaju od drugih procesa, preko mreže ili na neki drugi način) mogu se izvršavati i u prvom planu i u pozadini. Takvi procesi mogu poruke korisniku upisivati u neku log datoteku, ili prikazivati periodično na ekranu, što zahteva periodično izvršenje u prvom planu.

Procesi koji se izvršavaju u pozadini (background)

Isključivo pokretanje procesa u prvom planu ograničava korisnika na serijsko izvršenje komandi. Takva metoda onemogućava korišćenje konzole ili Terminal prozora u periodu kada se izvršava bilo koji proces osim komandnog interpretera. Ako postoji potreba da se više procesa izvršava u isto vreme korisnik ih mora pokrenuti iz druge konzole ili Terminal prozora. Drugo rešenje je pokretanje procesa u pozadini.

Korisnik može pokrenuti proces u pozadini dodavanjem znaka & (ampersand) na kraj komandne linije. Procesi se u pozadini izvršavaju konkurentno sa svim sistemskim i korisničkim procesima. Procesi se izvršavaju u pozadini ukoliko su vremenski zahtevni, ukoliko troše veliku količinu procesorskog vremena ili ukoliko ne zahtevaju interakciju sa korisnikom. Kvazi paralelno ili paralelno sa procesima u pozadini izvršava se i proces koji je u prvom planu, a sa kojim korisnik može da ostvari interakciju. Sledeći primer ilustruje pokretanje procesa u pozadini.

```
$ sort verybigfile > sortedfile &
[1] 123
$ mail
No mail for nmacek
$ tar cf /home/nmacek /dev/rmt0
$ date
Sat Apr 17 20:05:14 CEST 2004
$ [1] + Done sort verybigfile > sortedfile &
```

Kada korisnik zada komandu "sort verybigfile > sortedfile &", proces koji uređuje datoteku verybigfile počinje da se izvršava u pozadini. Shell odmah vraća komandni prompt, tako da korisnik može da čita poštu ili izvrši neku drugu komandu koja zahteva interakciju dok se proces u pozadini izvršava. Pri tom, korisnik ne može da ostvari interakciju sa procesom koji je inicirala komanda sort, jer se on izvršava u pozadini. Kada proces u pozadini završi svoj rad shell na ekranu prikazuje poruku kojom obaveštava korisnika o tom događaju.

Prioriteti procesa

U administraciju procesa spada i kontrola potrošnje procesorskog vremena. Na UNIX sistemima postoji jednostavan mehanizam za određivanje relativnog značaja izvršenja jednog procesa u odnosu na drugi. Smanjivanjem potrošnje procesorskog vremena za velike poslove koji nisu vremenski kritični može se obezbediti pristojno funkcionisanje sistema.

UNIX, odnosno Linux kernel dodeljuje procesor procesima na osnovu prioriteta i vremena čekanja procesa na izvršenje. Procesu koji je spreman za izvršenje (proces koji ne čeka da se izvrši neki događaj ili ulazno-izlazna operacija, odnosno oslobađanje nekog resursa) dodeljuje se najniža numerička vrednost (najviši prioritet). Interaktivnim procesima dodeljuje se viši prioritet u odnosu na prioritet koji je dodeljen dugotrajnim procesima koji intenzivno koriste procesor. Time se obezbeđuje dobar odziv sistema na zahteve korisnika u svakom trenutku i relativno korektno izvršenje aplikacija koje nisu vremenski zahtevne ili zahtevne po pitanju potrošnje procesorskog vremena.

Napomena: niža numerička vrednost kojom je opisan prioritet znači viši prioritet procesa! Razlog za takvu konvenciju je sledeći: prilikom dodele procesora kernel najpre ispituje da li postoje procesi čiji je prioritet 0, i ukoliko postoje, dodeljuje procesor onom procesu koji je najviše vremena proveo u redu za čekanje (opisano stanjem READY u teoriji operativnih sistema). Ukoliko takvih procesa nema, procesor se dodeljuje procesima sa prioritetom 1, zatim sa prioritetom 2 i tako dalje.

Sledeća tabela opisuje način dodele procesora procesima koji rade u pozadini i dugo traju, a istog su prioriteta. Procesor se dodeljuje svakom procesu na određeno vreme (time-slice), koji se u tom intervalu izvršava. Zatim proces prelazi u stanje čekanja, a procesor se dodeljuje drugom procesu na korišćenje. Procesi su istog prioriteta, tako da se dodela procesora vrši po principu rotacije, pri čemu je time-slice jednak za sve procese. Proces koji se ne može izvršavati u momentu kada mu je procesor dodeljen (na primer čeka na oslobađanje resursa) premešta se na kraj procesorskog reda, a procesor se dodeljuje drugom procesu. Ukoliko korisnik pokrene neki interaktivni program (kao što je vi), imaće viši prioritet u odnosu na ove procese.

Proces 1	Proces 2	Proces 3
RUN	Wait	Wait
Wait	RUN	Wait
Wait	Wait	RUN
RUN	Wait	Wait
Wait	RUN	Wait
Wait	Wait	RUN

Tabela 10.1 *Dodela procesora procesima istog prioriteta*

Nice vrednost i prioriteti procesa

Jedan od faktora koji kernel uzima u obzir prilikom određivanja prioriteta procesa je nice vrednost koju kontrolišu korisnici, a koja se odnosi na pristojno ponašanje procesa u odnosu na ostale procese. Vrednosti nice se kreću u opsegu od 0 do 39, a podrazumevana vrednost je 20. Samo root može da smanji nice vrednost, tj. da povećava prioritet procesa, dok ostali korisnici mogu samo da je povećaju, i na taj način smanje prioritet procesa.

Sledeća tabela opisuje način dodele procesora procesima koji rade u pozadini i dugo traju, pri čemu je procesu 3 povećana nice vrednost. Time-slice je jednak za sve procese, ali se dodela procesora ne vrši po principu rotacije, jer je proces 3 sada nižeg prioriteta.

Proces 1	Proces 2	Proces 3 (Nice)
RUN	Wait	Wait
Wait	RUN	Wait
Wait	Wait	RUN

RUN	Wait	Wait
Wait	RUN	Wait
RUN	Wait	Wait
Wait	Wait	RUN
Wait	RUN	Wait
RUN	Wait	Wait
Wait	RUN	Wait

Tabela 10.2 *Dodela procesora procesima različitog prioriteta*

Pokretanje procesa sa sniženim prioritetom

Ukoliko želi da pokrene komandu u pozadini sa smanjenim prioritetom korisnik može da upotrebi komandu nice. Korisnik na taj način vodi računa o radu sistema, odnosno ne pokreće dugotrajne procese koji intenzivno koriste procesor sa visokim prioritetima, što je pristojno, odnosno lepo (nice). Time se obezbeđuje brže izvršavanje interaktivnih procesa koji rade u prvom planu. Reč nice se jednostavno dodaje ispred željene komande na sledeći način:

```
$ nice command &
```

Komanda nice zadata bez dodatnih argumentata (osim komande koju pokreće) smanjuje prioritet procesa koji ta komanda inicira za 10.

Promena prioriteta procesa komandom renice

BSD UNIX je uveo način za promenu nice vrednosti procesa. Komandom renice vlasnik procesa ili root mogu promeniti nice vrednost aktivnog procesa, čime se menja prioritet za dodelu procesora. Kao argumenti komande mogu se navesti PID procesa, grupe procesa (čime se menja nice vrednost svih procesa u grupi) ili korisničko ime (čime se menja nice vrednost svih procesa koje je inicirao dati korisnik). Sintaksa komande renice je sledeća:

```
$ renice priority [[-p] PID ...] [-g PGPID ...] [-u UID ...]
```

gde su argumenti:

priority nova nice vrednost procesa,

-p PID PID procesa kome treba promeniti nice vrednost,

-g PGPID PID procesne grupe kojoj treba promeniti nice vrednost.

-u UID ID korisnika čijim procesima treba promeniti nice vrednost

Napomena: ukoliko korisnik promeni nice vrednost svih svojih procesa, to će uključiti i promenu nice vrednosti komandnog interpretera, što znači da će svi procesi koji se pokrenu iz tog komandnog interpretera počev od tog trenutka imati niži prioritet.

U nastavku teksta dat je primer korišćenja komande renice, pod pretpostavkom da je pokrenut posao heavyduty, čiji je prioritet 20. Najpre je potrebno komandom ps odrediti PID procesa (na primer 1500), a zatim se zadaje komanda renice u sledećom obliku:

```
$ renice 30 1500
1500: old priority 20, new priority 30
```

Naknadno se pomoću komande ps može utvrditi da je prioritet procesa heavyduty smanjen, odnosno da ima veću nice vrednost (kolona NI).

Grupe procesa i kontrola poslova

Neki komandni interpreteri (kao što je Bourne Again Shell) uvode pojam kontrole poslova, koji je zasnovan na procesnim grupama. Svaki put kada korisnik zada komandu (ili više komandi spregnutih u pipeline) komandni interpreter kreira jednu grupu procesa. Grupu procesa čine oni procesi koji su nastali kao posledica izvršenja jedne komande. Ukoliko je komanda jednostavna, grupu obično čini jedan proces. Ukoliko se radi o komandama spregnutim u pipeline, grupu čini nekoliko procesa. Komandni interpreter procesnoj grupi dodeljuje identifikator koji je jednak identifikatoru jednog procesa iz grupe.

Posao (job) je grupa procesa koja se izvršava u pozadini. Poslovima se, kao i procesima, dodeljuju celobrojni numerički identifikatori (brojevi poslova - job number). Ukoliko korisnik pokrene neki posao komandni interpreter na ekranu prikazuje poruku koja uključuje broj posla. Grupa procesa je jako slična poslu - jedina bitna razlika je u tome što svako pokretanje komande rezultuje procesnom grupom, dok se broj posla dodeljuje samo ako se procesna grupa suspenduje ili smesti u pozadinu. Komandni interpreteri koji podržavaju koncept kontrole poslova korisnicima nude izvestan skup komandi koje služe za upravljanje poslovima. Dodatno, postojeće komande za upravljanje procesima (poput komande kill) mogu se upotrebiti i za upravljanje poslovima.

Kontrola poslova obuhvata sledeće operacije:

- pomeranje procesa iz pozadine u prvi plan i obrnuto,
- suspendovanje i nastavak izvršenja procesa.

Za svaki posao i procesnu grupu uvodi se pojam kontrolišućeg terminala (controlling terminal). Kontrolišući terminal je konzola (ili terminal prozor) iz koga je zadata komanda koja je inicirala procesnu grupu, odnosno posao. U svakom terminalu se samo jedan proces može izvršavati u prvom planu.

Komanda jobs

Komanda jobs prikazuje listu svih poslova koji su pokrenuti iz tekućeg komandnog interpretera, bez obzira da li se izvršavaju u pozadini ili su suspendovani.

```
$ jobs
[1] Stopped vi mydoc.txt
[2] - Running sort verybigfile > sortedfile &
```

```
[3] + Stopped (tty output) summararize_log &
```

Svaka linija u izlazu komande odgovara jednoj procesnoj grupi, pri čemu je celobrojna vrednost na početku linije broj posla. Broj posla se može koristiti kao argument komande kill, pri čemu se mora koristiti prefiks %. Na primer, korisnik može poslati signal procesu vi mydoc.txt pomoću identifikatora procesa i pomoću broja posla:

```
$ kill %1
```

Komandni interpreter takođe vodi evidenciju o trenutnim poslovima, koji su u izlazu komande jobs označeni znakom +, i prethodnim poslovima, koji su u izlazu komande jobs označeni znakom -. Ukoliko je korisnik pokrenuo više poslova, ostali neće imati posebne oznake. Koncept prethodnih i trenutnih poslova je samo pogodnost za korisnika - neke komande podrazumevano izvršavaju akcije nad trenutnim poslom ukoliko se broj posla (ili identifikator procesa) ne specificira kao argument.

Napomena: trenutni posao i procesna grupa koja se izvršava u prvom planu nisu ista stvar. Posao se izvršava isključivo u pozadini.

Premeštanje poslova u prvi plan (komanda fg)

U komandnim interpreterima koji ne podržavaju kontrolu poslova, proces koji je pokrenut u pozadini (pomoću sufiksa &) ostaje u pozadini dok se ne izvrši ili dok ne primi signal koji prekida rad procesa. Procesna grupa se može premestiti iz pozadine u prvi plan samo ako korisnik radi u komandnom interpreteru koji podržava kontrolu poslova. U tom slučaju, korisnik može pomoću komande fg premestiti izvršenje posla u prvi plan, nakon čega posao postaje procesna grupa.

Komandi fg se kao argument može navesti broj posla (sa prefiksom %) ili PID procesne grupe. Ako se argument ne navede, podrazumeva se trenutni posao. Rezultat izvršenja komande fg je premeštanje posla u prvi plan.

Suspendovanje procesne grupe

Procesna grupa se može suspendovati slanjem odgovarajućeg signala iz grupe signala za kontrolu poslova, što se može izvršiti na dva načina:

- kontrolnim karakterima za suspenziju procesa koji se izvršava u prvom planu,
- komandom kill.

Kontrolni karakter za suspenziju procesa (najčešće <Ctrl-Z>) šalje odgovarajući signal procesu koji se izvršava u prvom planu. Kombinacija tastera kojom se šalje ovaj signal se može redefinisati komandom stty int karakter. Aktuelna kombinacija može se videti u izlazu komande stty -a.

```
$ stty -a
...
eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
werase = ^W;
...
```

Sledeći primer ilustruje suspenziju procesa. Nakon pokretanja komande koja inicira izvršenje procesne grupe u prvom planu, potrebno je pritisnuti <Ctrl-Z> čime se procesna grupa suspenduje i postaje posao. Komandni interpreter na ekranu prikazuje poruku o suspenziji, zaključno sa brojem posla koji je dodeljen suspendovanoj procesnoj grupi. Nakon toga korisnik može komandama fg i bg nastaviti izvršenje procesa u prvom planu ili u pozadini.

```
$ sort verybigfile > sortedfile
<Ctrl-Z>
[1]+  Stopped                 sort verybigfile > sortedfile
```

Premeštanje poslova u pozadinu (komanda bg)

Komanda bg premešta suspendovani posao u pozadinu. Posao se najčešće premešta u pozadinu nakon suspenzije, odnosno slanja kontrolnih karaktera za suspendovanje procesne grupe koja se izvršava u prvom planu. Nakon premeštanja u pozadinu posao nastavlja sa izvršenjem u pozadini. Posao dalje ostaje u pozadini dok se ne izvrši, dok ne primi neki signal od korisnika, ili ne pokuša da izvrši ulazno/izlaznu operaciju vezanu za terminal.

```
$ bg %1
[1]+ sort verybigfile > sortedfile &
```

Komanda wait i čekanje izvršenja poslova

Komanda wait, koja je implementirana u većini komandnih interpretera, inicira čekanje izvršenja jednog ili svih procesa koji radi u pozadini. Najčešće se koristi u shell programima, ali se može koristiti i interaktivno, ukoliko je potrebno sačekati izvršenje određenog posla, odnosno izvršenje svih poslova ukoliko treba pokrenuti nove.

Ukoliko se komanda wait zada bez argumenata čeka se na sve poslove. Kao argumenti se mogu navesti PID procesne grupe ili broj posla, čime se inicira čekanje na određeni posao.

```
$ job1 &
[1] 20233
$ job2 &
 [2] 20234
$ job3 &
[3] 20235
$ job4 &
[4] 20237
$ wait %1
$ wait 20234
$ wait
[4] + Done job4 &
[3] + Done job3 &
$ jobs
$
```

Primer korišćenja kontrole poslova

Sledeći primer ilustruje sekvencu komandi kojima se izvršavaju sledeće aktivnosti: iniciranje posla, prikazivanje broja posla, smeštanje posla u prvi plan, suspendovanje procesne grupe koja se izvršava u prvom planu, smeštanje suspendovane procesne grupe u pozadinu i nastavak izvršenja, uništenje posla.

Napomene: broj u uglastim zagradama [x] je broj posla. Posao koji je obeležen znakom + je trenutni posao i njime se može upravljati komandama fg i bg bez eksplicitnog navođenja broja posla ili identifikatora procesa.

```
$ sleep 500&
[1] 6989
$ jobs
[1]+  Running                 sleep 500 &
$ fg %1
sleep 500
<Ctrl-Z>
[1]+  Stopped                 sleep 500
$ bg %1
[1]+ sleep 500 &
$ jobs
[1]+  Running                 sleep 500 &
$ kill %1
$ jobs
[1]+  Terminated              sleep 500
```

Zakazivanje i periodično izvršavanje komandi

Višeprocesni rad ozbiljnih operativnih sistema proširen je mogućnostima za pokretanje procesa u određenim trenucima vremena. UNIX i Linux sistemi sadrže programe pomoću kojih se može zakazati:

- izvršenje komande u određenom vremenskom trenutku
- periodično izvršenje komande

Komanda at

Komanda at omogućava korisnicima da zakažu izvršenje komandi, odnosno binarnih i shell programa u određenom trenutku vremena. Na primer, korisnik može da zakaže slanje e-mail poruke sa datotekom prijatelju za 25.maj u 7 sati, a administrator sistema pokretanje komande find koja će na svim diskovima tražiti određenu datoteku u 3h ujutru, kada je aktivnost korisnika relativno slaba.

Izvršenje komande u određenom vremenskom trenutku zakazuje se komandom at. Komanda zahteva navođenje vremena izvršenja u komandnoj liniji. Vreme se može navesti na nekoliko načina od kojih su neki prikazani u sledećem primeru (detaljna uputstva se nalaze u dokumentaciji komande - man page):

```
$ at 10:30am today
```

```
$ at midnight
$ at 12:01 4 July 2004
```

Nakon zadavanja komande pojavljuje se at prompt (at>), u koji se unose redom komande koje čine at posao (at job), odnosno komande koje u datom trenutku treba izvršiti. Nakon poslednje komande unosi se <Ctrl-D>, čime se označava kraj at posla (End Of Task - EOT) kao što je prikazano u sledećem primeru:

```
$ at midnight
warning: commands will be executed using /bin/sh
at> find / -name dummy -print > timmy
at> sort verybigfile > sortedfile
at> <Ctrl-D> <EOT>
job 1 at 2004-04-21 00:00
```

Dodatno, može se koristiti komanda atq (at query) koja prikazuje sve poslove koji su zakazani i atrm (at remove) kojom se zakazani posao briše iz liste.

```
$ atq
1       2004-04-21 00:00 a nmacek
$ atrm 1
$ atq
$
```

Shell program se može pokrenuti u određeno vreme na sledeći način:

```
$ at [time] [script filename]
```

Na primer, administrator sistema može napraviti shell program diskmedic koji detektuje neispravne blokove na površinama diskova, a zatim proverava integritet i defragmentuje sisteme datoteka. Sledeća komanda zakazuje izvršenje shell programa u ponoć.

```
$ at midnight discmedic
```

Periodično izvršavanje komandi

Pomoću alata crontab korisnici mogu zakazati periodično izvršenje komande, odnosno izvršenje u specificiranim intervalima. Alat crontab se može iskoristiti za zakazivanje periodičnog izvršenja rutinskih poslova, kao što su backup, pronalaženje i brisanje core datoteka u home direktorijumima korisnika, pa čak i slanje rođendanskih čestitki prijateljima.

Komanda crontab koristi se za pregled i izmenu crontab datoteke u kojoj se čuvaju informacije o zakazanim poslovima i rasporedu njihovog izvršavanja. Na nekim UNIX sistemima, kao što je Solaris, korisnici mogu da vrše izmene u crontab datoteci, koja je inicijalno prazna. Na drugim sistemima korisnicima nije dato pravo za izmenu crontab datoteke iz sigurnosnih razloga, tako da administrator mora najpre da im dodeli odgovarajuće dozvole.

Datoteka crontab organizovana je u vidu jednostavne tabele. Svaka linija u crontab datoteci sastoji se od šest polja razdvojenih razmaknicom ili tabulatorom, i predstavlja

jednu komandu za koju je zakazano periodično izvršenje. Sledeći primer ilustruje mogući zapis u datoteci:

```
0 17 * * 5 banner "Goodbye, cruel world !" > /dev/console
```

Ukoliko se ovakav zapis unese u crontab datoteku, poruka "Goodbye, cruel world !" će biti prikazana na ekranu svakog petka u 17:00h.

Polja su organizovana sledećim redom: m h dom mon dow command, gde je:

m	minut
h	čas
dom	dan u mesecu (Day Of Month)
mon	mesec (Month)
dow	dan u nedelji (Day Of Week)

Ova polja mogu sadržati:

- jednu numeričku vrednost
- više numeričkih vrednosti razdvojenih zarezima (1,3,5 u polju dow znači ponedeljak, sreda i petak)
- opseg numeričkih vrednosti (1-5 u polju dow znači od ponedeljka do petka)
- znak "*", koji zamenjuje sve dozvoljene vrednosti

Izmena crontab datoteke vrši se komandom crontab -e (edit) command. Sva polja se moraju popuniti da bi zapis bio regularno prihvaćen. Linux koristi vi (ili vim) kao podrazumevani editor za izmenu crontab datoteke. Sadržaj crontab datoteke može se izlistati komandom crontab -l, a datoteka se može obrisati komandom crontab -r.

cron daemon

Cron daemon je program koji pokreće zakazane komande (Vixie Cron). Cron se pokreće prilikom prelaska u višekorisničke nivoe izvršenja, odnosno prilikom podizanja sistema (ukoliko je sistem konfigurisan da se podigne u višekorisničkom režimu). Cron proverava korisničke crontab datoteke svakog minuta i pokreće programe koje tada treba izvršiti. Korisničke crontab datoteke se nalaze u spool direktorijumu (/var/spool/cron/crontabs) a imenovane su na osnovu korisničkih naloga.

Napomena: crontab datoteke ne treba menjati direktno, editorom, već isključivo komandom crontab.

Alternativni programski paket koji se može koristiti na Linux sistemima je anacron.

11

PODIZANJE I ZAUSTAVLJANJE SISTEMA

Prva stvar koju je potrebno uraditi da bi se korisnicima omogućio pristup UNIX sistemu je podizanje sistema (booting, bootstrap, ili jednostavno - boot). Prilikom podizanja sistema vrši se inicijalizacija hardverskih uređaja i eventualna rekonfiguracija kernela, nakon čega sistem aktivira sve auto-mount sisteme datoteka i veliki broj procesa koji omogućavaju višekorisnički rad, grafičko okruženje i mrežno okruženje. Skup procesa, odnosno servisa koje je potrebno aktivirati određen je stanjem u koje se sistem dovodi, odnosno nivoom izvršenja (runlevel). Prilikom zaustavljanja, odnosno obaranja sistema (shutdown) sistem šalje signale TERM i KILL kojima postupno završava i ubija procese, zatim deaktivira sve sisteme datoteka, po potrebi radi memory-dump i na kraju isključuje sistem.

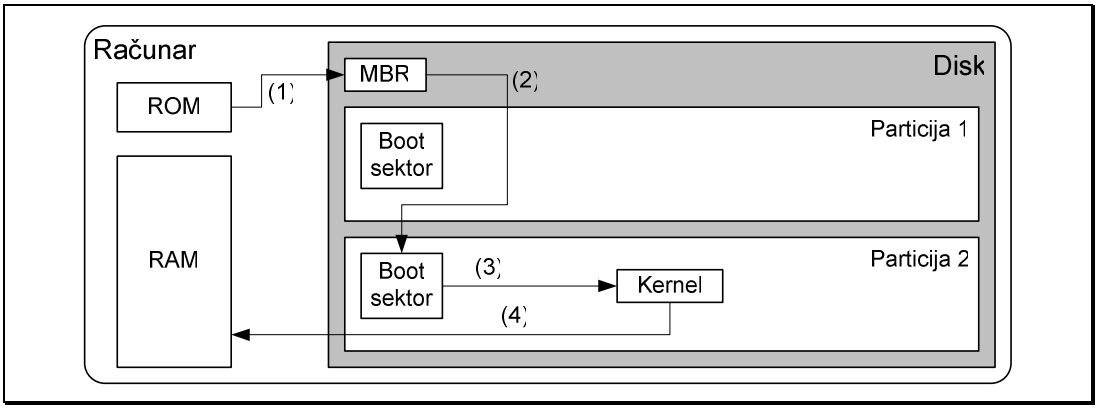
Podizanje sistema (boot)

Boot procedura. Komanda init i nivoi izvršenja. Inicijalizacione rc datoteke.

Kada se računar uključi BIOS izvršava POST rutinu (Power On Self Test), odnosno pokreće seriju testova hardvera nakon čega kreće proces podizanja sistema. Podizanje sistema (boot) je procedura koja se izvršava u cilju dovođenja sistema u operativno stanje.

Sistem se podiže sa masovnog memorijskog medijuma (hard disk, disketa), sa kog se najpre pročita prvi sektor (boot sector na disketama, master boot record na hard diskovima). U boot sektoru se nalazi mali program koji je zadužen da pokrene punjenje RAM memorije operativnim sistemom. Ukoliko se sistem podiže sa diskete bez sistema datoteka, boot sektor može sadržati jednostavan kôd koji memoriju puni kernelom, sekvencijalno smeštenim na blokove diskete. Moguće je sistem podići i sa diskete na kojoj je kreiran sistem datoteka, a u tom slučaju se koristi LILO (Linux Loader). Prilikom podizanja sistema sa hard diska, kôd upisan u master boot record najpre identifikuje aktivnu particiju u particionoj tabeli, a

zatim izvršava kôd upisan u boot sektoru aktivne particije, koji dalje puni memoriju kernelom.



Slika 11.1 *Bootstrap rutina*

Ukoliko administrator sistema želi da pri podizanju sistema izabere jedan od više mogućih kernela potrebno je koristiti boot manager, kao što su LILO i GRUB (Grand Unified Bootloader). Dodatno, boot manager se mora koristiti ukoliko je na računaru instalirano više operativnih sistema od kojih se bira jedan prilikom podizanja sistema. LILO se može konfigurisati da odmah napuni memoriju podrazumevanim operativnim sistemom, odnosno podrazumevanim kernelom Linux sistema. U tom slučaju se izbor alternativnog operativnog sistema, odnosno kernela, može izvršiti pritiskom na tastere alt, shift ili ctrl pri podizanju sistema. LILO se takođe može konfigurisati tako da pre punjenja memorije čeka da korisnik odabere kernel, odnosno operativni sistem koji želi da podigne. Dodatno, korišćenjem LILO boot managera kernelu se mogu proslediti razni parametri (navode se nakon imena kernela).

U svaku Linux distribuciju je uključen najmanje jedan boot manager koji se instalira i konfiguriše prilikom instalacije Linux sistema. Boot manager se naknadno može rekonfigurisati (komandama liloconfig ili grub) ili instalirati. Takođe, mogu se koristiti alternativni boot manageri raznih proizvođača, kao što je Power Quest.

Sistem administrator može konfigurisati LILO na dva načina:

- pomoću interaktivnog programa /usr/sbin/liloconfig, koji zahteva da se odgovori na nekoliko jednostavnih pitanja. Pri tom se korisniku najpre nudi da instalira podrazumevanu LILO konfiguraciju (opisana u datoteci /etc/lilo.conf);

```
# /usr/sbin/liloconfig
LILO, the LInux LOader, sets up your system to boot Linux directly
from your hard disk, without the need for a boot floppy.

You already have a LILO configuration in the file /etc/lilo.conf

Checking your /etc/lilo.conf for incompatible options...

Install a boot block using your current LILO configuration? [Yes]
```

- modifikacijom konfiguracione datoteke /etc/lilo.conf i pokretanjem programa lilo, koji ažurira informacije boot sektora sa informacijama u datoteci. Datoteka lilo.conf ima sledeću strukturu: svaka stavka u datoteci opisuje jedan dostupan operativni sistem, odnosno particiju na kojoj se on nalazi. Ukoliko se stavka odnosi na Linux, potrebno je navesti putanju i ime slike kernela. U nastavku teksta dat je isečak datoteke /etc/lilo.conf.

```
# Support LBA for large hard disks
lba32
# Specifies the boot device
boot=/dev/sda
# Specifies the device that should be mounted as root
root=/dev/sdb3
# Specifies the boot delay in deciseconds
delay=20
# Specifies the VGA text mode at boot time.
vga=normal
# Boot up Linux by default
default=Linux
image=/vmlinuz
        label=Linux
        read-only
image=/vmlinuz.old
        label=LinuxOLD
        read-only
# Specify another OS on this machine.
other=/dev/hda1
        label=WindowsXP
```

Prilikom podizanja sistema LILO zahteva od korisnika da izabere operativni sistem. Ukoliko se izabere Linux u memoriju se najpre učita slika kernela.

Prilikom punjenja memorije kernelom na sistemu se izvršavaju sledeće aktivnosti:

- dekompresija kernela. Ukoliko je kernel komprimovan prilikom učitavanja se vrši dekompresija. Komprimovani kernel zauzima manje mesta na disku, ali se zbog dekompresije sporije učitava. Dekompresiju vrši mali program (gzip ili bzip2) koji se nalazi na početku slike kernela;
- detekcija video adaptera i promena tekstualnog režima (na primer 100x40 karaktera) ukoliko je to specificirano prilikom prevođenja kernela. Tekstualni režim se dodatno može specificirati pomoću LILO boot managera ili pomoću komande rdev (rdev služi za ispitivanje i promenu parametara u slici kernela);
- detekcija hardvera i konfiguracija odgovarajućih drajvera za uređaje;
- aktiviranje root sistema datoteka u režimu čitanja. Particija na kojoj se nalazi root sistem datoteka može se specificirati pomoću komande rdev ili LILO boot manager programa. Tip root sistema datoteka automatski se detektuje. Ukoliko se root sistem datoteka iz nekih razloga ne aktivira (na primer, u kernel nisu uključeni drajveri za taj tip sistema datoteka) kernel prelazi u stanje panike (kernel panic) i zaustavlja sistem.

Nakon toga, sistem se nalazi u jednokorisničkom režimu rada, što znači da samo jedan korisnik može da koristi sistem. U ovom režimu rada aktiviran je jedino root sistem datoteka u režimu čitanja (ostali sistemi datoteka se aktiviraju nakon prelaska u višekorisnički režim rada) i shell koji prihvata isključivo komande superusera.

Kada obavi svoje aktivnosti prilikom podizanja operativnog sistema, tj. kada se učita u memoriju, pokrene i inicijalizuje sve neophodne drajvere i strukture podataka, kernel pokreće program init, i novokreiranom procesu dodeljuje PID=1, čime završava svoj deo boot procedure. Program init svoje aktivnosti obavlja u skladu sa konfiguracionim datotekama, nakon čega sistem prelazi u višekorisnički režim rada i može se normalno koristiti.

init

Init je prvi proces korisničkog nivoa kog pokreće kernel, a zbog svog značaja za pravilno funkcionisanje sistema u literaturi se pominje pod imenom "super daemon". Init je zadužen za:

- pokretanje procesa getty (tako da korisnici mogu da se prijave na sistem) i upravljanje terminalima. Init obezbeđuje ispravno funkcionisanje programa getty i prekida njegov rad ukoliko administrator zabrani prijavljivanje na sistem. Na sistemima sa grafičkim okruženjem, init obezbeđuje pravilno funkcionisanje grafički orijentisanog login programa;
- implementaciju nivoa izvršenja (runlevels);
- usvajanje procesa siročića (orphans).

Izvršna datoteka, kojom se pokreće proces init, na Linux sistemima se po pravilu nalazi u direktorijumu sbin. Ukoliko je iz nekih razloga tamo nema, kernel je traži na lokacijama u kojima je drugi UNIX sistemi smeštaju. Ukoliko ne može da pronađe datoteku init kernel pokreće Bourne Shell (/bin/sh). Ukoliko ni u tome ne uspe, podizanje sistema se završava neuspehom.

Kada se pokrene init izvršava razne administrativne zadatke kojima završava proceduru podizanja sistema:

- pokreće program fsck koji proverava integritet sistema datoteka,
- po potrebi čisti direktorijum /tmp,
- pokreće razne servise,
- pokreće po jednu instancu programa getty za svaki terminal i virtuelnu konzolu s koje korisnici mogu da se prijave na sistem.

Kada se korisnik odjavi sa sistema init ponovo pokreće getty za svaki terminal čime obezbeđuje mogućnost da se sledeći korisnik prijavi na sistem.

Nivoi izvršenja (runlevels)

Nivo izvršenja je generalizovan režim rada Linux sistema, odnosno stanje init procesa i sistema u celini koje definiše skup servisa koji se u tom nivou izvršavaju. Linux dozvoljava najviše 10 nivoa izvršenja. Nivoi izvršenja se označavaju brojevima:

0	gašenje sistema (system halt)
1	jednokorisnički režim
6	ponovno podizanje sistema (system reboot)
2-5	korisnički definisani nivoi izvršenja

Nivoi se definišu u datoteci /etc/inittab i mogu biti različiti za različite Linux distribucije. Sistem administratori putem nivoa izvršenja definišu aktivnosti podsistema, kao što su X i razni mrežni servisi.

Konfiguraciona datoteka /etc/inittab i init-getty relacija

Kada se pokrene, init čita /etc/inittab konfiguracionu datoteku. U toku rada, init će ponovo pročitati ovu datoteku ukoliko mu korisnik pošalje HUP signal, čime se eliminiše procedura zaustavljanja i podizanja sistema prilikom izmene init konfiguracije.

Format datoteke /etc/inittab je relativno komplikovan. Sve linije datoteke sadrže četiri polja razdvojena dvotačkom:

```
id:runlevels:action:process
```

Značenje polja je sledeće:

id	jedinstvena oznaka do četiri karaktera;
runlevels	nivoi izvršenja u kojima se komanda izvršava (nivoi se navode kao jednocifreni brojevi i ne razdvajaju se drugim karakterima);
action	akcija koju init treba da izvrši nad datim procesom;
process	proces koji init startuje prilikom prelaska sistema u navedeno stanje.

Proces init nad procesima može da izvrši sledeće akcije:

boot	init pokreće proces i bez čekanja prelazi na sledeći. Akcija se može izvršiti samo prilikom prelaska u jednokorisnički režim rada (nivo izvršenja 1);
bootwait	init pokreće proces, a na sledeći prelazi tek nakon završetka. I ova akcija se može izvršiti samo prilikom prelaska u jednokorisnički režim rada (nivo izvršenja 1);
off	init ubija proces;
once	init pokreće proces i čeka na izvršenje;
respawn	init pokreće proces i ne čeka na izvršenje. Ako se proces uništi, init ga ponovo pokreće;
wait	init pokreće proces i čeka na izvršenje.

Prilikom konfigurisanja init - getty relacije u polje id se upisuje oznaka terminala na kom se getty pokreće (odnosno karakter koji stoji iza /dev/tty u nodu terminala). Na primer, za pokretanje getty procesa na prvom virtuelnom terminalu (/dev/tty1), u svim normalnim višekorisničkim nivoima izvršenja (2,3,4 i 5), pri čemu se getty proces pokreće ponovo nakon završetka rada, u datoteku /etc/inittab treba upisati sledeću liniju:

```
1:2345:respawn:/sbin/getty 9600 tty1
```

Administrator u datoteku mora upisati po jednu liniju za svaki terminal na kom želi da pokrene getty proces.

Komanda init

Komandom init vrši se promena nivoa izvršenja u aktivnom sistemu.

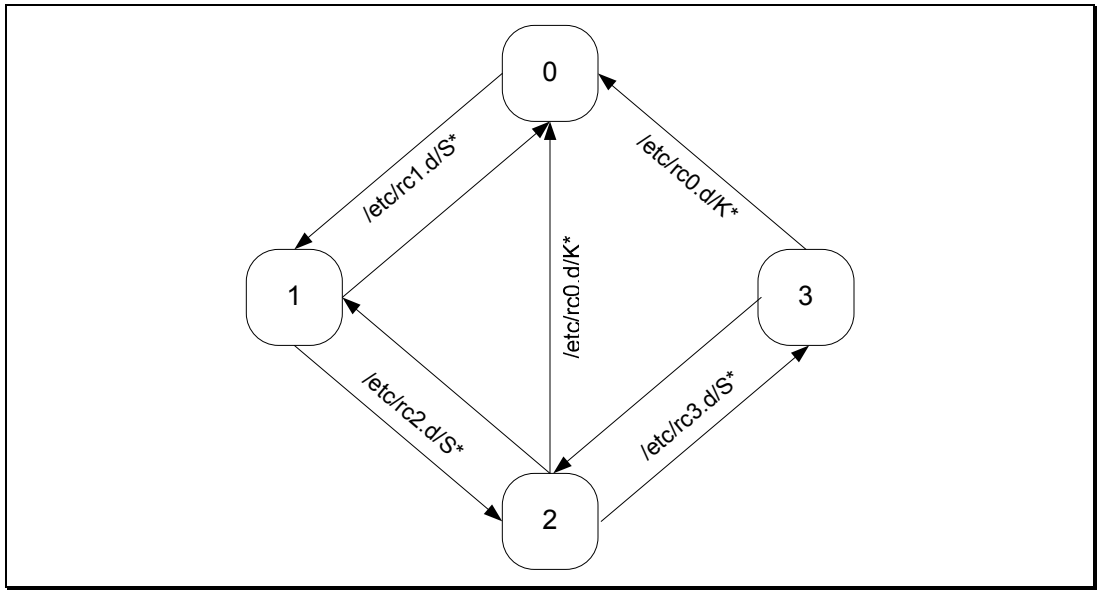
```
# init 0            # isključivanje sistema
# init 1            # jednokorisnički režim rada (init s)
# init [2-5]        # korisnički definisani nivo izvršenja
# init 6            # zaustavljanje sistema
```

Dodatno, komandom init se može saopštiti procesu init da ponovo pročita sadržaj datoteke /etc/inittab na sledeći način:

```
# init q
```

Inicijalizacione rc datoteke

Prilikom prelaska u novi nivo izvršenja sistem izvršava komande, odnosno shell programe, koji se nalaze u /etc/rc?.d direktorijumima. Ove datoteke su označene zajedničkim imenom inicijalizacione rc datoteke, ili skraćeno - rc datoteke. Na slici 11.2. prikazani su nivoi izvršenja i odgovarajuće rc datoteke koji se pri prelasku izvršavaju.



Slika 11.2 *Nivoi izvršenja i rc datoteke*

Sve rc datoteke, odnosno shell programi nalaze se u direktorijumu /etc/init.d. U /etc/rc?.d direktorijumima smešteni su linkovi na ove datoteke:

/etc/rc0.d — shell programi koji se izvršavaju prilikom zaustavljanja sistema, odnosno prilikom prelaska u nivo 0 (shutdown) ili 6 (reboot). Ovi skriptovi šalju signale procesima, ubijaju većinu servisa (cron, samba, apache, inetd, lpd) i deaktiviraju sisteme datoteka;

/etc/rc1.d — shell programi koji se pokreću prilikom prelaska u jednokorisnički režim rada (nivo 1);

/etc/rc2.d — shell programi koji se pokreću prilikom prelaska u nivo 2, odnosno višekorisnički režim rada (multiuser);

/etc/rcS.d — shell programi koji se pokreću prilikom podizanja sistema, čak i ako se sistem podiže u jednokorisničkom režimu rada.

Datoteke u rc direktorijumima imenuju se na osnovu sledeće konvencije:

/etc/rcX.d — direktorijum koji sadrži linkove na shell programe koji se izvršavaju prilikom prelaska u nivo izvršenja X;

Sxxime — skriptovi za pokretanje programa;

Kxxime — skriptovi za ubijanje programa.

Shell programi se izvršavaju po abecednom redu. Na primer, ukoliko je sadržaj direktorijuma /etc/rc2.d:

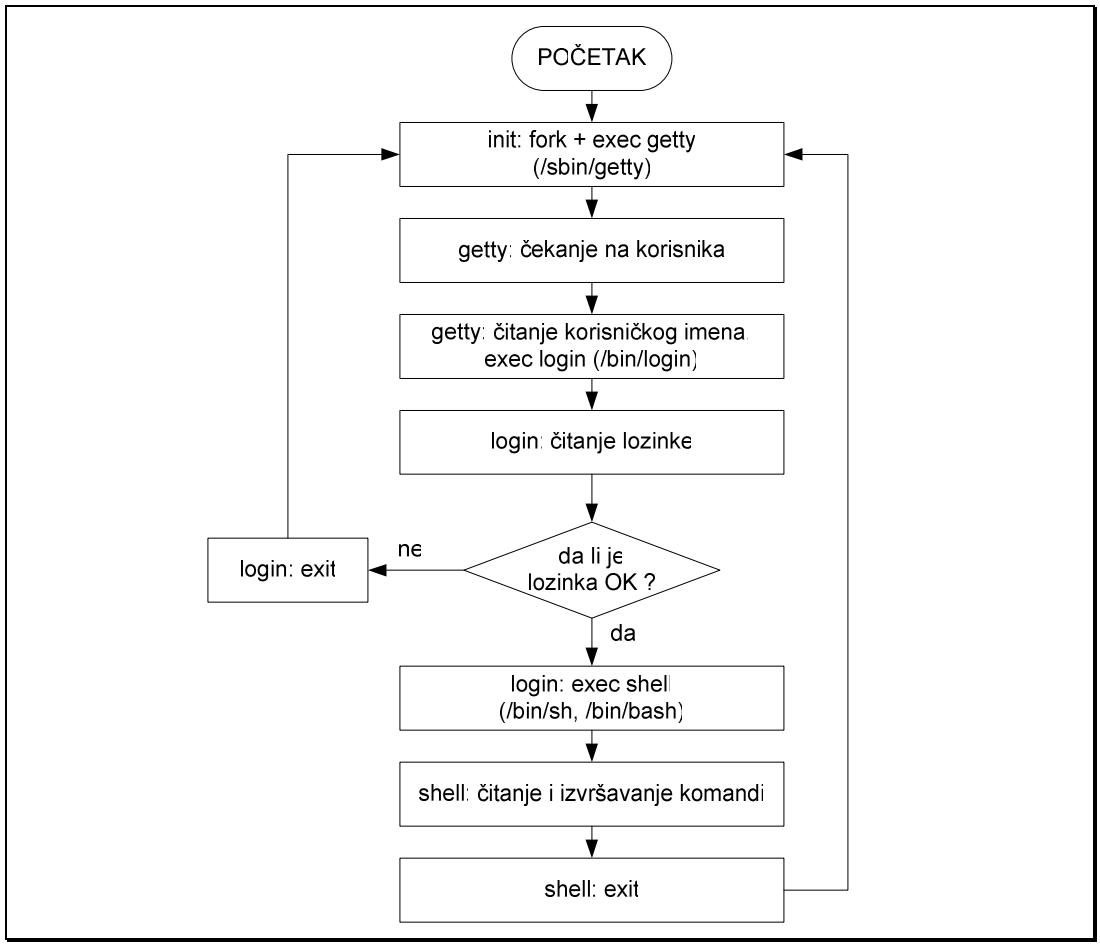
```
# ls /etc/rc2.d
S10sysklogd   S14ppp        S20exim       S20lpd        S20mysql
S20postfix    S20samba      S89atd        S91apache     S11klogd
S19nfs-common S20inetd      S20makedev    S20nfs-kernel-server
S20postgresql S20ssh        S89cron       S99rmnologin
```

prilikom prelaska u nivo 2 izvršavaju se redom skriptovi: S10sysklogd, s14ppp, s20exim, i tako redom.

Prijavljivanje na sistem

Slika 10.4 ilustruje proces prijavljivanja na sistem preko terminala.

Najpre init pokreće po jednu instancu procesa getty za svaki terminal sa kog je korisnicima dozvoljeno prijavljivanje na sistem. getty na ekranu štampa poruku koja je upisana u datoteku /etc/issue, a zatim osluškuje terminal i čeka da korisnik unese korisničko ime. getty zatim pokreće login proces kome kao parametar prenosi korisničko ime. login zahteva od korisnika da unese lozinku, a zatim proverava par korisničko ime-lozinka. Ukoliko je lozinka validna login pokreće komandni interpreter koji je za datog korisnika naveden u /etc/passwd, a u suprotnom prekida izvršenje, nakon čega init pokreće novu instancu getty procesa. Kada korisnik završi rad i odjavi se sa sistema komandni interpreter prekida izvršenje i vraća kontrolu procesu init.



Slika 11.3 *Prijavljivanje na sistem preko terminala*

Napomena: init za svaki terminal sa kog je dozvoljeno prijavljivanje na sistem kreira samo jedan proces pomoću sistemskog poziva fork i u njegovom kontekstu izvršava program getty. getty, login i shell se izvršavaju u kontekstu istog procesa.

Prijavljivanje na sistem preko mreže

Prijavljivanje na sistem preko mreže razlikuje se od prijavljivanja na sistem sa terminala. Virtuelne veze se uspostavljaju svaki put kada dva programa na dva različita umrežena računara žele da komuniciraju. Korisnici se praktično mogu prijaviti na sistem sa svakog računara u mreži, tako da je pokretanje posebne instance programa getty za svaki računar, odnosno svaki potencijalni login, nepraktično. Umesto toga koristi se program inetd (odnosno xinted) koji se ponaša kao obvojnica (wrapper). Proces inetd osluškuje zahteve za uspostavljanjem konekcije i za svaki zahtev pokreće adekvatan program kom predaje kontrolu nad tim zahtevom. U slučaju da se pojavi zahtev za prijavljivanje na sistem preko mreže inetd pokreće odgovarajuću telnet ili rlogin sesiju i nastavlja dalje da osluškuje portove.

Funkcija procesa login

login je proces koji se bavi autentifikacijom korisnika, odnosno proverom ispravnosti korisničkog imena i lozinke, nakon čega štampa poruku iz datoteke /etc/motd (Message Of The Day) i pokreće komandni interpreter.

Ukoliko datoteka /etc/nologin postoji prijavljivanje na sistem je zabranjeno. Datoteku obično kreira shutdown proces. Proces login proverava postojanje ove datoteke i odbija sve zahteve za prijavljivanje na sistem ukoliko ona postoji, a korisnicima koji pokušaju da se prijave na sistem na ekranu prikazuje njen sadržaj.

Korišćenjem syslog procesa login beleži sve neuspešne pokušaje prijavljivanja na sistem kao i svako prijavljivanje superusera. Ovi događaji mogu se iskoristi za praćenje uljeza.

Trenutno prijavljeni korisnici sistema su upisani u datoteku /var/run/utmp, koja je validna do sledećeg zaustavljanja sistema. Komanda who prilikom prikazivanja trenutno prijavljenih korisnika čita sadržaj datoteke /var/run/utmp.

Svako uspešno prijavljivanje na sistem upisuje se u datoteku /vat/log/wtmp. Ova datoteka nema ograničenje veličine, tako da je potrebno redovno čistiti sadržaj ove datoteke ukoliko se korisnici često prijavljuju na sistem. Komanda last čita sadržaj datoteke wtmp.

Zaustavljanje sistema

Procedura zaustavljanja sistema. Komanda shutdown. Jednokorisnički režim rada.

Linux sistem u memoriji čuva najvažnije delove sistema datoteka, kao što su superblock i liste slobodnih blokova i i-nodeova. Tokom normalnog rada sistem u određenim trenucima usklađuje sadržaj sistema datoteka sa keš memorijom, odnosno vrši upis keširanih podataka iz memorije u sisteme datoteka. Ukoliko se sistem nasilno isključi može doći do

oštećenja sistema datoteka kao posledica neizvršene sinhronizacije. Dodatno, ukoliko sistem koristi više korisnika, koji su na sistem prijavljeni sa različitih terminala, potrebno ih je blagovremeno upozoriti da svoj rad snime na disk. Zato se sistem zaustavlja pomoću komande shutdown, ili pomoću odgovarajuće procedure za zaustavljanje sistema iz grafičkog okruženja.

Prilikom zaustavljanja Linux sistema izvršavaju se sledeće aktivnosti:

- svim prijavljenim korisnicima šalje se upozorenje,
- ubijaju se svi korisnički procesi i svi servisi, osim procesa shutdown,
- vrši se sinhronizacija sistema datoteka sa keš memorijom,
- deaktiviraju se svi aktivni sistemi datoteka,
- po potrebi se radi memory dump, odnosno kreira se slika memorije na disku,
- sistem se isključuje ili ponovo podiže, zavisno od zahteva. Ukoliko računar nema mogućnost softverskog isključenja napajanja, na ekranu se prikazuje poruka "System halted." nakon koje administrator može da isključi napajanje.

Komanda shutdown

Komandom shutdown se pokreće ispravna procedura zaustavljanja sistema. Sintaksa komande shutdown je:

```
$ /sbin/shutdown [-t sec] [-akrhfFc] time [message]
```

Nakon pokretanja komande svim prijavljenim korisnicima se šalje upozorenje, a dalje prijavljivanje na sistem se zabranjuje. Sistem se može oboriti odmah, ili nakon specificiranog vremenskog intervala. Slanjem signala TERM svi procesi se upozoravaju da će sistem biti oboren, čime im se daje mogućnost da regularno završe svoje aktivnosti. Nakon toga, shutdown šalje signal procesu init i zahteva promenu nivoa izvršenja (nivo 0 je gašenje sistema, a nivo 6 reboot). Dalje se izvršavaju akcije predviđene rc inicijalizacionim datotekama i datotekom /etc/inittab.

Argumenti komande shutdown su:

-t sec — uvodi se čekanje od sec sekundi između slanja TERM signala procesima i prelaska u drugi nivo izvršenja, odnosno slanja KILL signala;

-k — sistem se ne gasi, već se svima šalje poruka upozorenja;

-r — podizanje sistema nakon zaustavljanja (reboot);

-h — halt - nakon zaustavljanja sistema, računar se gasi;

-f — prilikom sledećeg podizanja sistema fsck se ne pokeće, odnosno ne proverava se integritet sistema datoteka. Ukoliko se navede fleg -f (reboot fast) shutdown će kreirati datoteku /fastboot koja se uklanja prilikom sledećeg podizanja sistema;

-F prilikom sledećeg podizanja sistema fsck se obavezno pokreće (force fsck) i proveravaju se svi sistemi datoteka, bez obzira na to da li su prilikom zaustavljanja sistema pravilno deaktivirani ili ne. Ukoliko se navede fleg -F shutdown će kreirati datoteku /forcefsck, koja se uklanja prilikom sledećeg podizanja sistema;

-c prekida se zakazano zaustavljanje sistema. Korisnicima se može poslati nova poruka;

time vreme kada sistem treba oboriti. Ovaj argument je obavezan! Ukoliko se shutdown pokrene sa odlaganjem, kreira se datoteka /etc/nologin koja sprečava prijavljivanje na sistem. Shutdown uklanja ovu datoteku pre promene nivoa izvršenja ili ako je zahtev za zaustavljanjem sistema prekinut (fleg -c) pre slanja signala procesu init. Vreme se može navesti u apsolutnom ili relativnom formatu. Apsolutnim formatom hh:mm inicira se zaustavljanje sistema u hh časova i mm minuta (hh je jednocifren ili dvocifren, a mm isključivo dvocifren broj). Relativnim formatom vremena +m specificira se odlaganje - na zaustavljanje sistema čeka se m minuta. Ukoliko se navede NOW sistem se zaustavlja odmah (ekvivalent +0);

message poruka upozorenja koja se šalje svim korisnicima.

Kontrola pristupa rutinama za zaustavljanje sistema

Shutdown se može pozvati kombinacijom tastera <Ctrl+Alt+Del>. Akcija, odnosno komanda koju ova kombinacija tastera pokreće definiše se u datoteci /etc/inittab:

```
# What to do when CTRL-ALT-DEL is pressed.
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```

Ako korisnik na konzoli prititsne <Ctrl+Alt+Del>, izvršava se reboot procedura, odnosno komanda shutdown -t1 -a -r now. Parametar -a uvodi kontrolu pristupa rutinama za gašenje sistema. Shutdown najpre proverava da li datoteka /etc/shutdown.allow postoji, a ukoliko postoji proverava ko je sve prijavljen na sistem. Dalje se upoređuju imena korisnika prijavljenih na virtuelne konzole (odnosno korisnika koji rade za tim računarom) sa imenima koja su navedena u datoteci /etc/shutdown.allow. Ukoliko na konzoli radi root ili neki od korisnika navedenih u /etc/shutdown.allow, sistem se zaustavlja i podiže ponovo, a u suprotnom se na ekranu ispisuje poruka "no authorized users logged in to the (physical) system console". Format datoteke /etc/shutdown.allow je jedno korisničko ime po liniji. Prazne linije i komentari (linije koje počinju znakom #) su dozvoljene. Najveći broj korisnika koji se smeju navesti u datoteci /etc/shutdown.allow je 32. Ukoliko datoteka /etc/shutdown.allow ne postoji parametar -a se ignoriše.

12

INSTALACIJA SOFTVERSKIH PAKETA

Prilikom instalacije Linux operativnog sistema instalacioni program kreira određeni broj sistema datoteka, formira aktivno UNIX stablo, a zatim započinje kopiranje softverskih paketa sa instalacionih medijuma. Neki paketi koji se isporučuju uz standardne Linux distribucije predstavljaju konačne verzije paketa, dok se drugi nalaze u fazi testiranja i kao takvi nisu najpouzdaniji. Takođe, neki paketi se ne isporučuju uz sve distribucije iako su besplatni. Nakon pojavljivanja konačne (ili nove) verzije, paket se može preuzeti sa Interneta (ukoliko je besplatan) i instalirati na sistem. U ovom poglavlju objašnjeni su osnovni postupci instalacije softverskih paketa, odnosno korišćenje paket menadžera.

Standardni formati paketa

Razlozi za korišćenje paket menadžera. Osobine najčešće korišćenih vrsta softverskih paketa.

Bilo kakva automatska obrada informacija zahteva najmanje tri stvari: hardver koji će procesirati informacije (računar), program za obradu informacija i same informacije. Da bi program mogao da se izvršava na računaru i obradi informacije na onaj način na koji korisnici to očekuju, potrebno je programu obezbediti odgovarajuće "uslove za rad". Osim prostora na disku većina programa zahteva sledeće:

- informacije koje će procesirati, smeštene na tačno određenom mestu na sekundarnim memorijama (diskovi, trake), pod odgovarajućim imenom i u odgovarajućem formatu;
- jednu ili više konfiguracionih datoteka, pomoću kojih se kontroliše ponašanje programa, i donekle omogućava prilagođavanje programa korisniku. Konfiguracione datoteke se takođe moraju smestiti na diskove pod odgovarajućim imenima na odgovarajuća mesta;

- pravilno određena mesta na diskovima gde će program snimati rezultate i međurezultate obrade ulaznih informacija.

Dodatno, neki programi mogu zahtevati da na sistem budu instalirani i drugi softverski paketi, bez kojih upotreba samih programa nije moguća. Iako to nije obavezno, uz program se u najvećem broju slučajeva isporučuje i prateća dokumentacija koja omogućava korisnicima da lakše i brže nauče da koriste program.

To znači da proces instalacije programskog paketa obuvata veći broj postupaka od jednostavnog kopiranja datoteka. U opštem slučaju instalacioni proces obuhvata sledeće aktivnosti:

- proveru međuzavisnosti paketa (package dependencies), odnosno proveru postojanja prethodno instaliranih paketa koji su neophodni za izvršavanje konkretnog programa (na primer, bashburn zahteva da na sistemu bude instaliran paket cdrecord, dok Midnight Commander zahteva da na sistemu bude instaliran paket ncurses). Ti paketi se moraju instalirati pre instalacije samog programa;
- proveru slobodnog mesta u sistemima datoteka. Ukoliko na diskovima nema dovoljno prostora za smeštaj paketa, proces instalacije se prekida;
- kreiranje privremenih direktorijuma za kopiranje i raspakivanje paketa;
- kreiranje destinacionih direktorijuma za smeštaj izvršnih i konfiguracionih datoteka i dokumentacije;
- prevođenje i povezivanje izvornog koda programa;
- kreiranje ili kopiranje inicijalnih konfiguracionih datoteka;
- kopiranje dokumentacije;
- kreiranje linkova;
- brisanje privremenih datoteka i direktorijuma koje koristi instalacioni proces.

Skup aktivnosti koje obavlja instalacioni proces zavisi od formata programskog paketa. Na primer, ukoliko se program isporučuje u binarnom obliku (preveden i povezan), instalacioni proces ne uključuje korake prevođenja i povezivanja, a često ni kreiranja privremenih direktorijuma.

Ukoliko paket uključuje veći broj datoteka ovi koraci mogu biti veoma komplikovani, a ručno instaliranje se može izvesti samo ukoliko se prethodno pročita obimna dokumentacija. Da bi se postupak instalacije pojednostavio koriste se specijalni alati za instalaciju softverskih paketa - paket menadžeri (package managers). Ovi alati najčešće koriste specifičan format datoteke - paketa, kao što su RPM i DEB.

Svaki sistem za upravljanje paketima treba da poseduje mogućnost praćenja svakog dela softvera na operativnom sistemu, kao i da poseduje sledeće funkcije:

- instaliranje novih paketa,
- deinstaliranje (uklanjanje) starih ili neželjenih paketa,
- nadogradnja paketa novijim verzijama (update, odnosno upgrade),

- prikazivanje informacija o instaliranim paketima.

U nastavku teksta dat je kraći opis najznačajnijih formata paketa, kao i uputstvo za korišćenje odgovarajućih paket menadžera.

Osobine standardnih formata paketa

tarball (tgz, tar.gz)

Paket ili preciznije rečeno - arhiva ovog formata kreira se pomoću tar arhivera, koji je detaljno opisan u poglavlju "Arhiviranje i backup". Tarball pakete odlikuju sledeće osobine:

- ne mogu se potpisati i ne sadrže checksum;
- datoteke u paketima zadržavaju vlasničke odnose i pristupna prava;
- paket može jednostavno kreirati i raspakovati standardnim Linux alatima tar i gzip. Komanda file ne prepoznaje tar.gz kao poseban format, već kao gzip komprimovanu datoteku;
- paket ne sadrži metadata informacije;
- veličina paketa je određena parametrima sistema datoteka.

Tarball paketi se moraju ručno raspakovati, nakon čega se pokreće odgovarajući instalacioni program (na primer, sh install.sh) ili program za prevođenje izvornog koda (make). U okviru paketa obično se nalazi datoteka koja opisuje dalji postupak instalacije nakon raspakivanja paketa.

Sledeći primer ilustruje instaliranje programskog paketa bashburn (front-end za cdrecord sa tekstualnim menijima):

```
# mkdir /root/bashburn-1.3
# cd /root/bashburn-1.3
# tar -xvf /root/newpackages/bashburn-1.3.tar.gz
# sh install.sh
```

RPM

Svaki program koji dolazi u RPM formatu ima svoje ime, odnosno labelu (package label) koja ga jednoznačno identifikuje. Primeri mogućih labela su:

```
nls-1.0-1
perl-5.001m-4
```

Svaka od ovih labela sastoji se iz imena programa, verzije programa i revizije verzije programa. Ime programa može biti puno ime ili adekvatna skraćenica, razdvojeno je od ostatka labele crtom. Nakon imena sledi verzija programa i na kraju revizija iste verzije, odnosno broj koji pokazuje koliko je puta jedna verzija dorađivana ili ispravljana.

U opštem slučaju, RPM pakete odlikuju sledeće osobine:

- paketi se mogu potpisati i sadrže checksum;
- datoteke u paketima zadržavaju vlasničke odnose i pristupna prava;
- komanda file prepoznaje RPM format paketa. Za rad sa paketima koristi se Red Hat Package Manager;
- RPM paketi sadrže sledeće metadata informacije: ime i verziju paketa, opis, informacije o zavisnosti od drugih paketa (package dependencies) i konfliktima sa drugim paketima (conflicts) i copyright informacije;
- u RPM pakete uključena je prateća dokumentacija, konfiguracione datoteke i skriptovi koji se izvršavaju pre i posle same instalacije, odnosno pre i posle uklanjanja programa;
- format je skalabilan po pitanju korisnih podataka (veličina paketa je određena parametrima sistema datoteka) i metadata informacija-

deb

Debian paketi se koriste na svim Linux sistemima u čijoj je osnovi Debian Linux, a odlikuju se sledećim osobinama:

- paketi se mogu potpisati i sadrže checksum;
- datoteke u paketima zadržavaju vlasničke odnose i pristupna prava;
- komanda file prepoznaje deb format paketa. Za rad sa paketima koristi se Debian Package Manager;
- deb paketi, kao i RPM, od metadata informacija sadrže ime i verziju paketa, opis, informacije o zavisnosti od drugih paketa (package dependencies) i konfliktima sa drugim paketima (conflicts), ali za razliku od RPM paketa ne sadrže informacije o autorskim pravima (Debian je ipak GNU Linux);
- u deb pakete uključene su konfiguracione datoteke i skriptovi koji se izvršavaju pre i posle same instalacije, odnosno pre i posle uklanjanja programa;
- format je skalabilan po pitanju korisnih podataka (veličina paketa je određena parametrima sistema datoteka) i metadata informacija.

Rad sa paket menadžerima

Red Hat Package Manager. Debian Package Management system.

U nastavku teksta dat je kraći opis alata za rad sa RPM i deb paketima. Ovi alati su u početku korišćeni kao deo specifične distribucije po kojoj su dobili i ime, ali se danas koriste u skoro svim Linux distribucijama.

RPM (Red Hat Package Manager)

RPM (Red Hat Package Manager) kreira i održava bazu svih instaliranih paketa i svih datoteka koje su sadržane u tim paketima, i time omogućava lako uklanjanje i osvežavanje paketa. RPM koristi istoimeni format paketa u koji korisnici mogu da spakuju izvorni kod novih programa ili izvršne binarne datoteke. Za razliku od prethodnih verzija, verzija 2 programa RPM napisana je na programskom jeziku C i kao takva se ne oslanja na Perl. Poseban dodatak RPM paket menadžera je rpmlib, odnosno kolekcija RPM rutina koje se mogu koristiti iz drugih programa a služe za rad sa RPM paketima.

Instaliranje RPM paketa

Za instaliranje RPM paketa koristi se sledeća komanda:

```
$ rpm -i [options] pack1.rpm [pack2.rpm ... [packN.rpm]...]
```

odnosno:

```
$ rpm -install [options] pack1.rpm [pack2.rpm ... [packN.rpm]...]
```

Argumenti pack1.rpm, pack2.rpm, ... packN.rpm predstavljaju imena paketa koje će RPM instalirati.

Gore pomenuta komanda instalira RPM paket i to u sledećim koracima:

- proverava da li su instalirani paketi bez kojih ovaj paket neće raditi (takozvani Depedecies check) i istovremeno proverava da li će ovaj paket praviti probleme drugim paketima,
- proverava konflikte, odnosno proverava da li će paket biti instaliran preko već postojećeg ili novije verzije istog i da li će možda neka datoteka biti prepisana,
- izvšava komande koje su potrebne pre same instalacije paketa,
- podešava konfiguracione datoteke,
- otpakuje datoteke i smešta ih na odgovarajuća mesta na disku (inače, ovo je korak na koji najveći broj ljudi pomisli kada se pomene instalacija paketa),
- izvršava komande koje su potrebne posle same instalacije datoteka,
- tokom instalacije rpm upisuje u bazu šta je sve urađeno.

Kao dodatne opcije mogu se navesti:

--percent	prikazivanje procenata tokom instalacije
--excludedocs	instalacija paketa bez dokumentacije
--includedocs	instalacija paketa sa dokumentacijom
--replacepkg	zamena paketa istim (copy)
--force	ignorisanje konflikata među paketima i datotekama

--noscripts	instalacija bez izvršavanja pre-install i post-install skriptova
--prefix <path>	dodavanje prefiksa <path> pre destinacione putanje
--nodeps	instaliranje paketa bez prethodne provere prisutnosti paketa koji su ovom paketu neophodni za rad
--test	testiranje instalacije, bez stvarne instalacije.

Sledeći primer ilustruje komandu kojom se instalira paket eject-1.2-2.i386.rpm:

```
$ rpm -i eject-1.2-2.i386.rpm
```

RPM će pravilno instalirati ovaj paket čak i ako mu se promeni ime. Takođe, paket se može instalirati i ako se datoteka nalazi na nekom ftp serveru:

```
$ rpm -i ftp://ftp.server.com/directory/packet.rpm
```

Pri tome, opcijom --ftpproxy host može se specificirati host kao ftp proxy server.

Uklanjanje paketa iz sistema

Komandom:

```
$ rpm -e pack1.rpm [pack2.rpm ... [packN.rpm]...]
```

odnosno

```
$ rpm --erase pack1.rpm [pack2.rpm ... [packN.rpm]...]
```

može se ukloniti jedan ili više paketa iz sistema. Prilikom uklanjanja paketa, RPM obavlja sledeće aktivnosti:

- proverava u RPM bazi da neki drugi paket ne zavisi od paketa izabranog za brisanje;
- izvršava preuninstall skript (ako postoji);
- proverava da li je neka od konfiguracionih datoteka promenjena (ako jeste, pravi kopiju te datoteke);
- proverava u RPM bazi za svaku datoteku koja se vodi kao deo paketa da li pripada još nekom paketu. Ukoliko datoteka ne pripada drugim paketima, briše je;
- izvršava postuninstall skript (ako postoji);
- briše sve podatke o paketu iz RPM baze.

Sledeći primer ilustruje brisanje paketa eject:

```
$ RPM -e eject
```

Ova komanda će obrisati paket, ali na ekranu neće prikazivati nikakve poruke o izvršenim aktivnostima. Ukoliko korisnik koji uklanja paket želi da isprati proces na ekranu, može da zada sledeću komandu:

```
$ RPM -evv eject
D: uninstalling record number 286040
```

```
D: running preuninstall script (if any)
D: removing files test = 0
D: /usr/man/man1/eject.1 - removing
D: /usr/bin/eject - removing
D: running postuninstall script (if any)
D: removing database entry
D: removing name index
D: removing group index
D: removing file index for /usr/bin/eject
D: removing file index for /usr/man/man1/eject.1
```

Brisanje paketa može biti simulirano sledećom komandom:

```
$ RPM -e -test eject
```

Nadogradnja (upgrade) paketa

Nadogradnja, odnosno upgrade paketa je postupak identičan instalaciji paketa, s tim što se pre instalacije briše starija verzija paketa. Primer ilustruje nadogradnju paketa eject novom verzijom:

```
$ RPM -U eject-1.2-2.i386.rpm
```

deb (The Debian package management system)

Debian package management sistem je skup alata za rad sa deb paketima koji koristi APT biblioteke (Advanced Package Toolkit). U ove alate spadaju: dpkg, apt-get, aptitude, synaptic (front-end za grafičko okruženje), dselect i tasksel.

apt-get

apt-get je alat za rad sa paketima iz komandne linije, u odgovarajućoj stranici uputstva definisan kao korisnički front-end za ostale alate koji koriste APT biblioteke. Komandom apt-get mogu se izvršiti sledeće aktivnosti:

- sinhronizacija indeksa paketa (sinhronizaciju je potrebno izvršiti pre nadogradnje paketa)

```
$ apt-get update
```

- nadogradnja svih paketa koji su trenutno instalirani

```
$ apt-get upgrade
```

- primena svih izmena načinjenih u statusnim poljima paketa programom dselect

```
$ apt-get dselect-upgade
```

- instaliranje novog paketa (napomena: apt-get će instalirati i sve pakete od kojih obeleženi paket zavisi).

```
$ apt-get install package
```

- uklanjanje instaliranog paketa

```
$ apt-get remove package
```

- provera zavisnosti paketa (dependencies)

```
$ apt-get check
```

Sledeći primer ilustruje komandu kojom se na sistem instalira paket aptitude:

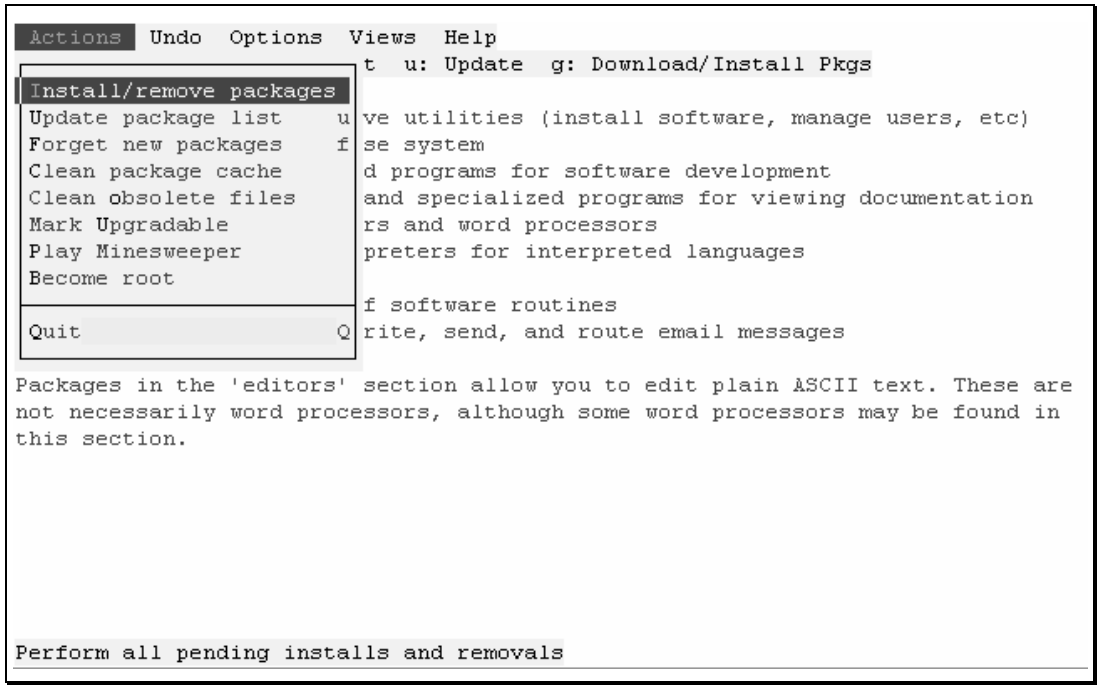
```
# apt-get install aptitude
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  libsigc++0
The following NEW packages will be installed:
  aptitude libsigc++0
0 packages upgraded, 2 newly installed, 0 to remove and 0  not
upgraded.
Need to get 940kB of archives. After unpacking 3127kB will be used.
Do you want to continue? [Y/n] y
Get:1 http://http.us.debian.org stable/main libsigc++0 1.0.4-3
[28.0kB]
Get:2 http://http.us.debian.org stable/main aptitude 0.2.11.1-2
[912kB]
Fetched 940kB in 37s (25.1kB/s)
Selecting previously deselected package libsigc++0.
(Reading database ... 21844 files and directories currently
installed.)
Unpacking libsigc++0 (from .../libsigc++0_1.0.4-3_i386.deb) ...
Selecting previously deselected package aptitude.
Unpacking aptitude (from .../aptitude_0.2.11.1-2_i386.deb) ...
Setting up libsigc++0 (1.0.4-3) ...

Setting up aptitude (0.2.11.1-2) ...
```

Sledeći primer ilustruje komandu kojom se uklanja paket quota:

```
# apt-get remove quota
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  quota
0 packages upgraded, 0 newly installed, 1 to remove and 0  not
upgraded.
Need to get 0B of archives. After unpacking 836kB will be freed.
Do you want to continue? [Y/n] y
(Reading database ... 22111 files and directories currently
installed.)
Removing quota ...
Stopping quota service: rpc.rquotad.
Turning off quotas.
```

Aptitude

Za razliku od programa apt-get, program aptitude je front-end sa sistemom menija, sličan programu dselect. Prilikom rada sa programom aptitude, u padajući meni se ulazi pritiskom na taster F10. Na slici 12.1 prikazan je radni ekran programa aptitude:



Slika 12.1 *Radni ekran programa Aptitude*

Aptitude dozvoljava korisniku da pregleda:

- sve trenutno instalirane pakete,
- dostupne pakete koji nisu instalirani,
- zastarele pakete,
- grupe paketa (task-ove).

Glavni padajući meni je meni action iz koga korisnik može pregledno instalirati i uklanjati pakete. Takođe, aptitude se može koristiti i iz komandne linije, slično programu apt-get. Sledeći primeri ilustruju takvu primenu programa aptitude:

- instaliranje paketa kao i svih paketa od kojih taj paket zavisi

```
$ aptitude -R -G install paket
```

Ova komanda obavlja istu funkciju kao i:

```
$ apt-get install paket
```

- prikazivanje informacija o paketu

```
$ aptitude show paket | less
```

Ova komanda obavlja istu funkciju kao i:

```
$ apt-cache show paket | less
```

- brisanje paketa sa čuvanjem konfiguracionih datoteka

```
$ aptitude remove paket
```

- brisanje paketa sa svim konfiguracionim datotekama

```
$ aptitude purge paket
```

dpkg

U dokumentaciji dpkg je definisan kao alat srednjeg nivoa (medium-level tool) kojim se mogu instalirati, izgraditi i ukloniti Debian GNU/Linux paketi. Rad alata dpkg se kontroliše isključivo preko parametara komandne linije, tako da se dpkg najčešće ne koristi direktno, već preko prijateljski nastrojenog front-end alata po imenu dselect.

Alat dpkg čuva informacije o dostupnim paketima. Ove informacije se najčešće modifikuju alatom dselect, a mogu se podeliti u tri kategorije:

- status paketa: installed (paket je instaliran), half-installed (paket je delimično instaliran, odnosno instalacija je započeta, ali nije završena), not-installed (paket nije instaliran), unpacked (paket je raspakovan, ali nije obavljena konfiguracija), half-configured (paket je raspakovan i delimično konfigurisan, odnosno, konfiguracija je započeta ali nije završena) i config-files (na sistemu su od datog paketa ostale samo konfiguracione datoteke);
- status odabira paketa: install (paket je odabran za instaliranje), deinstall (paket je odabran za uklanjanje, pri čemu se konfiguracione datoteke ne uklanjaju) i purge (paket je odabran za uklanjanje, pri čemu se brišu sve datoteke, uključujući i konfiguracione);
- flegovi paketa: hold (paket sa ovim flegom nije namenjen za instalaciju pomoću programa dpkg, tako da se pomoću njega ne može instalirati, osim ukoliko se ne navede --force-hold opcija), reinst-required (paket je pokvaren i zahteva ponovnu instalaciju - ovakvi paketi se mogu ukloniti jedino ako se navede opcija --force-reinstreq).

Sintaksa komande dpkg je sledeća:

```
$ dpkg [options] action
```

Komanda dpkg zahteva da se navede jedan akcioni parametar. U zavisnosti od navedenog parametra dpkg može da:

- instalira paket (kompletan proces raspakivanja i konfigurisanja)

```
$ dpkg -i | --install package_file...
```

- raspakuje paket bez konfigurisanja

```
$ dpkg --unpack package_file ...
```

- konfiguriše paket. Napomena: ukoliko se umesto imena paketa navede argument -a ili --pending, dpkg će konfigurisati sve raspakovane ali nekonfigurisane pakete (paketi sa statusom unpacked ili half-configured).

```
$ dpkg --configure package ... | -a | --pending
```

- ukloni paket (argument -P povlači uklanjanje konfiguracionih datoteka). Napomena: ukoliko se umesto imena paketa navede argument -a, ukloniće se svi raspakovani paketi obeleženi za uklanjanje (paketi sa statusom deinstall ili purge).

```
$ dpkg  -r  |  --remove  |  -P  | --purge package ... | -a |
```

- prikaže informacije o paketu

```
$ dpkg  -p|--print-avail package
```

- prikaže status paketa

```
$ dpkg -s | --status package-name ...
```

dselect

Alat dselect je front-end za dpkg, odnosno okruženje sa tekstualnim menijima iz koga korisnik može:

- odabrati medijum, odnosno metod instalacije (CD-ROM, NFS, harddisk, montirani sistem datoteka, APT aquisition),
- nadograditi listu dostupnih paketa,
- odabrati pakete koje želi da instalira,
- instalirati i nadograditi odabrane pakete,
- konfigurisati sve pakete koji nisu konfigurisani,
- ukloniti neželjene pakete sa sistema.

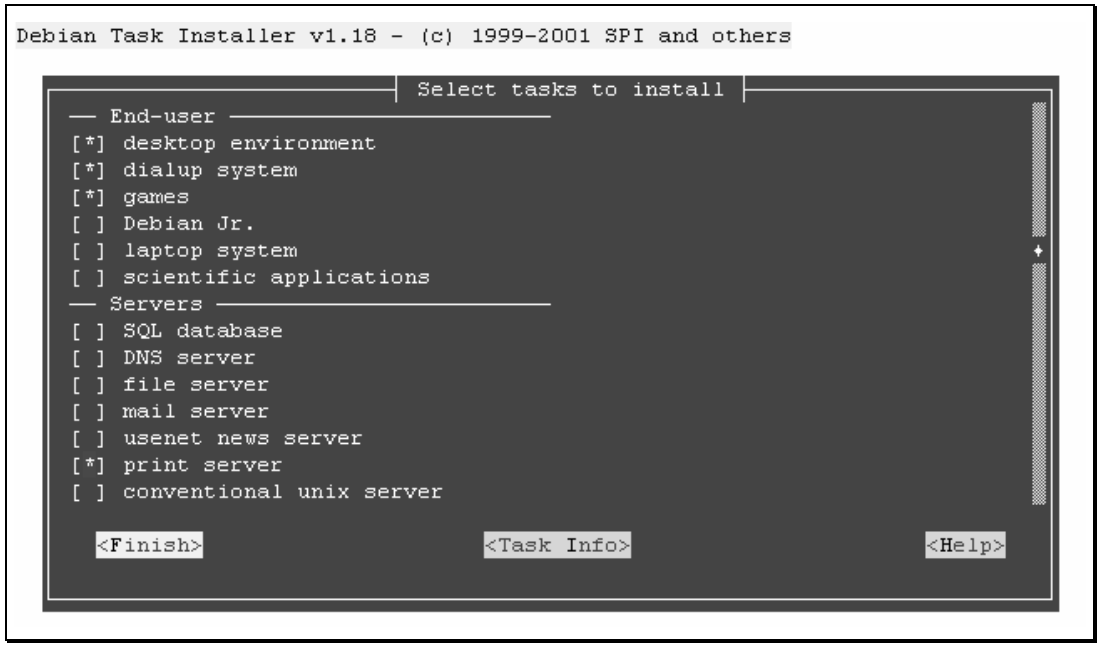
Na slici 12.2 prikazan je radni ekran programa dselect.

```
Debian `dselect' package handling frontend.

   0. [A]ccess     Choose the access method to use.
   1. [U]pdate     Update list of available packages, if possible.
   2. [S]elect     Request which packages you want on your system.
   3. [I]nstall    Install and upgrade wanted packages.
   4. [C]onfig     Configure any packages that are unconfigured.
   5. [R]emove     Remove unwanted software.
 * 6. [Q]uit       Quit dselect.

Move around with ^P and ^N, cursor keys, initial letters, or digits;
Press <enter> to confirm selection.   ^L redraws screen.

Version 1.9.21 (i386).
Copyright (C) 1994-1996 Ian Jackson.
Copyright (C) 2000 Wichert Akkerman.
This is free software; see the GNU General Public Licence version 2
or later for copying conditions.  There is NO warranty.  See
dselect --licence for details.
```

Slika 12.2 *Radni ekran programa dselect*

tasksel

Za razliku od prethodno pomenutih alata pomoću kojih se instaliraju pojedinačni paketi, program tasksel omogućava korisniku da odabere predefinisane grupe paketa (task) koje naknadno može instalirati na sistem. U ove predefinisane grupe paketa spadaju aplikacije za krajnjeg korisnika (aplikacije za desktop sisteme, laptop sisteme, igre, dial-up) i serveri (SQL, DNS, file, mail, print server). Na slici 12.3 prikazan je radni ekran programa tasksel.



Slika 12.3 *Radni ekran programa tasksel*

13

KONFIGURISANJE JEZGRA LINUX SISTEMA

Linux je veoma prilagodljiv - većina parametara sistema, počev od izgleda ekrana a zaključno sa podrškom za odgovarajući hardver mogu se podesiti tako da odgovaraju potrebama korisnika. Prilikom konfigurisanja kernela korisnik pravi kompromise. Na primer, na sistemu može biti instalirano manje jezgro koje radi brže ali podržava samo osnovni hardver, ili ono koje podržava više hardverskih uređaja ali je veće i radi sporije. Takođe, sistem se može optimizovati za prosleđivanje IP paketa ili uobičajne aktivnosti radne stanice sa grafičkim okruženjem. Ovi kompromisi su slični kompromisima koje korisnici prave prilikom rada sa raznim softverom: na primer, prilikom arhiviranja može se koristiti veći stepen kompresije koji opterećuje procesor, tako da program radi sporo ili manji stepen kompresije s kojim program radi brže.

Jezgro se konfiguriše tako da njegova veličina, brzina i hardverska podrška koju pruža budu u ravnoteži. Tu ravnotežu određuje proizvodač konkretne Linux distribucije na osnovu svoje predstave o prosečnom pripadniku ciljne korisničke grupe.

Rad sa modulima

Programski moduli jezgra. Dodavanje i uklanjanje modula iz kernela. Šta se ne može rešiti modulima?

Većina početnika smatra da je konfigurisanje kernela komplikovana procedura. Linux je danas mnogo pristupačniji nego što je bio do pre nekoliko godina - jednostavan postupak instaliranja, user-friendly grafičko okruženje, i hardverska podrška, učinili su Linux pristupačnim velikom broju korisnika. U postupku konfigurisanja kernela koji je ovde opisan, koristi se okruženje za podešavanje parametara kernela vođeno menijem i jedna komandna linija za prevođenje izvornog koda jezgra kao i relativno jednostavan skup komandi za učitavanje programskih modula u kernel koji se izvršava.

Programski moduli jezgra

Programski moduli jezga su ključni deo Linux sistema koji koriste modularni kernel. Programski moduli omogućavaju dodavanje funkcionalnosti jezgru bez ponovnog prevođenja izvornog koda jezgra. Na primer, ukoliko postoji potreba za povezivanjem SCSI uređaja na sistem, programski modul za podršku odgovarajućeg SCSI interfejsa se može učitati u jezgro jednom komandom (insmod). Inače, ukoliko se koristi monolitni kernel, Linux zahteva da se celo jezgro ponovo prevede sa uključenom podrškom za taj uređaj. S obzirom na veliki broj hardverskih uređaja koje Linux podržava programskim modulima je omogućeno:

- da jezgro ostane relativno malo (mikrokernel arhitektura),
- da korisnici na relativno jednostavan način dodaju podršku za svoj hardver.

Dodatno, velika prednost Linuxa u odnosu na druge operativne sisteme leži u mogućnosti dodavanja ili uklanjanja podrške za hardver, sisteme datoteka itd., bez ponovnog podizanja sistema.

Komande za rad sa modulima

U nastavku teksta ukratko su opisane komande za rad sa programskim modulima jezgra. Dodatne informacije o ovim komandama, kao i o samim modulima, mogu se naći u pratećoj dokumentaciji (man pages).

Komanda lsmod (list modules) prikazuje spisak svih programskih modula instaliranih u jezgru koje se trenutno izvršava.

```
# lsmod
Module            Size    Used by        Not tainted
smbfs            32608     0  (unused)
parport          15072     0  (unused)
3c59x            25448     1
```

Modul se dodaje u jezgro komandom insmod (insert module). Komanda kao parametar zahteva ime modula i opcione parametre za definisanje ponašanja modula (na primer, IRQ i adresu hardverskog uređaja). Sintaksa komande insmod je:

```
$ insmod options [-p] module [symbol=value ...]
```

Parametri značajni za komandu insmod (poput dodatnih direktorijuma za smeštaj modula) mogu se specificirati u konfiguracionoj datoteci /etc/modules.conf. Ukoliko se navede parametar -p komanda insmod testira da li se modul može učitati u jezgro bez samog učitavanja modula. Dodatno, mogu se specificirati vrednosti značajnih simbola za modul koji se učitava. Sledeći primer ilustruje dodavanje programskog modula reiserfs kojim se aktivira podrška za ReiserFS sistem datoteka.

```
# insmod reiserfs
Using /lib/modules/2.4.19-686/kernel/fs/reiserfs/reiserfs.o
# lsmod
Module                  Size  Used by    Not tainted
reiserfs              157568   0  (unused)
```

```
smbfs                   32608   0  (unused)
parport                 15072   0  (unused)
3c59x                   25448   1
```

Nakon izvršenja ove komande sistem je spreman za aktiviranje i korišćenje ReiserFS sistema datoteka, bez naknadnog zaustavljanja i podizanja sistema (reboot).

Napomena: najčešće se moduli za podršku sistema datoteka učitavaju sami, prilikom aktiviranja sistema datoteka (mount). Realnija situacija je dodavanje modula za zvučnu ili grafičku karticu, SCSI kontroler ili neki drugi hardver. Moduli tog tipa se po pravilu ne učitavaju sami, već ih korisnik dobija na pratećim diskovima uz odgovarajući hardver.

Pošto se moduli učitavaju u kernel u toku izvršavanja, oni se na disku čuvaju u prevedenom obliku. Svaki programski modul ima ekstenziju ".o" (na primer, reiserfs.o). Većina modula se nalazi u direktorijumu /lib/modules/version-architecture (na primer /lib/modules/2.4.19-686). U okviru ovog direktorijuma moduli su zavisno od namene smešteni u odgovarajuće poddirektorijume (moduli kojima se uključuje podrška za sisteme datoteka nalaze se u direktorijumu fs, moduli kojima se uključuju mrežne funkcije u direktorijumu net, drajveri za hardver u direktorijumu drivers itd.)

Moduli se iz jezgra uklanjaju komandom rmmod (remove module). Pri uklanjanju modula potrebno je obratiti pažnju na međusobnu zavisnost modula (module dependencies), koja se može videti u izlazu komande lsmod. Sledeći primer ilustruje uklanjanje modula reiserfs iz jezgra, čime se uklanja podrška za ReiserFS.

```
# rmmod reiserfs
# lsmod
Module                  Size  Used by    Not tainted
smbfs                  32608   0  (unused)
parport                15072   0  (unused)
3c59x                  25448   1
```

Alternativno, moduli se mogu učitati u kernel pomoću programa depmod i modprobe. Komandom depmod pravi se datoteka u kojoj je opisana međusobna zavisnost programskih modula. Nakon toga, moduli se učitavaju u kernel komandom modprobe.

Koji se problemi ne mogu rešiti modulima?

U nastavku teksta opisani su osnovni problemi koji se ne mogu rešiti modulima, već samo prevođenjem Linux kernela.

- Na procesoru postoji greška (bug) koja povremeno izaziva krah sistema. Problem se rešava preuzimanjem zakrpe (patch) za kernel sa Interneta i prevođenjem izvornog koda.
- Na sistem je instaliran nov SCSI kontroler. Kontroler ima podršku u programskom modulu, ali se sistem ne može podići sa SCSI diska koji je vezan na taj kontroler ukoliko modul nije deo minimalnog kernela, odnosno ukoliko modul nije ugrađen u kernel. Problem se rešava izgradnjom novog jezgra u kom je SCSI podrška ugrađena, a ne napravljena kao samostalnl modul.

- Pojavljuje se problem sigurnosti konkretne verzije kernela. Problem se rešava preuzmanjem odgovarajuće zakrpe sa Interneta i prevođenjem jezgra, čime se povećava sigurnost računara.
- Korisnici sistema se žale da ne mogu da pristupe nekim sistemima datoteka, ukoliko pre toga ne upotrebe komande insmod i mount. Problem se rešava prevođenjem jezgra sa ugrađenom podrškom za odgovarajuće sisteme datoteka, tako da se ti sistemi datoteka mogu upotrebiti bez izričitog dodavanja programskih modula jezgra. Nakon toga je potrebno modifikovati sadržaj /etc/fstab datoteke.
- Nadogradnja jezgra novom verzijom

Prevođenje kernela

Osnovni pojmovi vezani za prevođenje kernela. Pronalaženje novih verzija kernela.

Linux je jedan od malog broja operativnih sistema koji dozvoljava korisnicima da promene osnovne parametre rada računara. Izvorni kôd Linux kernela isporučuje se sa svakom distribucijom - korisnik može modifikovati izvorni kôd i nakon toga ga prevesti (kompajlirati), čime se Linux prilagođava potrebama konkretnog korisnika. Ovo poglavlje opisuje postupke podešavanja parametara kernela pomoću uslužnih programa i prevođenja modifikovanog izvornog koda.

Terminologija prevođenja

Postupak stvarnog prevođenja izvornog koda projekta u oblik koji računar može da upotrebi (binarnu datoteku) zove se prevođenje (compiling). Za prevođenje izvornog koda koriste se standardni prevodioci jezika C, kao što je gcc.

Prilikom izrade softvera pod Linux sistemom koristi se konfiguraciona datoteka po imenu makefile koja opisuje kako treba uklopiti napisane delove koda i sve korišćene biblioteke. Datoteku makefile upotrebljava uslužni program make prilikom prevođenja programa. Medutim, pošto programski projekat obično ima veliki broj komponenti, uslužni program make proverava njihove datume i vremena nastanka, i ponovo prevodi samo one komponente koje su bile izmenjene nakon poslednjeg prevođenja. Time se štedi mnogo vremena pri izradi nove verzije nekog projekta, ukoliko je bio izmenjen samo deo izvornog koda. Ukoliko postoji potreba, programu make može se naznačiti da ponovo prevede sve komponente projekta.

Priprema za izradu novog jezgra

Pronalaženje novih verzija kernela

Većina Linux distribucija ne instalira izvorni kôd jezgra automatski - ukoliko korisnik želi da prevede kernel, mora najpre da instalira programski paket u kome se nalazi izvorni kôd (na primer, kernel-source-version-architecture.rpm).

Linuxovo jezgro se može izmeniti svakog dana. Većina izmena u Linux kernelu dešava se u razvojnim verzijama jezgra - razvojnu verziju jezgra ne treba isprobavati na računarima koji imaju konkretne zadatke. Dodatno, ukoliko je sistem stabilan, ove promene ne treba pratiti, osim ako se ne radi o značajnim promenama. Dokumentacija i najnovije verzije kernela mogu se naći na web stranicama www.kernel.org i www.kernelhq.com.

Uz sam izvorni kôd, neophodno je da na sistemu budu instalirani i prevodilac (compiler) za prevođenje C izvornog koda (na primer gcc) i uslužni program make.

Snimanje starog kernela

Iako se ovde implicira jednostavnost prevođenja kernela, pre samog prevođenja potrebno je obezbediti način za podizanje tekućeg kernela. Kada se kernel ponovo izgradi, automatski se kreira kopija stare verzije sa ekstenzijom .old. Medutim, ta kopija nije automatski dostupna za podizanje sistema, tako da je neophodno napraviti kopiju kernela i dodati nove stavke u konfiguraciju programa za podizanje sistema (boot loader).

Izrada rezervne kopije kernela je jednostavna - jezgro se komandom cp kopira u datoteku sa drugim imenom.

Napomena: datoteka vmlinuz u root direktorijumu aktivnog UNIX stabla je simbolički link na kernel koji se nalazi u direktorijumu /boot (vmlinuz -> boot/vmlinuz-2.4.26-686). Korisnik može ukloniti ovaj link, a zatim napraviti novi pod drugim imenom, koji će ukazivati na staro jezgro.

```
$ rm /vmlinuz
$ ln -s boot/vmlinuz-2.4.26-686 /vmlinuz.old
```

Nakon toga se u datoteku lilo.conf dodaje još jedna stavka, čime se konfiguriše LILO, odnosno omogućava podizanje sistema sa starim kernelom. Ovo se najlakše može izvesti tako što se ceo odeljak koji se odnosi na staro jezgro iskopira, a zatim se u kopiji i u originalu izmene imena jezgra, odnosno labela. Time se boot loaderu naznačava putanja i ime još jednog jezgra sa kojim se sistem može podići.

```
default=LinuxOld
image=/vmlinuz.test
   label=LinuxTest
   read-only
image=/vmlinuz.old
   label=LinuxOld
   read-only
```

Nakon toga, potrebno je zadati komandu lilo čime se ažurira konfiguracija boot loadera sa datotekom /etc/lilo.conf.

```
# lilo
Added Linux *
Added LinuxOLD
```

Staro jezgro će biti dostupno u grafičkom ili tekstualnom meniju prilikom sledećeg podizanja operativnog sistema kao stavka LinuxOld (ukoliko se koristi LILO u komandnoj liniji, spisak dostupnih kernela i operativnih sistema može se dobiti pomoću tastera TAB).

Napomena: umesto ručnog modifikovanja datoteke /etc/lilo.conf i pokretanja programa lilo, može se koristiti konfiguracioni Perl skript liloconfig.

Izrada novog jezgra

Izrada novog jezgra obuhvata sledeće procedure:

- podešavanje konfiguracije novog jezgra,
- prevođenje nove konfiguracije izvornog koda.

Podešavanje novog jezgra

Pre samog prevođenja potrebno je podesiti konfiguraciju novog kernela. U ovom koraku se odlučuje šta će jezgro sadržati, šta neće sadržati i šta će biti dostupno kao programski modul.

Za podešavanje jezgra može se koristiti klasično okruženje komandne linije ili okruženje sa menijima. Dodatno, većina distribucija uključuje i programe koji rade u grafičkom okruženju (xconfig). U okruženju komandne linije program postavlja jedno po jedno pitanje o tome šta treba da uključi u jezgro. Ovo okruženje je pogodno za one korisnike koji su detaljno upoznati sa kernelom ili vrše konfiguraciju kernela pomoću skriptova. Najveći nedostatak okruženja komandne linije leži u nemogućnosti povratka na prethodno pitanje i promene datog odgovora. Tekstualno okruženje sa menijima koristi višenivovske menije - korisnik može da istraži sve dostupne opcije, može da se vrati na bilo koju stavku menija i izmeni već dati odgovor.

Oba okruženja zahtevaju od korisnika da odgovori na isti skup pitanja i kreiraju istu konfiguracionu datoteku koju uslužni program make koristi prilikom prevođenja i sklapanja kernela. Sve stavke konfiguracije mogu se podeliti na modularne opcije i opcije koje nisu modularne. Opcija koja nema mogućnost da se učita kao modul se ili ugrađuje ili ne ugrađuje u minimalno jezgro. Modularne opcije se mogu isključiti iz jezgra, ugraditi u minimalno jezgro (kasnije se ne moraju učitavati u jezgro jer su uvek deo sistema) ili uključiti kao moduli koji nisu deo minimalnog jezgra (kasnije se mogu dodavati ili uklanjati po svojoj volji iz jezgra koje se izvršava).

Pokretanje okruženja komandne linije

Okruženje komandne linije se pokreće tako što se na sistem prijavi root, zatim pređe u direktorijum u kome se nalazi izvorni kôd kernela (na primer /usr/src/kernel-source-2.4.19) i pokrene komandu make config.

```
$ su -
Password:
# cd /usr/src/kernel-source-2.4.19
# make config
```

Program dalje postavlja pitanja o konfiguraciji kernela. Niz pitanja koje program postavlja zavisi od prethodno datih odgovora. Podrazumevani odgovor na svako pitanje se daje jednostavnim pritiskom na Enter. U svakom slučaju taj izbor je prikazan kao veliko slovo. Na primer, jedno od pitanja je:

```
Networking support (CONFIG_NET) [Y/n/?]
```

Ukoliko korisnik pritisne Enter biće izabrana podrazumevana vrednost Y (Yes - uključi podršku za umrežavanje). Modularne opcije uključuju i odgovor m (realizuj kao modul), dok neka pitanja koja se odnose na uređaje dozvoljavaju samo odgovore M i n. Ukoliko se na pitanje odgovori znakom pitanja (?), program na ekranu prikazuje opisne informacije koje su obično veoma korisne pri odlučivanju o ponuđenim opcijama. Iste opisne informacije dostupne su i u tekstualnom režimu sa menijima.

Osnovne mane ovog načina podešavanja su: nepreglednost, veliki broj pitanja (350 pojedinačnih pitanja, pri čemu ukupan broj varira zavisno od prethodno datih odgovora) i nemogućnost promene već datog odgovora.

Osnovna prednost ovog načina podešavanja je mogućnost kreiranja skriptova koji će programu dati podatke o konfiguraciji kernela.

Pokretanje okruženja sa menijima

Podešavanje putem menija je lakši i pregledniji način izrade konfiguracione datoteke (human-friendly interfejs za konfigurisanje). Okruženje sa menijima se pokreće tako što se na sistem prijavi root, zatim pređe u direktorijum u kome se nalazi izvorni kôd kernela (na primer /usr/src/kernel-source-2.4.19) i pokrene komanda make menuconfig. Alat za podešavanje putem menija zahteva da je na sistem instalirana biblioteka ncurses.

```
$ su -
Password:
# cd /usr/src/kernel-source-2.4.19
# make menuconfig
```

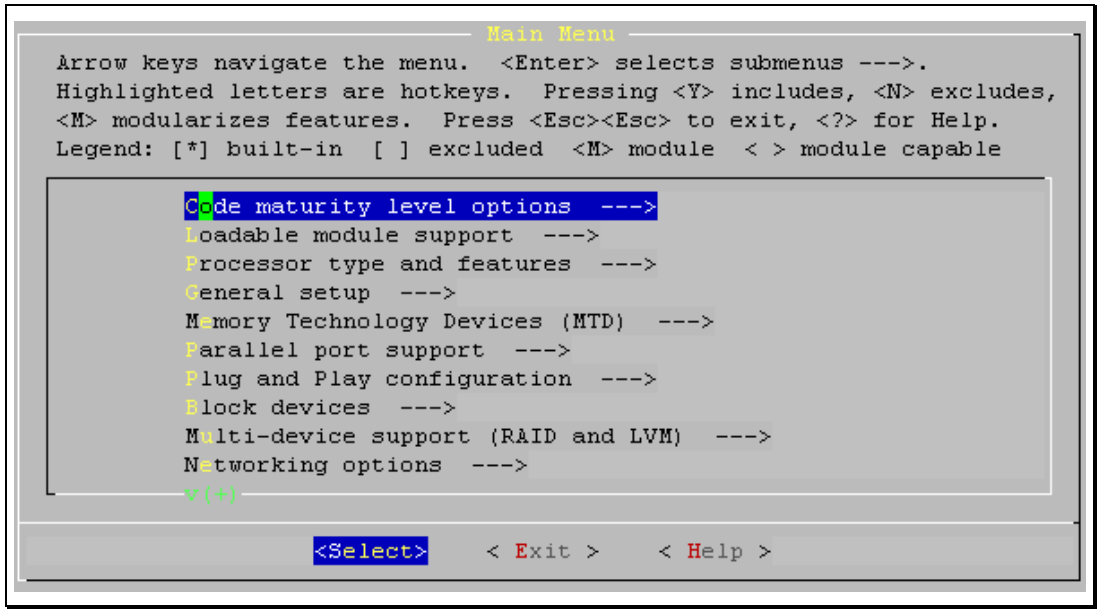
Podešavanje putem menija počinje spiskom kategorija u kojima se biraju opcije kernela.

Za navigaciju po menijima koriste se kursorski tasteri (strelice), za ulazak u podmeni taster Enter. Taster Tab služi za prelazak u deo ekrana sa dugmadima Select (prikazuje podmeni opcija pridruženih istaknutoj stavci), Exit (vraća se na prethodni meni, završavajući podešavanje ukoliko je izabrano sa najvišeg nivoa) i Help (prikazuje pomoćni ekran pun informacija o istaknutoj stavci menija).

Alati za podešavanje jezgra koriste se za izradu konfiguracione datoteke koja se upotrebljava prilikom prevođenja i sklapanja jezgra. Korisnici koji jednom konfigurišu sve parametre kernela mogu da sačuvaju kopiju konfiguracione datoteke za kasniju upotrebu. To se radi pomoću dve poslednje stavke glavnog menija: „Save Configuration to an

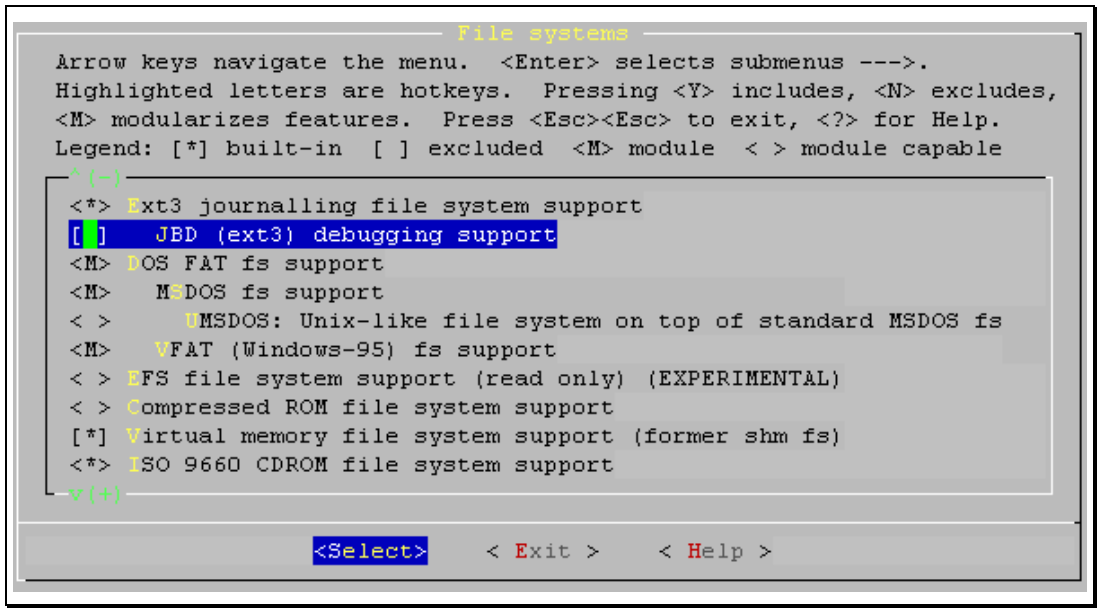
Alternate File" i „Load an Alternate Configuration File", pomoću kojih se može snimiti konfiguracija ili učitati postojeća konfiguraciona datoteka.

Na slici 13.1 prikazan je glavni meni.



Slika 13.1 *Okruženje sa menijima*

Stavke sa podmenijem prikazane na slici 13.1 imaju strelicu sa svoje desne strane. Na primer, izborom opcije File Systems iz glavnog menija otvara se podmeni prikazan na slici 10.2.



Slika 13.2 *Podmeni stavke File systems*

Stavke menija koje određuju neku opciju jezgra, a ne mogu biti modularne, sa leve strane imaju jednu od sledećih oznaka:

[*] opcija je deo jezgra

[ ] opcija nije deo jezgra

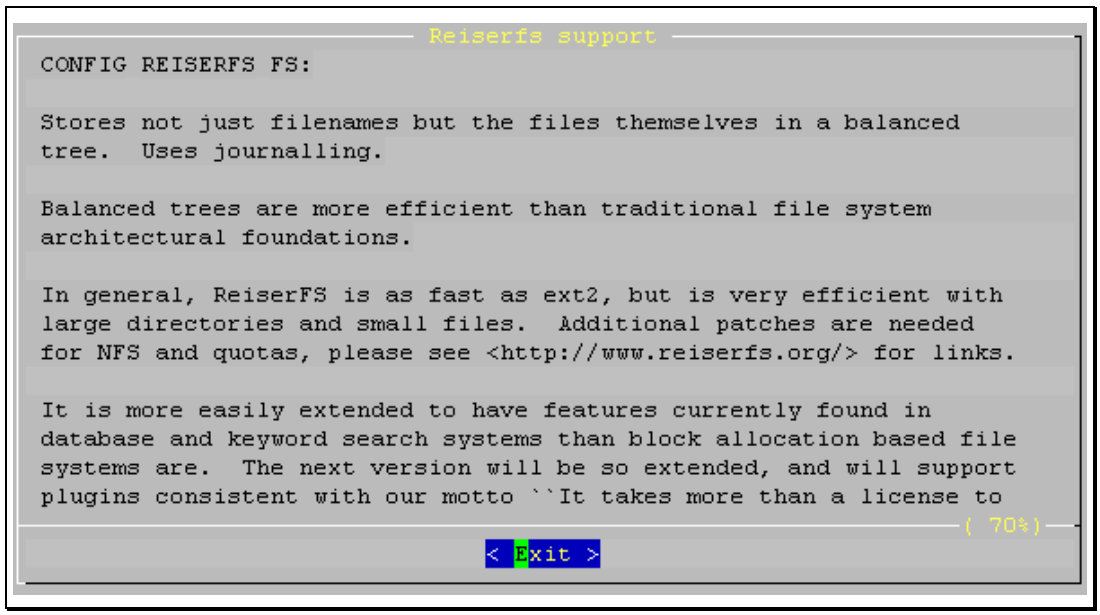
Stavke menija koje određuju neku opciju jezgra, a mogu biti deo jezgra ili realizovane kao modul, sa leve strane imaju jednu od sledećih oznaka:

< > opcija nije uključena kao deo jezgra, niti napravijena kao modul koji može kasnije da se učita;

<*> opcija je deo jezgra i samim tim je uvek deo sistema;

<M> opcija je uključena kao modul, ali nije deo samog jezgra. Modul se kasnije po potrebi može učitati ili ukloniti iz jezgra koje se izvršava.

Oznake sa leve strane ukazuju na trenutno stanje opcije. Stanje se može izmeniti jednim od sledećih tastera: Y (uključi opciju u jezgro), M (uključi opciju kao modul) ili N (opcija se ne uključuje, niti realizuje kao modul). Dodatno, pomoću tastera ? se za tekuću opciju na ekranu prikazuje detaljan opis.



Slika 13.3 *Opis opcije Reiserfs support*

Nakon postavljanja svih opcija na željenu vrednost potrebno je iz glavnog menija odabrati dugme Exit, nakon čega će se na ekranu pojaviti pitanje da li želite da snimite izmenjenu konfiguracionu datoteku.

Slika 13.4 *Napuštanje okruženja sa menijima*

Prevođenje i testiranje novog jezgra

Nakon kreiranja nove konfiguracione datoteke potrebno je prevesti izvorni kôd kernela. Zavisno od konfiguracije računara (brzina procesora, količina memorije, tip i brzina diskova) i verzije kernela, prevođenje i sklapanje jezgra traje od nekoliko minuta do nekoliko sati. Kernel može biti nekomprimovan (image) i komprimovan programom gzip ili bzip2 (zlmage, bzimage), koji je manji, ali se sporije učitava. Komande za prevođenje variraju sa konkretnom Linux distribucijom, ali su u svakom slučaju jednostavne i mogu se, zadate iz jedne komandne linije, izvršavati jedna za drugom. Sledeće komande prevode izvorni kôd, prave novo jezgro i sklapaju sve module jezgra, smeštajući ih u odgovarajuće sistemske direktorijume, čime se obezbeđuje njihova dostupnost pomoću standardnih komandi za rad sa modulima:

```
# make dep; make clean; make zImage; make modules; make
modules_install
```

Nakon izvršenja ovih naredbi novo jezgro je spremno za testiranje i može se iskopirati u /boot direktorijum, nakon čega mu se u root direktorijumu aktivnog UNIX stabla može kreirati i simbolički link na novo jezgro:

```
# cp /usr/src/kernel-source-2.4.19/arch/i386/boot/zImage
/boot/vmlinuz-2.4.19-i386-TEST
# ln -s /boot/vmlinuz-2.4.19-i386-TEST /vmlinuz.test
```

Napomena: ime kopirane slike jezgra može biti proizvoljno, kao i direktorijum u kom se jezgro nalazi. Važno je da se ime jezgra ili simboličkog linka na jezgro i direktorijum poklapaju sa linijom image=/vmlinuz.test u datoteci /etc/lilo.conf.

Nakon toga, sistem se može podići radi testiranja novog jezgra, odnosno isprobavanja novih mogućnosti koje su u jezgro dodate tokom kreiranja konfiguracione datoteke. Testiranje najčešće obuhvata aktiviranje sistema datoteka ili pristup nekom uređaju bez prethodnog učitavanja modula (ukoliko je u jezgro ugrađena podrška za taj uredaj) ili korišćenje mrežnih resursa koji nisu bili raspoloživi u starom jezgru. Dodatno, potrebno je uporediti stepen iskorišćenja sistemske memorije (naredbom free) starog i novog kernela i proveriti vremenski žig tekućeg jezgra naredbom uname -v. Datum i vreme novog jezgra moraju da odgovaraju datumu i vremenu sklapanja jezgra).

```
# free
             total       used       free     shared    buffers     cached
Mem:         62176      59148       3028          0      13520      15936
-/+ buffers/cache:      29692      32484
Swap:       192772      11124     181648
# uname -v
```

```
#1 SMP Sat Apr 24 10:56:05 CEST 2004
```

Napomena: ukoliko naredba uname pokaže da se ne koristi novo jezgro, potrebno je proveriti LILO konfiguracionu datoteku (/etc/lilo.conf) i pokrenuti program lilo, kako bi se na osnovu konfiguracione datoteke ažurirali podaci u boot sektoru.

14

SIGURNOST I ZAŠTITA UNIX I LINUX SISTEMA

Informacije koje poseduje jedna ustanova su jedan od najznačajnijih delova njene imovine. Sigurnost na mreži je postala sve značajnija otkako mreže postaju sve veće i kompleksnije. Pojavom Internet mreže resursi i informacije jedne kompanije postaju dostupne van lokalne mreže, tako da zaštita informacija i resursa postaju veoma značajan deo aktivnosti sistem administratora.

Pregled zaštite UNIX i Linux sistema

Zaštitne polise i standardni mehanizmi zaštite pod UNIX sistemom. Opšta sigurnost Linux sistema.

Opšti cilj informacionih sistema je da obezbede informacije koje su verodostojne i raspoložive uvek kada su potrebne. Sveobuhvatna zaštitna polisa informacionih sistema može da pomogne da se obezbedi raspoloživost verodostojnih informacija isključivo ovlašćenim korisnicima.

Standardni mehanizmi zaštite

Metode napada

Za uspešno administriranje mreže izuzetno je važno razumeti prirodu potencijalnih napada na sistem, odnosno mrežu. Sistem se može napasti na mnogo načina: jednostavnim "rubber-hose" metodama, odnosno ucenjivanjem administratora (razne psihofizičke metode ucene i iznuđivanja informacija se u praksi pokazuju kao vrlo uspešne), kao i sofisticiranim metodama tipa buffer-overflow (prepunjenje bafera). U najčešće korišćene vrste napada na sistem spadaju neautorizovani pristup i eksploatacija poznatih slabosti programa, najčešće mrežnih servisa (telnet, rlogin, rexec). Neautorizovani pristup je vrsta napada koja obuhvata neautorizovano korišćenje resursa računara (najčešće procesorskog

vremena i podataka). Najčešće korišćene metode eksploatacije slabosti programa su DoS, spoofing i sniffing. Administrator može najbolje zaštititi mrežu od napada ove vrste korišćenjem mrežne barijere (firewall). Dodatno, svi mrežni servisi koji nisu pouzdani trebaju biti isključeni ili zamenjeni alternativnim paketom (na primer, telnet se može zameniti paketom ssh).

DoS - denial of service (odbijanje usluga) kao napad izaziva prestanak rada servisa ili programa, čime se drugima onemogućava rad sa tim servisima ili programima. DoS napad se može izvršiti na mrežnom sloju slanjem zlonamernih datagrama kojima se izaziva raskid mrežne konekcije. Ovi napadi se mogu izvršiti i na aplikacionom sloju, slanjem specijalnih komandi programu, što kao posledicu ima prestanak rada programa.

Spoofing je podržavanje akcija napadača od strane servera ili aplikacije - napadač prati IP adrese u IP paketima i predstavlja se kao drugi računar. Kako DNS ne proverava odakle dolaze informacije, napadač može da izrši spoof napad dajući pogrešnu informaciju (ime računara od poverenja) DNS servisu. Najbolja zaštita od ovog napada je sprečavanje rutiranja datagrama sa neispravnim izvorišnim adresama.

Sniffing je metod u kome se specijalnim programima (sniffer) presreću TCP/IP paketi koji prolaze kroz određeni računar i po potrebi pregleda njihov sadržaj. Kako se kroz mrežu obično kreću podaci koji nisu šifrovani, snifer lako može doći do poverljivih informacija.

Zaštitne polise

Zaštitne polise svakog ozbiljnog informacionog sistema uključuju sledeće nivoe zaštite:

- Zaštita na nivou fižičkog pristupa sistemu (Physical Access Security). Prva linija odbrane je zaštita mrežne opreme kao što su serveri, ruteri itd. Ako mrežna oprema nije fizički dostupna neovlašćenim osobama, male su šanse da dođe do slučajnih ili namernih oštećenja.
- Zaštita na nivou prijavljivanja na sistem (Login & Password Security). Na ovom nivou zaštite zahteva se da svaki korisnik koji pristupa radnoj stanici ili serveru, mora imati važeće korisničko ime i lozinku. UNIX sistemi imaju specifične zahteve za kreiranje i promenu lozinki. Sistem administratori mogu zahtevati da se lozinke menjaju periodično.
- Zaštita na nivou sistema datoteka (Filesystem Security). Glavna komponenta u svakoj sveobuhvatnoj zaštitnoj polisi je zaštita na nivou sistemu datoteka, koja određuje ko može pristupiti podacima i šta sa njima može da radi. Sistem administratori postavljaju zaštitu u sistemu datoteka baziranu na korisnicima, grupama i pravima pristupa. Prava pristupa se dodeljuju svakoj datoteci i direktorijumu i na bazi tih prava se određuje nivo pristupa tim objektima.
- Zaštita od virusa (Virus Protection). Virusi mogu krajnje opasno da unište ili oštete pojedinačne radne stanice ili mrežne servere. Mnogi mrežni operativni sistemi i radne stanice zahtevaju antivirus softver za adekvatnu zaštitu. Na sreću, tvorci virusa nisu dovoljno familijarni sa sistemskim programiranjem pod UNIX i Linux okruženjem, tako da retko kreiraju UNIX viruse.

- Zaštita na nivou udaljenog pristupa sistemu (Remote Access Security). Sistem administrator mora da obezbedi udaljeni pristup resursima za legitimne korsinike sistema, ali i istovremeno da zaštiti ove resurse od neautorizovanog pristupa.
- Internet firewalls. Organizacije koje obezbeđu svojim korisnicima pristup Internetu ili održavaju Web sajtove, moraju se zaštititi od zlonamernih napada. U te svrhe se koriste različite vrste firewall softvera i softvera za autentifikaciju korisnika, pomoću kojih se zaustavljaju zlonamerni napadi.
- Rezervne kopije podataka (Data Backups). Redovno kreiranje rezervnih kopija značajnih podataka (backup) minimiziraće vreme oporavka podataka u slučaju oštećenja, gubitka podataka i raznih drugih havarija. Pouzdanost rezervne kopije podataka obezbeđuje se povremenim čitanjem sadržaja backup medijuma ili test-restore procedurama.
- Plan restauracije u slučaju teških havarija (Disaster Recovery Plan). Ovaj plan identifikuje kritične podatke i dokumenta koji su od funkcionalnog značaja za ustanovu. Plan opisuje zaštitne mere i korake potrebne da se u slučaju teške havarije brzo i sa minimalnim gubicima obezbedi normalno funkcionisanje sistema.
- Statistika (Audits). Programi za praćenje statistike omogućavaju sistem administratorima da analiziraju i detektuju slabe tačke zaštite, kao što su datoteke sa otvorenim pravima pristupa.

Standardni mehanizmi zaštite pod UNIX/Linux sistemom

Primarna funkcija sistemske zaštite je zabrana pristupa neovlašćenim korisnicima. Čuvanje informacija u bezbednom stanju značajno je kako za korisnike tako i za sistem administratore. Zaštitom datoteka od neovlašćenog korišćenja, korisnici štite integritet svog rada. U standardne mehanizme zaštite pod UNIX sistemom spadaju:

- Korisnički nalog i lozinka koji ograničavaju pristup sistemu, odnosno dozvoljavaju pristup samo autorizovanim korisnicima.
- Zaštita datoteka i direktorijuma vlasničkim odnosima i pravima pristupa. Sve datoteke i direktorijumi su nakon kreiranja zaštićeni vlasničkim odnosima i pravima pristupa. Na svakom UNIX sistemu postoji specijalni korisnički nalog (root), pomoću kog administrator sistema ostvaruje neograničen pristup celom UNIX sistemu, bez obzira na pristupna prava. Korisnik root se često naziva i superuser i jedini može da izvršava komande za administraciju sistema ili da modifikuje sadržaj kritičnih datoteka kao što je /etc/passwd.
- Kontrola udaljenog pristupa (prijavljivanja na sistem preko mreže - remote login). Na Linux sistemima postoje specijalni programi (ipchains i iptables) koji se mogu koristiti kao “vatreni zid” (firewall). Ovi programi se mogu konfigurisati tako da kontrolišu saobraćaj na osnovu IP adresa, odnosno da određuju koje IP adrese mogu da pristupaju pojedinom sistemu ili grupi sistema. Dodatno se može zabraniti rad nesigurnih servisa za udaljeni pristup, kao što su telnet, ftp i rlogin. Umesto njih treba koristiti programe poput ssh (secure shell), koji šifruje poruke između klijenta i servera, čime se obezbeđuje sigurnost veze, a samim tim i opšta sigurnost sistema.

Programi za analizu sigurnosti sistema

UNIX sistem administratori mogu periodično da proveravaju sistemske log datoteke i da na taj način odrede da li su sistemu pristupali neautorizovani korisnici. Takođe, periodičnim izvršavanjem programa kao što je SATAN (Security Administrators Tool for Analyzing Networks) mogu se skenirati portovi na računarima u mreži kako bi se pronašle slabe tačke, odnosno ranjiva mesta na mreži. Program Tripwire, koji se izvršava kao daemon proces, čuva informacije o statusu sistema i obaveštava administratore ukoliko se nešto neobično dogodi. Tripwire je uključen u Red Hat Linux distribuciju, ali se mora instalirati posebno. Programski paketi SATAN (novije verzije ovog paketa se distribuiraju pod imenom SAINT - Security Administrator's Integrated Network Tool) i Tripwire, kao i veliki broj drugih programa (na primer nmap) mogu se besplatno preuzeti sa Interneta.

Opšta sigurnost Linux sistema

Linux sistem je siguran onoliko koliko ga administrator učini sigurnim.

Nakon eliminisanja potencijalnog rizika sigurnosti uklanjanjem neželjenih mrežnih servisa, potrebno je preuzeti određene mere u cilju povećanja sigurnosti preostalih servisa i softvera na računaru (serveru). U nastavku teksta objašnjeno je nekoliko osnovnih postupaka i tehnika kojima se može povećati opšta sigurnost mrežnog servera. Ovim postupcima administrator sistema može za relativno kratak vremenski period (jedan sat) sprečiti razne vrste napada na sistem.

Sigurnost na nivou BIOS-a

Prvi koraci koje ozbiljni sistem administratori mogu preuzeti su:

- zabrana podizanja operativnog sistema sa flopi diska i CD-ROM uređaja,
- postavljanje lozinke za pristup BIOS konfiguraciji.

Time se sprečava mogućnost podizanja operativnog sistema sa izmenljivog medijuma i sticanje root privilegija na krajnje jednostavan način. Korisnik koji podigne "live" distribuciju Linux sistema sa CD-ROM medijuma može se na sistem prijaviti kao superuser. Nakon toga lako može aktivirati sve lokalne sisteme datoteka i preuzeti i/ili uništiti sve podatke na njima. Zabranom pristupa BIOS rutinama sprečava se mogućnost da zlonamerni korisnik dozvoli podizanje sistema sa izmenljivih medijuma i na taj način stekne root privilegije. Takođe, na ovaj način se može sprečiti podizanje servera bez lozinke. Ova sigurnosna mera se može zaobići ukoliko korisnik ima fizički pristup serveru (uređaju). U tom slučaju korisnik može izvući BIOS bateriju, nakon čega se vrši reset BIOS parametara, odnosno povratak na podrazumevane vrednosti (koje najčešće ne uključuju šifru za pristup BIOS rutinama).

Prevođenje monolitnog kernela

U slučaju mrežnih servera, hardver se ne dodaje često (izuzetak su diskovi i mrežne kartice). Zato se preporučuje da root prevede monolitni kernel u koji će biti uključena podrška za sve što je potrebno da bi server normalno funkcionisao. Nepostojanje modula smanjuje šansu da napadač uključi modul u rezidentni modularni kernel i time omogući neke funkcije koje za taj server nisu predviđene, a koje bi on mogao da iskoristi za dalje eksploatisanje sistema.

Privremeno isključivanje servera sa mreže

Sigurnosne mere ne treba primenjivati dok je server "na mreži". Preporučuje se da root deaktivira mrežne interfejse pre primene sigurnosnih mera. Mrežni interfejs se može deaktivirati sledećom komandom:

```
# ifdown eth0
```

Nakon primene sigurnosnih mera mrežni interfejs se može aktivirati komandom:

```
# ifup eth0
```

Ukoliko na sistemu postoji više mrežnih uređaja svi se mogu isključiti komandom:

```
# /etc/rc.d/init.d/network stop
Shutting down interface eth0 [OK]
Disabling Ipv4 packet forwarding [OK]
```

Nakon primene sigurnosnih mera svi mrežni interfejsi mogu se aktivirati komandom:

```
# /etc/rc.d/init.d/network start
Setting network parameters [OK]
Bringing up interface lo [OK]
Bringing up interface eth0 [OK]
```

LILO i datoteka /etc/lilo.conf

LILO je najčešće korišćeni Linux boot loader, odnosno program koji puni memoriju kernelom operativnog sistema. LILO omogućava podizanje kernela sa flopi i hard diskova, a takođe može poslužiti i kao boot manager za druge operativne sisteme. Kao takav, LILO je vrlo značajan deo Linux sistema i treba ga zaštititi.

Najznačajnija LILO konfiguraciona datoteka je /etc/lilo.conf. Sledeće tri direktive koje se mogu navesti u ovoj datoteci mogu značajno povećati sigurnost LILO programa:

timeout=00 — određuje period čekanja na podizanje podrazumevanog izbora operativnog sistema. C2 nivo sigurnosti zahteva da ovaj period bude 0 ukoliko se podiže samo jedan operativni sistem. Ukoliko se timeout postavi na vrednost 0, LILO prompt će biti nedostupan;

password=pass — direktiva štiti sliku kernela lozinkom. Ukoliko se direktiva password navede, korisnik koji želi da podigne sistem moraće da

navede lozinku pass. LILO lozinke su osetljive na mala i velika slova;

restricted — ukoliko je ova direktiva navedena u datoteci /etc/lilo.conf, LILO će zahtevati lozinku samo u slučajevima učitavanja slike kernela sa dodatnim parametrima iz LILO komandne linije (na primer: linux single). Direktiva se navodi zajedno sa direktivom password za svaku sliku kernela posebno.

Napomena: korišćenje direktive password bez direktive restricted je loša praksa. U tom slučaju je nemoguće izvesti reboot proceduru sa udaljenog računara, jer LILO zahteva lozinku koja se može uneti samo sa tastature servera. Direktiva restricted dozvoljava reboot sa udaljenog računara, jer se lozinka zahteva samo ako se navedu dodatni parametri u LILO promptu. Sledeći isečak datoteke /etc/lilo.conf prikazuje ispravnu LILO konfiguraciju, pod uslovom da se na računaru nalazi samo jedan operativni sistem. U vremenskom periodu od pet sekundi moguće je navesti parametre kernelu, ali se u tom slučaju zahteva lozinka. Nakon tog perioda podiže se operativni sistem bez dodatnih parametara i za to se ne zahteva lozinka.

```
boot=/dev/sda
...
# enable 5 second prompt
prompt
timeout=05
linear
default=linux
restricted
password=CocoNut1
image=/boot/vmlinuz-2.4.26-686
label=linux
read-only
root=/dev/sda2
...
```

Kako se lozinke u ovoj datoteci ne šifruju, potrebno je radi dodatne sigurnosti zaštititi datoteku /etc/lilo.conf adekvatnim pravima pristupa:

```
# chmod 600 /etc/lilo.conf
```

Sledeći korak je ažuriranje boot loadera novim sadržajem datoteke /etc/lilo.conf, odnosno, sa novom konfiguracijom:

```
# lilo
```

Poslednji korak je zabrana slučajne ili namerne izmene sadržaja datoteke /etc/lilo.conf postavljanjem immutability flega komandom chattr:

```
# chattr +i /etc/lilo.conf
```

Nakon toga se sadržaj datoteke /etc/lilo.conf može izmeniti samo ako se ukine immutability fleg, što se može učiniti sledećom komandom:

```
# chattr -i /etc/lilo.conf
```

Korisničke lozinke

Povećanje opšte sigurnosti Linux sistema obično počinje formulisanjem ograničenja nad lozinkama korisnika. Veliki broj korisnika čuva vredne informacije na računaru, a jedino što njihove podatke štiti od neovlašćenog pristupa je niz od osam karaktera - lozinka. Nasuprot opštem mišljenju neprobojna lozinka ne postoji. Postoje samo slabe i jake lozinke, od kojih svaka može biti otkrivena korišćenjem odgovarajućeg softvera (dictionary, brute force) ili alternativnim metodama, koje se najčešće i koriste. Pokretanje programa za otkrivanje lozinki jednom ili dvaput mesečno je dobra mera zaštite - tako se mogu izdvojiti korisnici čije su šifre slabe (na primer, jako kratke lozinke ili reči koje se mogu naći u rečniku), a nakon toga se mogu i upozoriti.

Ukoliko se sledeća pravila ispoštuju lozinke će dobiti svoj zaštitni smisao:

- svaka lozinka mora biti duga bar šest karaktera (poželjno osam), formirana od velikih i malih slova, uključujući najmanje jedan numerički karakter,
- lozinke ne smeju biti trivijalne, odnosno formirane na osnovu korisničkog imena, imena devojke, firme i slično (na primer, username: jsmith, passwd: smithj),
- za sve lozinke mora biti definisan rok važenja, odnosno period nakon koga se lozinka mora zameniti novom.

root korisnički nalog

Korisnički nalog root je deo sistema i kao takav nema sigurnosne restrikcije, odnosno ima najveće privilegije. To znači da sistem podrazumeva da administrator koji koristi taj nalog zna šta radi i da neće ispitivati njegove postupke. Superuser može, zahvaljujući grešci u pisanju, da obriše kritične sistemske datoteke, tako da je jako bitno da svi koji taj nalog koriste budu jako pažljivi. Preporučuje se da se administrator prijavi na sistem kao root samo ako za to postoji izričita potreba (na primer, kreiranje većeg broja korisnika ili administracija mrežnih servisa). Root nalog ne sme biti dostupan regularnim korisnicima sistema - na primer, administrator se ne sme prijaviti na računar kao root i napustiti prostoriju.

Promenljiva TMOUT

Kako root nalog ne bi bio dostupan običnim korisnicima, preporučuje se postavljanje time-out parametra (vremena neaktivnosti nakon kog će sistem automatski odjaviti superusera). To se najlakše postiže postavljanjem vrednosti promenljive TMOUT na broj sekundi neaktivnosti u datoteci /etc/profile:

```
TMOUT=300
```

Nakon toga superuser se mora odjaviti sa sistema i ponovo prijaviti na sistem da bi se promene načinjene u datoteci /etc/profile primenile na root nalog. Ove promene, odnosno time-out period, važiće i za sve ostale korisnike sistema. Administrator dodatno može postaviti login time-out period za grupu regularnih korisnika sistema navodeći promenljivu TMOUT u datotekama .bashrc u home direktorijumima tih korisnika.

Datoteka /etc/securetty

Datoteka /etc/securetty (secure getty) omogućava administratoru sistema da odredi terminale sa kojih se na sistem može prijaviti superuser. Datoteku /etc/securetty čita login proces i na osnovu nje utvrđuje da li se superuser može prijaviti na sistem sa tog terminala. Format datoteke je jednostavan: u svakoj liniji datoteke navodi se po jedan terminal ili virtuelna konzola sa koje se superuser može prijaviti. Iz ove datoteke je potrebno ukloniti sve nepotrebne terminale i virtuelne konzole (ili ih pretvoriti u komentare). Na primer, može se dozvoliti prijavljivanje superusera samo sa terminala tty1 i tty2:

```
# vc/1
# vc/2
# vc/3
...
# vc/11
tty1
tty2
# tty3
...
# tty11
# tty12
```

Ukoliko postoji potreba da se root prijavi na više terminala administrator se najpre može prijaviti kao regularan korisnik, a zatim iskoristiti komandu su da se prijavi kao root.

Sistemski korisnički nalozi

U cilju povećanja opšte sigurnosti sistema potrebno je ukloniti sve nepotrebne sistemske korisničke naloge. Ovi nalozi služe za specijalne namene, a isporučuju se kao deo Linux distribucije. Većina njih postoji na sistemu čak i ako servisi za koje su ti nalozi namenjeni nisu instalirani. Nakon svake nadogradnje ili instaliranja novog softverskog paketa potrebno je proveriti datoteku /etc/passwd i u njoj locirati nove sistemske korisnike. Sistemski korisnici koji su vezani za servise koji nisu instalirani mogu se ukloniti komandom userdel.

```
# userdel adm
# userdel lp
# userdel shutdown
# userdel halt
# userdel news
# userdel mail
# userdel uucp
# userdel operator
# userdel games
# userdel gopher
# userdel ftp
```

Napomena: specijalni korisnički nalozi mogu se obrisati komandom userdel username. Nalozi regularnih korisnika brišu se komandom userdel -r username, koja uklanja i home direktorijum korisnika.

Nakon uklanjanja nepotrebnih sistemskih korisnika, potrebno je ukloniti i prateće grupe:

```
# groupdel adm
# groupdel lp
# groupdel news
# groupdel mail
# groupdel uucp
# groupdel games
# groupdel dip
```

Napomena: korisnički nalog mail i grupa mail se ne smeju obrisati ukoliko se Sendmail koristi kao mail server. U slučaju da se kao alternativni mail server koristi qmail, ovi nalozi se mogu obrisati.

Nakon toga se mogu kreirati regularni korisnici i grupe i podesiti članstvo korisnika u grupama.

U slučaju da se radi o mrežnom serveru root će kreirati jednog ili dva korisnika (na primer, admin i operator). Kao dodatna mera zaštite mrežnog servera, preporučuje se postavljanje immutable flega za datoteke /etc/passwd, /etc/shadow i /etc/group. To se može izvesti i ne predstavlja opterećenje u daljem radu, jer se na serverima korisnici ne kreiraju često, tako da se ove datoteke smatraju relativno nepromenljivim. Ukoliko postoji potreba za dodavanjem novog korisnika potrebno je najpre ukinuti immutable fleg sa ovih datoteka.

Jednokorisnički režim rada

Na Linux sistemima postoji specijalan komandni režim rada poznatiji pod imenom jednokorisnički režim. Računar se može uvesti u jednokorisnički režim pomoću posebnih parametara koji se navode prilikom podizanja operativnog sistema. Konkretno, potrebno je programu LILO zadati sledeću komandu:

```
LILO: linux single
```

gde je linux ime (labela) slike kernela. Nakon toga sistem će biti doveden u nivo izvršenja 1 (runlevel 1), a na sistem prijavljen superuser. U ovom slučaju od korisnika se ne zahteva da unese root lozinku, što predstavlja potencijalnu opasnost.

Dodavanjem sledeće linije u datoteku /etc/inittab se od administratora eksplicitno zahteva da unese root lozinku prilikom podizanja sistema u jednokorisničkom režimu:

```
# What to do in single-user mode.
~~:S:wait:/sbin/sulogin
```

To znači da će proces init u jednokorisničkom režimu pokrenuti program sulogin pre nego što pokrene root shell.

Da bi promene postale validne potrebno je zadati sledeću komandu:

```
# /sbin/init q
```

Nakon ove komande proces init će ponovo pročitati konfiguracionu datoteku /etc/inittab.

Zabrana korišćenja Ctrl-Alt-Del

Zaustavljanje sistema kombinacijom tastera Ctrl-Alt-Del može se najlakše zabraniti pretvaranjem sledeće linije u datoteci /etc/inittab u komentar:

```
# ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

Nakon toga je potrebno zadati komandu:

```
# /sbin/init q
```

Zabrana korišćenja ove kombinacije tastera je jako značajna ukoliko sistem nije adekvatno fizički obezbeđen.

Datoteka /etc/fstab

Aktivno UNIX stablo može biti sačinjeno od nekoliko sistema datoteka (na primer root, /home, /tmp) koji su montirani na odgovarajuće mount-point direktorijume. Ovi sistemi datoteka su opisani u datoteci /etc/fstab. Datoteka /etc/fstab sadrži informacije o sistemima datoteka, njihovim podrazumevanim mount-point direktorijumima i podrazumevanim opcijama koje se koriste pri aktiviranju. Opcije koje se mogu navesti u datoteci /etc/fstab, a koje se tiču sigurnosti su:

defaults	na sistemu datoteka je dozvoljeno sve (kvote, upis i suid)
noquota	zabranjuje se primena kvota
nosuid	SUID i SGID bitovi postavljeni na datotekama neće imati efekta
nodev	blok i karakter specijalne datoteke se ne interpretiraju kao uređaji
noexec	zabranjuje se izvršavanje izvršnih datoteka (datoteka sa x flegom)
quota	dozvoljava se primena kvota
ro	dozvoljava pristup sistemu datoteka isključivo u režimu čitanja
rw	dozvoljava pristup sistemu datoteka u režimu za čitanje i upis
suid	SUID i SGID bitovi postavljeni na datotekama imaće efekta

Napomena: kod opcija nosuid, nodev, noexec i suid pod datotekama se podrazumevaju sve datoteke koje se nalaze na aktiviranom sistemu datoteka za koji je ta opcija navedena. Važenje ostalih opcija je limitirano samo na one sisteme datoteka za koje su te opcije navedene.

Pomoću ovih opcija lako se može povećati sigurnost aktiviranih sistema datoteka. Na primer: direktorijum /boot je mesto na kome se čuva kernel. Na velikom broju Linux sistema ovaj direktorijum se nalazi na posebnom sistemu datoteka, koji se podrazumevano aktivira u režimu čitanja i pisanja. Iz razloga sigurnosti ovo treba promeniti, odnosno dozvoliti aktiviranje /boot sistema datoteka isključivo u režimu čitanja. Najpre je potrebno u datoteci /etc/fstab načiniti sledeću izmenu:

```
/dev/hda2 /boot ext2 defaults,ro 1 2
```

Nakon toga se ovaj sistem datoteka mora ponovo aktivirati kako bi promene postale validne:

```
# mount /dev/hda2 –oremount
```

Administrator može proveriti da li je boot sistem datoteka sada aktiviran u režimu čitanja komandom cat /proc/mounts koja prikazuje sve aktivne sisteme datoteka i opcije sa kojima su aktivirani:

```
# cat /proc/mounts
/dev/hda1 / ext2 rw 0 0
/proc /proc proc rw 0 0
/dev/hda2 /boot ext2 ro 0 0
/dev/hdb1 /home ext2 rw,nodev 0 0
/dev/hdb2 /tmp ext2 rw,nodev,noquota 0 0
none /dev/pts devpts rw 0 0
```

Napomena: prilikom buduće nadogradnje Linux kernela, administrator mora privremeno ukinuti read-only atribut sistema datoteka.

Uklanjanje nepotrebnog softvera

Nakon instaliranja svih potrebnih softverskih paketa, paket menadžeri (poput programa RPM) nisu potrebni i mogu se skloniti u neki direktorijum (na primer /root) čiji sadržaj niko osim superusera ne može pročitati. Dodatno, samim izvršnim datotekama treba promeniti prava pristupa na 700, čime se dozvoljava korišćenje tog programa jedino superuseru. Alternativna metoda je kopiranje tih programa na disketu i uklanjanje sa diska. Time se onemogućava dalja instalacija programskih paketa, što predstavlja dodatni korak u povećanju sigurnosti.

Napomena: pod pomeranjem paket menadžera podrazumeva se pomeranje izvršnih datoteka, a ne celih paketa. Ukoliko se ceo paket ukloni sa sistema, administrator će morati opet da ga instalira pre nego što instalira bilo koji drugi softver.

U ostali softver koji se može ukloniti sa mrežnog servera ubrajaju se i programski prevodioci (na primer, prevodilac za jezik C i asembler), kao i svi mrežni servisi i programi koji se koriste za administraciju mreže a na tom serveru nisu neophodni. Na primer, ukoliko server ima funkciju Web servera, servisi ftpd, telnetd, routed i njima slični nisu potrebni i mogu se ukloniti.

Automatsko brisanje .bash_history datoteke

U cilju omogućavanja komfornijeg rada, shell čuva u datoteci ~/.bash_history određeni broj komandi koje je korisnik zadnje zadao. Svaki korisnik sistema imaće datoteku .bash_history u svom home direktorijumu, a broj komandi koje se čuvaju u toj datoteci se može odrediti dodavanjem linije HISTSIZE=n u datoteku /etc/profile. Preporučuje se da taj broj bude reda veličine 10 komandi. U cilju povećanja sigurnosti, potrebno je u datoteku /etc/profile upisati sledeću liniju:

```
HISTFILESIZE=0
```

To znači da će se sadržaj datoteke .bash_history obrisati svaki put kada se korisnik odjavi sa sistema.

Sigurnost skriptova u /etc/init.d direktorijumu

U /etc/init.d direktorijumu nalaze se skriptovi koji se prilikom podizanja operativnog sistema koriste za zaustavljanje i pokretanje procesa. Pristupna prava ovih skriptova nisu dovoljno restriktivna, tako da ih root mora promeniti, odnosno dozvoliti samo sebi pristup skriptovima u /etc/init.d direktorijumu:

```
# chmod -R 700 /etc/init.d/*
```

SUID bit

Regularan korisnik može pokrenuti program kao root ukoliko je postavljen bit SUID izvršne datoteke čiji je vlasnik root. Svi programi čija pristupna prava za vlasnika (rwsr-xr-x), odnosno grupu (rwxr-sr-s) uključuju pravo s (na mestu prava izvršavanja) imaju postavljen SUID ili SGID bit. Takvi programi regularnom korisniku daju privilegije superusera. Zato je potrebno SUID i SGID bit ukloniti sa svih programa na kojima on nije apsolutno neophodan.

Programi sa postavljenim SUID ili SGID bitom se mogu pronaći sledećom komandom:

```
# find / -type f \( -perm -04000 -o -perm -02000 \) -exec ls -l {}
\;
```

Nakon toga se bit s lako može ukloniti sa programa kojima nije neophodan (funkcija programa može se lako odrediti komandom man program name, nakon čega se izvodi zaključak). Na primer, komande mount i umount ne zahtevaju s bit, tako da se on može ukloniti:

```
# chmod a-s /bin/mount
# chmod a-s /bin/umount
```

Datoteka /etc/services

U datoteci /etc/services opisani su mrežni servisi i portovi na kojima ti servisi pružaju usluge klijentima. Ova datoteka omogućava procesima da pretvore ime servisa u adekvatan broj porta. Samo root može modifikovati sadržaj ove datoteke. Sadržaj ove datoteke najčešće ne treba menjati jer instalacioni program obezbeđuje datoteku /etc/services u kojoj su već upareni standardni servisi sa odgovarajućim portovima. Kao dodatna mera zaštite predlaže se postavljanje i flega (immutable) za tu datoteku.

Datoteka /etc/exports

Ukoliko se radi o NFS serveru datoteka /etc/exports mora biti konfigurisana sa najrestriktivnijim dozvolama, što podrazumeva sledeće:

- džoker znaci se ne smeju koristiti,

- superuser klijent računara ne sme naslediti pravo upisa na NFS od superusera servera,
- aktiviranje NFS sistema datoteka u režimu čitanja kad god je to moguće.

U nastavku teksta dat je jednostavan primer realizacije ovih mera. Pretpostavljamo sledeći sadržaj datoteke /etc/exports:

```
/share/project
```

Ovako konfigurisan, NFS dozvoljava pristup sa svih računara direktorijumu /share/project. Pretpostavljamo da tim direktorijumima treba dozvoliti pristup isključivo u režimu čitanja sa računara ws1 i ws2 iz istog domena. U datoteci /etc/exports treba izvršiti sledeće izmene:

```
/share/project ws1.mydomain.com (ro,root_squash)
/share/project ws2.mydomain.com (ro,root_squash)
```

Ovako konfigurisan, NFS dozvoljava pristup direktorijumu /share/project sa računara ws1 i ws2 iz istog domena. NFS se sa tih računara može aktivirati isključivo u režimu čitanja (opcija ro), a root korisniku se zabranjuje upis u taj direktorijum (root_squash). Da bi promene postale validne superuser mora zadati sledeću komandu:

```
# /usr/sbin/exportfs -a
```

Datoteke bez vlasnika

Datoteke koje nemaju vlasnika ili nisu dodeljene ni jednoj grupi obično ukazuju na to da je napadač uspeo da uđe u sistem. Ove datoteke mogu se lako pronaći pomoću sledeće komande:

```
# find / -nouser -o -nogroup
/usr/share/doc/apache/manual/programs/suexec.html.html
/usr/share/doc/apache/manual/programs/suexec.html.en
/usr/share/doc/apache/manual/programs/suexec.html.ja.jis
# ls -l /usr/share/doc/apache/manual/programs/suexec.html.html
-rw-r--r--   1   847   800   1659   jun 18 2002   suexec.html.en
```

Nakon pretrage potrebno je pregledati sadržaj ovih datoteka. Ukoliko je sadržaj datoteke normalan (na primer, tekstualna datoteka ili slika), dalje joj se dodeljuje vlasnik i grupa. Na primer:

```
# cd /usr/share/doc/apache/manual/programs
# less suexec.html.html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
...
# chown root suexec.html.html
# chgrp root suexec.html.html
```

Napomena: datoteke bez vlasnika mogu se pojaviti kao rezultat deinstalacije nekog programskog paketa. Takođe, datoteke u direktorijumu /dev mogu ostati bez vlasnika nakon brisanja nekih sistemskih grupa.

Datoteke .rhosts

Datoteke ~/.rhosts obezbeđuju mehanizam udaljene autentifikacije za rlogin, lpd i rsh procese. Svaka .rhosts datoteka specificira računare i korisnike kojima se veruje. Tim korisnicima je dozvoljen pristup sistemu bez navođenja lozinke. Kao takve, .rhosts datoteke predstavljaju ozbiljan sigurnosni problem svakog servera. Uklanjanje datoteka .rhosts čiji je sadržaj neadekvatan je važan postupak - pretraga za .rhosts datotekama spada u regularne periodične administrativne poslove i treba se automatizovati. Na primer, u direktorijumu /etc/cron.daily može se kreirati skript datoteka rhosts_audit sa sledećim sadržajem:

```
find /home –name .rhosts | mail –s "rhosts file audit report" root
```

Zatim skript teba učiniti izvršnim i dodeliti mu vlasnika i grupu root:

```
# chmod 755 /etc/cron.daily/rhosts_audit
# chown root /etc/cron.daily/rhosts_audit
# chgrp root /etc/cron.daily/rhosts_audit
```

Nakon toga, root će dobijati jednom dnevno mail sa izveštajem o pronađenim rhosts datotekama. Superuser nakon toga može proveriti sadržaj i po potrebi obrisati te datoteke.

Nakon instalacije Linux sistema, ovih datoteka obično nema u aktivnom stablu.

PAM (Pluggable Authentication Modules)

PAM (izmenljivi autentifikacioni moduli) omogućavaju razdvajanje autentifikacije od samih programa. To se vrši pomoću biblioteke funkcija koje aplikacije koriste kada zahtevaju autentifikaciju korisnika. Ssh, pop i imap su primeri PAM-aware programa, odnosno programa koji koriste PAM module. Konfiguracione datoteke PAM modula, odnosno datoteke koje koriste PAM-aware programi (na primer ssh, samba, passwd) se nalaze u direktorijumu /etc/pam.d. U slučaju da program nema svoju PAM konfiguracionu datoteku koriste se podrazumevane vrednosti. Sami moduli su smešteni u direktorijum /lib/security. Korišćenjem PAM modula, opšta sigurnost Linux sistema može se znatno povećati.

GNU Privacy Guard (GnuPG)

Administracija i upotreba programa GNU Privacy Guard.

Prema zvaničnoj README datoteci, GnuPG je alat koji obezbeđuje sigurnu komunikaciju i skladištenje podataka. GnuPG se može upotrebiti za šifrovanje podataka i kreiranje digitalnog potpisa. U GnuPG su uključene tehnike za lakše upravljanje ključevima. Paket je prilagođen sa predloženim OpenPGP Internet standardom, opisanim u RFC2440.

Algoritmi koje GnuPG koristi su:

- simetrični kriptoalgoritmi: 3DES, CAST5, BLOWFISH, RIJNDAEL, RIJNDAEL192, RIJNDAEL256, TWOFISH
- algoritmi sa javnim ključem: RSA, RSA-E, RSA-S, ELG-E, DSA, ELG
- hash funkcije: MD5, SHA1, RIPEMD160

GnuPG se može koristiti za:

- šifrovanje podataka,
- kreiranje digitalnih potpisa,
- proveru integriteta izvornog programa.

Pretpostavimo da administrator sistema treba da obezbedi siguran server. Jedna stvar koju je neophodno uraditi je zaštita servera mrežnom barijerom. Takođe, potrebno je primeniti neke opšte sigurnosne mere. Međutim, na svakom serveru se u određenom momentu instalira neki softverski paket. Ukoliko se ne izvrši provera verodostojnosti paketa, moguće je da u modifikovani paket bude ugrađen neki program za praćenje ili ostvarivanje nelegalnog pristupa. GnuPG nudi administratoru mogućnost da se uveri da je paket verodostojan, da dolazi od proverenog izvora i da nije modifikovan. Sa GnuPG alatkom administrator može verifikovati digitalni potpis programskog paketa i biti siguran da je program originalan. Zato se preporučuje da se ovaj besplatan programski paket instalira pre bilo kog drugog programskog paketa.

Šifrovanjem podataka obezbeđuje se visoki stepen pouzdanosti podataka. Programski paket GnuPG nudi korisnicima mogućnost šifrovanja raznih vrsta podataka uključujući elektronsku poštu (e-mail poruke) i datoteke. Time se proširuje primena ovog paketa na radne stanice: korisnici mogu šifrovati svoje datoteke i komunicirati sa ostatkom sveta na siguran način.

Uvod u tehnologiju šifrovanja

Šifrovanje obuhvata matematičke postupke modifikacije podataka takve da šifrovane podatke (šifrat) mogu pročitati samo korisnici sa odgovarajućim ključem. Proces šifrovanja matematičkom funkcijom uz određeni ključ transformiše otvoren tekst (plain text), odnosno originalnu poruku ili datoteku, u zaštićen, šifrovan tekst (ciphertext - šifrat). U osnovi svih kriptoalgoritama leže matematički postupci konfuzije (odnosno supstitucije - zamene karaktera) i difuzije (odnosno permutacije - premeštanja). Supstitucija obuhvata zamenu delova originalne poruke (pojedinačno ili u grupama konstantne dužine) drugim slovima ili kombinacijom tih slova sa ključem. Permutacijom se originalna poruka preuređuje po nekom algoritmu. Kombinovanjem ovih procesa ostvaruje se visok stepen zaštite podataka. Transformacije šifrata u otvoreni tekst (dešifrovanje) je obrnut proces - matematičkom funkcijom uz određeni ključ šifrovani podaci se transformišu u originalnu poruku ili datoteku. Šifrovani podaci su kao takvi zaštićeni od neautorizovanog pristupa (odnosno od svih korisnika koji nemaju odgovarajući ključ) i kao takvi se mogu preneti preko nesigurnog kanala ili čuvati na disku koji nije obezbeđen od neautorizovanog pristupa.

Osnovna aksioma teorije kriptografije (Applied Cryptography, Bruce Schneier) glasi: snaga kriptoalgoritma ne sme se zasnivati na tajnosti samog algoritma, već isključivo na tajnosti ključa.

Simetrični algoritmi i algoritmi sa javnim ključem

Kriptoalgoritmi se dele na simetrične (isti ključ se koristi i za šifrovanje i za dešifrovanje podataka) i algoritme sa javnim ključem (podaci se šifruju javnim ključem a dešifruju privatnim).

Funkcija šifrovanja simetričnim kriptoalgoritmom E na osnovu ključa k i ulaznih podataka p kreira šifrat c. Funkcija dešifrovanja D na osnovu istog ključa k i šifrata c kreira originalnu poruku p:

E(p,k)=c	šifrovanje
D(c,k)=p	dešifrovanje

Simetrični algoritmi su brzi i kao takvi se mogu koristiti za šifrovanje većih datoteka ili implementaciju u kripto sisteme datoteka. Najpoznatiji simetrični kriptoalgoritmi su DES (Data Encryption Standard), AES (Advanced Encryption Standard), IDEA, Blowfish i drugi.

Funkcija šifrovanja kriptoalgoritmom sa javnim ključem E na osnovu javnog ključa k1 i ulaznih podataka p kreira šifrat c. Funkcija dešifrovanja D na osnovu privatnog ključa k2 i šifrata c kreira originalnu poruku p:

E(p,k1)=c	šifrovanje javnim ključem
D(c,k2)=p	dešifrovanje privatnim ključem

Javni ključ je poznat onim osobama sa kojima korisnik želi da komunicira, dok je tajni ključ poznat samo korisniku koji ga je kreirao. Najpoznatiji algoritam sa javnim ključem je RSA (RSA Data Security - Ronald Rivest, Adi Shamir i Leonard Adelman). Asimetrični algoritmi su sporiji i primenjuju se za kreiranje digitalnih potpisa i šifrovanje ključeva simetričnih algoritama kojima su šifrovane datoteke.

Digitalni potpis

Digitalni potpis (digital signature) je elektronska verziju potpisa, na osnovu koga se može identifikovati pošiljalac i dokazati verodostojnost poruke. Prilikom kreiranja digitalnog potpisa ne šifruje se cela poruka, već samo hash vrednost. Hash funkcija na osnovu ulaznih podataka bilo koje dužine kreira niz tačno određene dužine (hash vrednost) koji na jedinstven način definiše ulazni podatak. Pri tome, na osnovu hash vrednosti ne mogu se odrediti originalni podaci - funkcija je strogo jednosmerna. Prilikom potpisivanja, najpre se jednosmernom hash funkcijom izračuna hash vrednost h poruke p, koja se nakon toga šifruje algoritmom sa javnim ključem:

H(p)=h1	kreiranje hash vrednosti
E(k1,h1)=s	potpisivanje (šifrovanje hash vredosti privatnim ključem)

Korisnik šalje originalnu poruku i digitalni potpis primaocu. Primalac na osnovu poruke određuje hash vrednost h2 i dešifruje primljeni potpis javnim ključem pošiljaoca. Upoređivanjem vrednosti h1 i h2 proverava se identitiet pošiljaoca:

H(p)=h2	kreiranje hash vrednosti
D(k2,s)=h1	provera primljenog potpisa (dešifovanje potpisa javnim ključem)
if h1=h2 then OK	poređenje hash vrednosti

Najčešće korišćena hash funkcija je MD5 (Message Digest).

Instaliranje GnuPG paketa

GnuPG RPM paket se distribuira uz većinu Linux sistema, ali ova verzija nije najnovija i preporučljivo je instalirati najnoviju dostupnu verziju za konkretan server i arhitekturu. Instalacija se obavlja pod root nalogom, a nakon instalacije ovog paketa nije potrebno ponovo prevoditi jezgro.

Poslednje verzije GnuPG paketa se mogu preuzeti sa sajta www.gnupg.org. Za instalaciju se preporučuje korišćenje RPM paketa. Ako se za instalaciju ne koristi RPM paket, u slučaju nekog budućeg ažuriranja biće teško naći sve instalacione datoteke na sistemu. Problem se može rešiti kreiranjem liste datoteka u sistemu pre i posle instaliranja GnuPG paketa. Upoređivanjem ovih listi komandom diff može se otkriti gde je koja datoteka smeštena.

```
# find /* > GnuPG.preinstall          # pre instalacije
# find /* > GnuPG.postinstall         # nakon instalacije
# diff GnuPG.preinstall GnuPG.postinstall > GnuPG.files
```

U daljem tekstu su koraci koji se moraju izvršiti pri konfigurisanju, kompajliranju i optimizovanju GnuPG programa pre samog instaliranja na Linux sistem. Administrator se na sistem prijavljuje kao superuser root.

Instalaciona arhiva se kopira u direktorijum /var/tmp (podrazumevani instalacioni direktorijum), nakon čega je treba raspakovati programom tar.

```
# cp gnupg-version.tar.gz /var/tmp/
# cd /var/tmp/
# tar xzpf gnupg-version.tar.gz
```

Nakon toga potrebno je proveriti autentičnost paketa kreiranjem MD5 hash vrednosti i proverom dobijene vrednosti sa onom koja se nalazi na sajtu http://www.gnupg.org/download.html (autentična vrednost):

```
# md5sum gnupg-version.tar.gz
```

Rezultat treba da bude sličan sledećem:

```
7c319a9e5e70ad9bc3bf0d7b5008a508 gnupg-version.tar.gz
```

Nakon toga se prelazi na novokreirani GnuPG direktorijum radi konfigurisanja i optimizacije.

```
# cd gnupg-1.0.6/
# CFLAGS="-O3 -march=i686 -mcpu=i686 -funroll-loops \
-fomit-frame-pointer" ./configure --prefix=/usr \
--mandir=/usr/share/man --infodir=/usr/share/info --disable-nls
```

Napomena: obratiti pažnju na vrednost dodeljenu promenljivoj CFLAGS. U primeru je upotrebom parametara -march=i686 i -mcpu=i686 izvršena optimizacija GnuPG paketa za bilo koju I686CPU arhitekturu.

Sada je potrebno napraviti listu datoteka u sistemu pre instaliranja programa (druga lista se kreira nakon instalacije, kada se one i upoređuju radi pronalaženja GnuPG datoteka). Nakon toga se GnuPG instalira na server:

```
# cd /var/tmp/gnupg-version/
# make check
# make install
# strip /usr/bin/gpg
# strip /usr/bin/gpgv
```

Komanda make install prevešće sve izvorne datoteke u izvršnu binarnu datoteku koju će zatim sa svim pratećim datotekama instalirati u odgovarajuće lokacije. Komanda make check će pokrenuti bilo koji automatizovani test koji dolazi sa paketom, i konačno, komanda strip će smanjiti veličinu gpg i gpgv binarnih datoteka da bi dobili najbolji rad ovih programa.

Nakon toga, mogu se obrisati tar arhive i direktorijum sa izvornim kodom.

Administracija GnuPG - rad sa ključevima

Pre svega, ukoliko se prvi put koristi GnuPG potrebno je kreirati novi par ključeva (javni i privatni) da bi se mogle koristiti mogućnosti šifrovanja. Pretpostavićemo da administrator kreira par ključeva za root nalog. To kasnije mogu uraditi i ostali korisnici sistema za svoje naloge.

Kreiranje para ključeva

Za kreiranje novog para ključeva, koriste se sledeće komande:

```
# gpg --gen-key
gpg: /root/.gnupg: directory created
gpg: /root/.gnupg/options: new options file created
gpg: you have to start GnuPG again, so it can read the new options
file
# gpg --gen-key
gpg: /root/.gnupg/secring.gpg: keyring created
gpg: /root/.gnupg/pubring.gpg: keyring created
```

Program gpg dalje postavlja nekoliko pitanja korisniku. Prvo pitanje se odnosi na tip ključeva.

```
Please select what kind of key you want:
(1) DSA and ElGamal (default)
```

```
(2) DSA (sign only)
(4) ElGamal (sign and encrypt)
Your selection? 1
DSA keypair will have 1024 bits.
About to generate a new ELG-E keypair.
```

Nakon toga program zahteva od korisnika da odabere veličinu ključeva (podrazumevana vrednost je 1024 bita):

```
minimum keysize is 768 bits
default keysize is 1024 bits
highest suggested keysize is 2048 bits
What keysize do you want? (1024) 1024
Requested keysize is 1024 bits
```

Korisnik zatim određuje koliko će dugo ključevi biti validni. Podrazumeva se da ključevi nemaju rok nakon koga više nisu validni:

```
Please specify how long the key should be valid.
0 = key does not expire
<n> = key expires in n days
<n>w = key expires in n weeks
<n>m = key expires in n months
<n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct (y/n)? y
```

Zatim je potrebno dati neke osnovne parametre o vlasniku ključeva:

```
You need a User-ID to identify your key; the software constructs
the user id from Real Name, Comment and Email Address in this form:
"Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"
Real name: John Smith
Email address: jsmith@sec.org
Comment: Johnny
You selected this USER-ID:
"John Smith (Johnny) <jsmith@sec.com>"
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
```

Nakon toga se programu daje šifra (passphrase) kojom se štiti par ključeva: Prilikom generisanja ključeva generator pseudoslučajnih brojeva oslanja se na razne akcije koje se dešavaju u pozadini sistema (procesorski šum, pomeranje miša, aktivnosti koje nastaju kao posledica keširanja diskova). Poželjno je radi postizanja slučajnije vrednosti da korisnik izvršava neku pozadinsku aktivnost (na primer, da kopira neku datoteku ili pomera miša).

```
You need a Passphrase to protect your secret key.
Enter passphrase: johnnyboy
Repeat passphrase: johnnyboy
We need to generate a lot of random bytes. It is a good idea to
perform some other action (type on the keyboard, move the mouse,
utilize the disks) during the prime generation; this gives the
random number generator a better chance to gain enough entropy.
...
public and secret key created and signed.
```

Novi par ključeva (privatni i javni ključ) je nakon toga kreiran i smešten u home direktorijum superusera (/root).

Izvoz ključeva

Par ključeva se može izvesti (export) nakon čega se javni ključ može distribuirati korisnicima sa kojima je potrebno ostvariti šifrovani komunikacioni kanal širom sveta. Takođe, javni ključ se može objaviti na nekoj web stranici, ali će u tom slučaju biti svima dostupan. GnuPG nudi korisnicima nekoliko korisnih opcija za objavljivanje javnih ključeva.

Za snimanje javnog ključa u ASCII zaštićenom formatu koristi se sledeća komanda:

```
# gpg --export -ao UID
```

Na primer, prethodno generisani ključ (koji pripada korisniku John Smith) se može izvesti u ASCII format sledećom komandom:

```
# gpg --export -ao John Smith
```

Opcije imaju sledeće značenje:

--export	izvlačenje javnog ključa iz objavljene šifrovane datoteke
-a	kreiranje ASCII zaštićene datoteke koja se može poslati poštom, objaviti ili staviti na web stranicu
-o	snimanje rezultata u datoteku
UID	korisnički ključ koji treba biti izvučen

Uvoz ključeva

Javni ključ koji korisnik dobije od osobe s kojom želi da komunicira preko šifrovanog kanala mora se dodati u bazu ključeva. Nakon toga, taj ključ se može koristiti za šifrovanje podataka i verifikaciju potpisa.

Za uvoz javnih ključeva u bazu ključeva koristi se sledeća komanda:

```
# gpg --import filename
```

Komanda će uvesti novi ključ u bazu samo ako je potpisan.

Označavanje ključeva

Nakon uvoza ključeva u bazu javnih ključeva, korisnik može potpisati javni ključ svojim privatnim ključem. Potpisivanjem se postiže dodatni nivo sigurnosti: tada se zna da je pomenuti korisnik stvarni vlasnik ključa.

Za označavanje ključa kog smo prethodno dodali u našu bazu, koristimo sledeću komandu:

```
# gpg --sign-key UID
```

Napomena: ključ treba potpisati samo ukoliko smo sigurni da je ključ zaista autentičan. Ključ se ne potpisuje na osnovu pretpostavke!

Provera potpisa

Kada je javni ključ korisnika jednom izvučen i izvezen, svako ko dobije taj javni ključ biće u mogućnosti da:

- šalje korisniku šifrovane podatke,
- proveri potpis tog korisnika.

Za proveru potpisa šifrovanih podataka koristi se sledeća komanda:

```
# gpg --verify Data
```

gde je Data šifrovani podatak (datoteka) čiji potpis proveravamo.

Šifrovanje i dešifrovanje

Nakon instaliranja i konfigurisanja programa, kreiranja i uvoženja ključeva mogu se obaviti postupci šifrovanja i dešifrovanja datoteka.

Za šifrovanje javnim ključem korisnika UID (čiji je javni ključ prethodno unesen u bazu) i potpisivanje šifrata svojim privatnim ključem, koristi se sledeća komanda:

```
# gpg -sear USER file
```

Od korisnika se zahteva da unese passphrase svog privatnog ključa.

Navedeni argumenti imaju sledeće značenje:

s	potpisivanje (sign) - potpisivanjem šifrata privatnim ključem izbegava se rizik da se neko predstavi kao korisnik koji je datoteku šifrovao
e	šifrovanje (encrypt)
a	kreiranje ASCII zaštićenog izlaza (".asc" - spremno za slanje poštom)
r	šifrovanje javnim ključem korisnika koji je naveden kao USER

Za dešifrovanje podatka, koristi se sledeća komanda:

```
# gpg -d file
```

Argument d se odnosi na dešifrovanje, a file je ime datoteke koju treba dešifrovati. Veoma je važno da javni ključ pošiljaoca poruke koju želimo da dešifrujemo bude u našoj javnoj bazi - u suprotnom proces dešifrovanja nije moguć.

Linux kao mrežna barijera

Mrežna barijera (firewall). Filtriranje paketa i prevođenje IP adresa. Administracija iptables mrežne barijere.

Mrežnim barijerama (firewall) kreiraju se kontrolne tačke bezbednosti na granicama privatnih mreža. Na ovim kontrolnim tačkama mrežne barijere ispituju sve pakete koji prolaze između privatne mreže i Interneta. Jednostavno rečeno, firewall se nalazi na granici privatne mreže i filtrira saobraćaj na relaciji privatna mreža - Internet. Mrežne barijere održavaju Internet konekciju što je moguće bezbednijom tako što ispituju i nakon toga odobravaju ili odbijaju svaki pokušaj povezivanja privatnih mreža i spoljnih mreža kao što je Internet. Snažne mrežne barijere štite mrežu na svim slojevima (počev od sloja veze do aplikacionog sloja). Mrežne barijere omogućavaju centralizaciju svih bezbednosnih servisa na računarima koji su optimizovani i posvećeni zadatku zaštite.

Metode zaštite mrežnim barijerama

Mrežne barijere koriste tri osnovna metoda zaštite na mrežnom, transportnom i aplikacionom sloju.

Na mrežnom sloju firewall filtrira pakete, odnosno odbacuje TCP/IP pakete neautentifikovanih računara. Mrežna barijera dozvoljava jedino prolaz IP paketa sa ispravnom IP adresom iz spoljne mreže. Prilikom kontrole saobraćaja iz interne mreže ka ostatku sveta neke mrežne barijere vrše maskiranje IP adresa. Neke mrežne barijere prevode interne IP adrese u validne IP adrese (NAT - Network Address Translation), dok druge zamenjuju sve interne adrese sa adresom mrežne barijere, čime se sprečava adresiranje računara u internoj mreži. Maskiranjem IP adresa uklanja se mogućnost praćenja spolja.

Na transportnom sloju mrežna barijera dozvoljava ili zabranjuje pristup TCP/IP portovima zavisno od izvorišnih i odredišnih IP adresa. Na taj način se vrši kontrola pristupa TCP servisima.

Na aplikacionom sloju proksi serveri prihvataju zahteve za pristup određenoj aplikaciji koji dalje upućuju ka odredištu ili blokiraju. Proksi servisi uspostavljaju konekciju na visokom aplikacionom nivou, čime se prekida konekcija na mrežnom sloju između računara u privatnoj mreži i računara iz spoljne mreže.

Kao mrežna barijera može se koristiti skup uređaja i servera od kojih svaki obavlja samo jednu od navedenih funkcija. Na primer, ruter obavlja filtriranje paketa, dok se proksi server nalazi na posebnom računaru unutar mreže.

Mrežne barijere takođe pružaju servise šifrovane autentifikacije za pristup privatnoj mreži iz spoljnog sveta i virtuelnog privatnog umrežavanja (VPN), kojim se omogućava sigurno povezivanje privatnih mreža preko Interneta.

Filtriranje paketa

Filtriranje paketa je jedna od ključnih funkcija mrežnih barijera. Filtri analiziraju pakete i upoređuju ih sa pravilima regulisanim u bazi podataka, nakon čega prosleđuju samo one koji zadovoljavaju specificirani kriterijum.

Mrežne barijere koje filtriraju pakete najčešće odbacuju dolazeće zahteve za uspostavljanjem konekcije, odbacuju TCP pakete upućene portovima koji ne bi trebalo da budu raspoloživi za Internet i zabranjuju pristup mreži određenim opsezima IP adresa.Sofisticirani filtri odbacuju sve konekcije u kojima postoje znaci koji ukazuju na napad (navođenje tačne putanje puta - Source Routing, preusmeravanje ICMP paketa i lažiranje IP adresa).

Paketi se filtriraju po tipu, izvorišnoj adresi i informaciji o portu. Filter paketa može da odluči da:

- odbaci paket (DROP),
- prihvati paket (ACCEPT),
- odbije paket, nakon čega obaveštava pošiljaoca da njegov paket nije prihvaćen (REJECT).

Generalno, filtri se mogu konfigurisati tako da određivanje vrše na bilo kom delu zaglavlja paketa, dok većina filtara donosi odluku na osnovu:

- tipa protokola (na ovaj način se može izvršiti diskriminacija čitavih skupova protokola, kao što su UDP, TCP, ICMP, IGMP);
- IP adrese (specifično prihvatanje izvesnih adresa hostova je najjači oblik bezbednosti koji nudi filtriranje paketa);
- TCP/UDP porta (dobri filtri paketa dozvoljavaju administratoru da specificira opsege IP adresa koje mogu pristupiti određenom portu). Na taj način se može dozvoliti svim računarima da pristupe TCP portu 80 za HTTP , dok se pristup TCP portu 23 (Telnet) ili 22 (ssh) može ograničiti samo određenom opsegu IP adresa;
- broja fragmenta;
- informacije rutiranja sa izvora.

Prevođenje mrežnih adresa

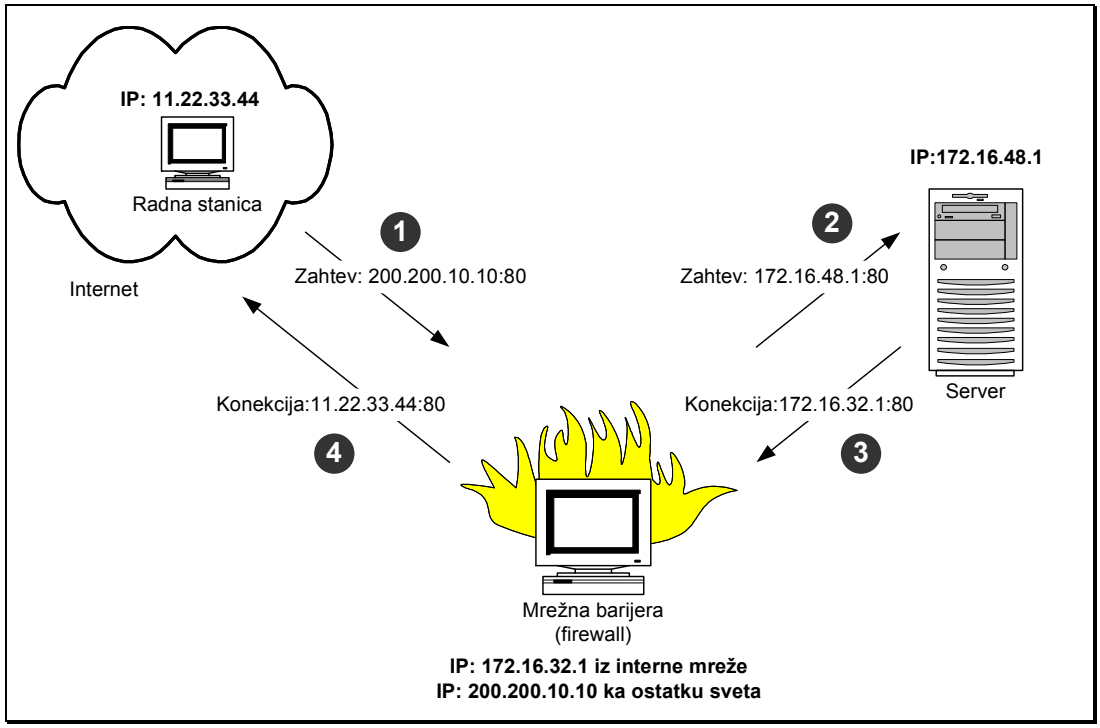
Prevođenje mrežnih adresa (Network Address Translation) je proces konverzije IP adresa privatne mreže u jedinstvenu IP adresu na Internetu. Iako je prevođenje mrežnih adresa inicijalno namenjeno radi veće dostupnosti IP adresa na Internetu, bezbednosni aspekt prevođenja adresa je veoma važan. NAT je implementiran samo na transportnom sloju referentnog modela. To znači da se informacija skrivena u delu za podatke može preneti do servisa viših slojeva i iskoristiti za eksploatisanje slabosti tih servisa. Da bi se sprečili bezbedonosni propusti servisa viših slojeva mora se koristiti servis višeg sloja kao što je proksi.

NAT efektivno skriva sve TCP/IP informacije o računarima u privatnoj mreži od napadača sa Interneta. NAT takođe dozvoljava da se unutar mreže koristi bilo koji opseg IP adresa, uključujući i one koje su već u upotrebi na Internetu. Prilikom prolaza paketa kroz mrežnu barijeru NAT skriva IP adrese računara iz privatne mreže konvertujući ih u adresu mrežne barijere (ili u adresu koja se odnosi na mrežnu barijeru). Mrežna barijera zatim ponovo šalje podatke koji se u tom paketu nalaze sa svoje adrese, koristeći pritom tablicu prevođenja adresa.

U opštem slučaju NAT tabele mapiraju:

- lokalnu IP adresu u globalnu IP adresu statički,
- lokalnu IP adresu u bilo koju adresu iz skupa dodeljenih globalnih IP adresa,
- lokalnu IP adresu sa posebnim TCP portom u bilo koju adresu iz skupa dodeljenih globalnih IP adresa,
- globalnu IP adresu u jednu iz skupa lokalnih IP adresa na osnovu Round Robin algoritma.

Primer sa slike 14.1 ilustruje funkcionisanje prevođenja adresa.



Slika 14.1 *Prevođenje IP adresa*

NAT funkcioniše na računaru čija je interna IP adresa 172.16.32.1, a adresa ka ostatku sveta 200.200.10.10. NAT prima zahtev za uspostavljanjem HTTP konekcije (odnosno konekcije na TCP portu 80) od računara 11.22.33.44. NAT će naći u svojim tabelama zapis u kome stoji da je port 80 preslikan (mapiran) na računar 172.16.48.1. NAT zamenjuje IP adresu 11.22.33.44:80 adresom 172.16.32.1:80 i prosleđuje paket HTTP

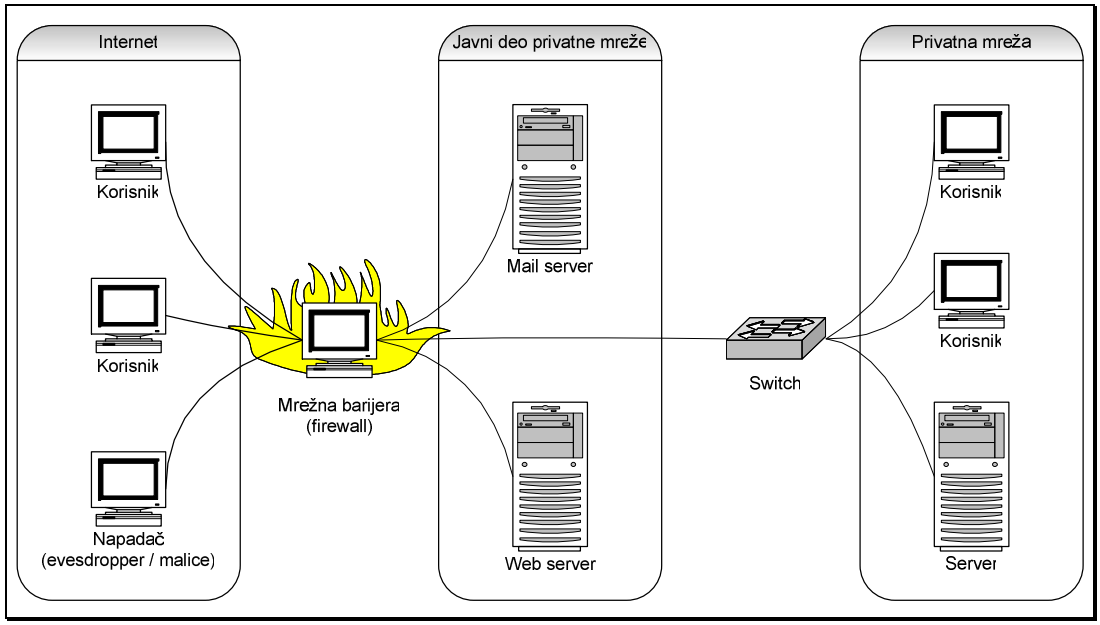
serveru. U povratnom nizu izvodi se prevođenje adrese unazad, tako da paket poslat sa adrese HTTP servera izgleda kao da je poslat sa javne adrese mrežne barijere.

Dvonivovska zaštita

Opasnost od napada sa Interneta može se znatno smanjiti korišćenjem dva nivoa zaštite. Osnovna ideja sastoji se u sledećem: jedan firewall štiti Web server od napada sa Interneta, ali dozvoljava pristup Internet servisima koje pruža taj deo mreže. Drugi firewall sa jačom bezbednosnom politikom ne dozvoljava pristup privatnoj mreži sa Interneta i skriva identitete računara iz privatne mreže.

Najveći broj firewall uređaja i softverskih paketa dozvoljava primenu različitih bezbednosnih politika na svakom interfejsu mrežne barijere. Na taj način jednom mrežnom barijerom sa tri interfejsa može se postići funkcionalnost dva firewall uređaja (slika 14.2).

Demilitarizovana zona (DMZ) je deo mreže koji ne pripada privatnoj mreži, a nije direktno povezan na Internet. Najčešće, ovo je područje između rutera za pristup Internetu i interne mreže, odnosno javni deo privatne mreže.



Slika 14.2 *Funkcionalnost dva firewall uređaja*

iptables

Filtriranje paketa

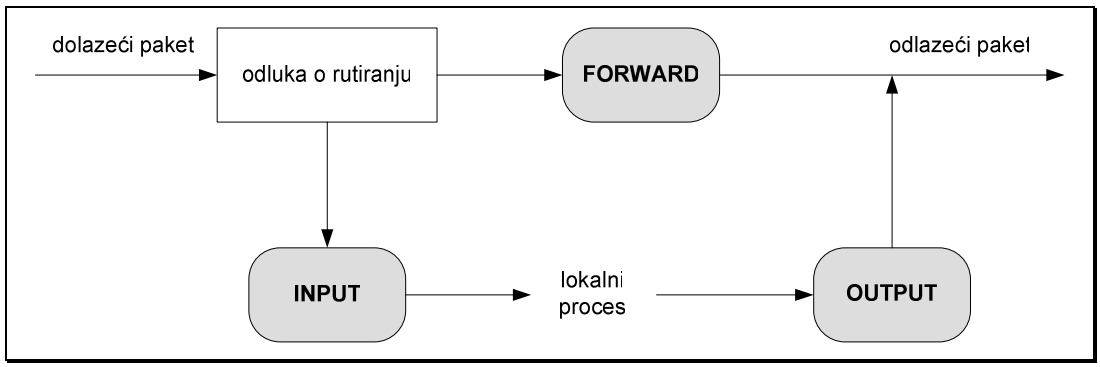
Netfilter koristi tri filtra (INPUT, OUTPUT i FORWARD), realizovanih u formi lanaca pravila (chains), koji su predstavljeni odgovarajućim krugovima na slici 14.3. Svaki lanac

sadrži pravila koja se primenjuju jedno za drugim na svaki paket koji prolazi kroz lanac. Kada paket stigne u lanac sa dijagrama, pravila se primenjuju na paket i na taj način se određuje dalja sudbina paketa. Ako paket ne zadovoljava jedno ili više pravila u lancu, paket se odbacuje (DROP). Ako paket zadovoljava sva pravila u lancu, paket se prihvata (ACCEPT) i može da nastavi svoj put kroz kernel.

Kada paket stigne kernel najpre analizira odredišnu adresu paketa. Ako kernel na osnovu te adrese odluči da je paket namenjen tom računaru, prosleđuje se INPUT lancu. Ako pravila u tom lancu dozvole da paket prođe, paket se prosleđuje odgovarajućem procesu koji ga očekuje.

Dolazeći paket koji nije namenjen lokalnim procesima predaje se FORWARD lancu. Ako kernel nije preveden s podrškom za prosleđivanje IP paketa (IP forwarding), ili kernel ne može da odredi kako da prosledi paket na osnovu tabele rutiranja, paket se odbacuje. Ako je prosleđivanje aktivirano i kernel može na osnovu tabele rutiranja da odredi preko kojeg mrežnog interfejsa paket treba da bude poslat, paket se prosleđuje FORWARD lancu. Ako je paket prihvaćen pravilima FORWARD lanca, prosleđuje se interfejsu prethodno određenom na osnovu tabele rutiranja.

Sudbinu mrežnih paketa koje je poslao program pokrenut na računaru određuju pravila OUTPUT lanca. Ako pravila lanca prihvate paket, paket se šalje na interfejs koji je određen tabelom rutiranja.



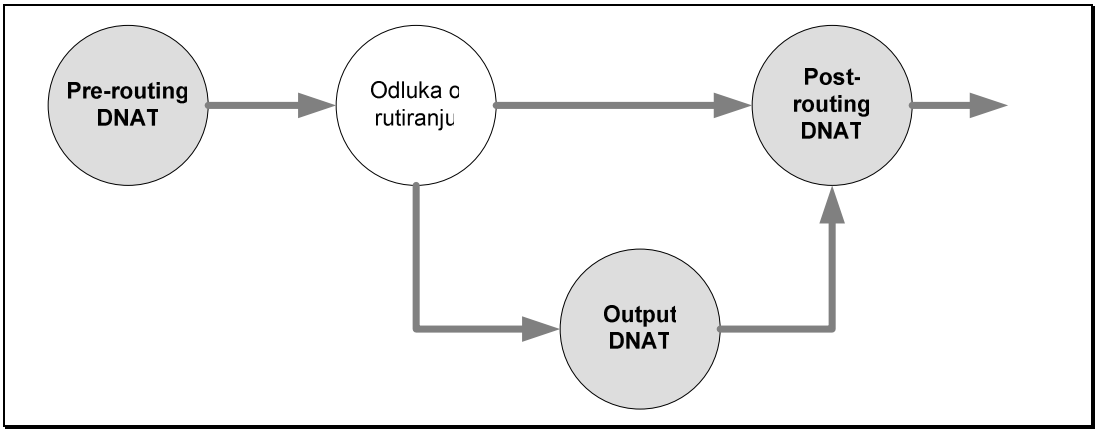
Slika 14.3 *Put paketa kroz lance*

NAT tabela

NAT tabele se koriste za prevođenje mrežne adrese (izvorišno ili odredišno polje) IP paketa u Linux kernelu 2.4. Source NAT se konsultuje prilikom prevođenja izvorišne adrese prvog paketa jedne konekcije, što znači da je početna tačka konekcije promenjena. Source NAT se izvodi nakon rutiranja, odnosno neposredno pre slanja paketa. Destination NAT se upotrebljava prilikom promene odredišne adrese prvog paketa jedne konekcije, odnosno prilikom usmeravanja konekcije ka drugom odredištu. DNAT se uvek izvodi pre rutiranja, odnosno odmah nakon primanja paketa. Koristi se najčešće za preusmeravanje pristupa mrežnoj barijeri sa javnom IP adresom na drugi računar. Maskiranje (IP masquerading) je poseban oblik SNAT-a sa većim overhead. Razlika je u tome što SNAT bira jednu konfigurisanu IP adresu, dok se prilikom maskiranja bira jedna od dozvoljenih IP adresa.

NAT tabela iptables mrežne barijere sastoji se od tri lanca pravila:

- prerouting (izmena paketa pre rutiranja),
- postrouting (izmena paketa nakon rutiranja),
- output (izmena paketa pre slanja).



Slika 14.4 *iptables NAT*

Pravila se primenjuju u serijama, sve dok jedan od lanaca pravila ne odgovara paketu. U svakoj tački (nakon primanja, pre rutiranja i pre slanja) proverava se kojoj konekciji paket pripada. Ako pripada novoj konekciji, odgovarajući lanac NAT tabele se konsultuje da bi se odredilo šta treba da se uradi sa paketom. To pravilo se automatski primenjuje na sve sledeće pakete koji pripadaju toj konekciji.

Mangle tabela

Mangle tabelom menjaju se svojstva paketa koja nemaju direktan uticaj na pakete. Kao i NAT, mangle ima ista tri lanca pomoću kojih se mogu postaviti svojstva paketa i pomoću kojih se paketi mogu obeležiti (markirati).

Mangle tabele mogu se koristiti za promenu sledećih polja IP paketa:

TOS — TOS polje se koristi za podešavanje ili promenu polja Type Of Service (tip servisa) u paketu. Ovo polje ne treba postavljati za pakete koji su namenjeni Internetu, osim ako se nakon toga ne donese odluka o rutiranju (na primer, programom iproute2);

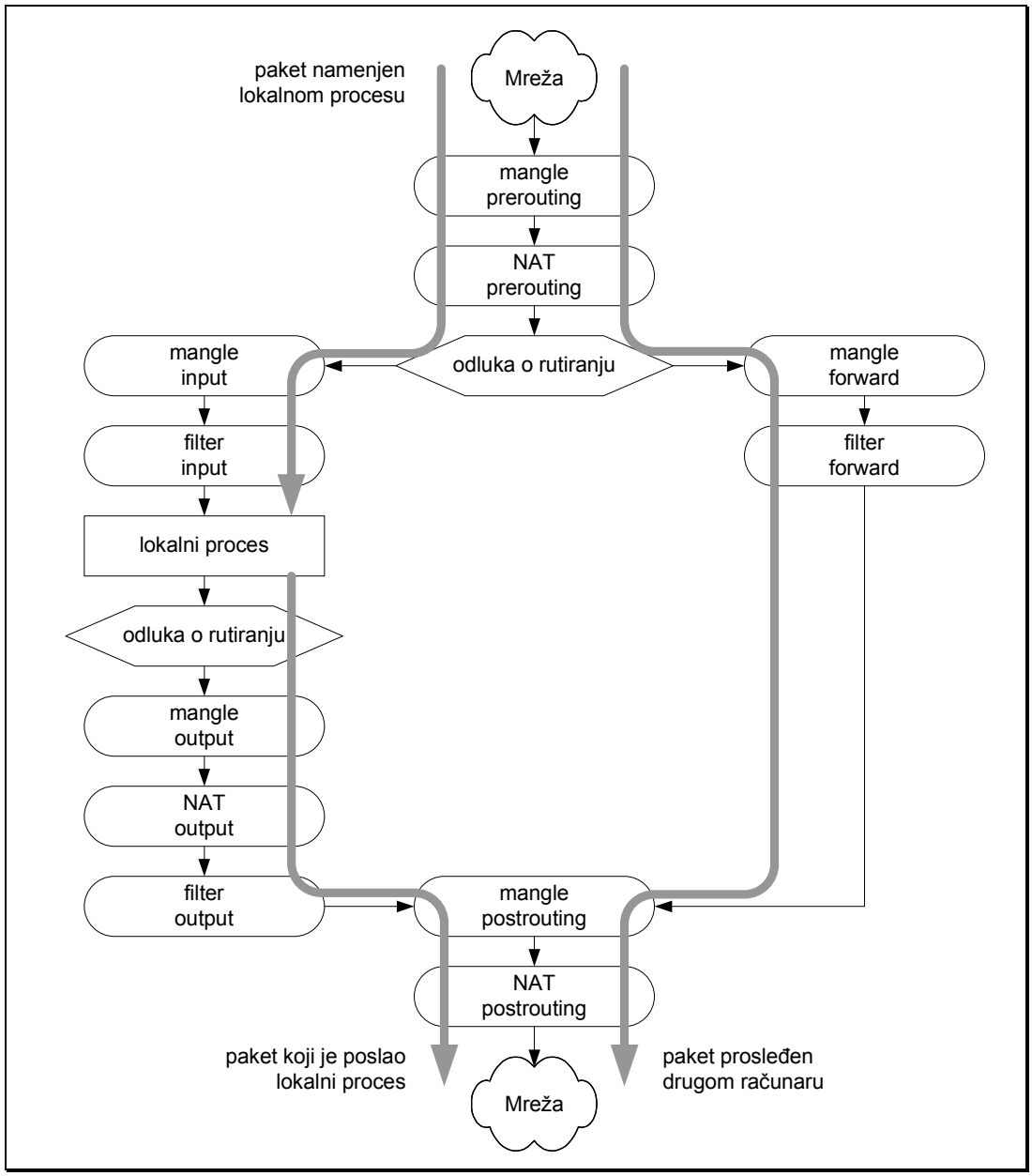
TTL — TTL se koristi da promeni Time To Live polje paketa;

MARK — MARK polje se koristi za markiranje paketa. Markiranje paketa je korisno ukoliko je potrebno da se izvrši neka operacija nad određenim paketom van iptables barijere. Markirane pakete prepoznaju programi poput iproute2 koji na osnovu vrednosti MARK polja izvršavaju različito rutiranje paketa. Takođe, na osnovu ovih ocena može se ograničiti širina opsega. Na primer, sledećom komandom se mogu markirati sva zaglavlja SMTP paketa za server sa brojem 1:

```
$ iptables -t mangle -p tcp -dport 25 -j MARK -mark 1
```

Put paketa kroz iptables

Paket koji stigne do mrežne barijere prosleđuje se na odgovarajući drajver za uređaj u kernelu. Paket dalje prolazi kroz seriju koraka u kernelu pre nego što se šalje pravoj aplikaciji (lokalno) ili preusmerava ka drugom računaru, kao što je prikazano na slici 14.5.



Slika 14.5 *Put paketa kroz iptables*

Paket namenjen procesu (aplikaciji), koji se izvršava na računaru na kom funkcioniše i iptables, prolazi kroz sledeće korake pre nego što se isporuči aplikaciji:

- paket ulazi u interfejs (na primer, eth0)
- mangle prerouting
- nat prerouting
- odluka o rutiranju
- mangle input
- filter input
- paket se šalje pravoj aplikaciji ili drugom računaru

Paket sa lokalnog računara, odnosno sa računara na kom funkcioniše i iptables, prolazi kroz sledeće korake pre nego što se pošalje na mrežu:

- lokalni proces generiše paket
- odluka o rutiranju
- mangle output
- nat output
- filter output
- mangle postrouting
- nat postrouting
- paket se šalje na mrežni interfejs

Paket koji dolazi sa mreže, a koji je namenjen drugom računaru na mreži, prolazi kroz sledeće korake:

- paket ulazi u mrežni interfejs (na primer eth0)
- mangle prerouting
- nat prerouting
- odluka o rutiranju
- mangle forward
- filter forward
- mangle postrouting
- nat postrouting
- paket se šalje na mrežni interfejs (na primer eth1)

Paket se može zaustaviti na bilo kom lancu iptables mrežne barijere. Lanci i tabele su jedinstveni za sve mrežne interfejse.

Administracija iptables mrežne barijere

Administracija iptables mrežne barijere vrši se pomoću skupa komandi koje se zadaju u sledećem obliku:

```
$ iptables command args
```

U nastavku teksta opisane su relevantne komande i dati odgovarajući primeri:

-A,--append — komanda dodaje novo pravilo na kraj lanca, odnosno kao posledenje u setu pravila. Primer komande append je:

```
$ iptables –A INPUT …
```

-D,--delete — briše pravilo iz lanca. Pravilo se može obrisati na dva načina: upisivanjem pravila (zahteva se tačan unos) ili navođenjem broja pravila koje se briše. Pravila se numerišu sa vrha svakog lanca počev od broja 1. Primer brisanja pravila iz lanca je:

```
$ iptables –D INPUT --dport 80 –j DROP , iptables –D INPUT 1
```

-R, --replace — zamena starog pravila na određenoj liniji novim pravilom. Primer komande je:

```
$ iptables –R INPUT 1 –s 192.168.0.1 –j DROP
```

-I, --insert — ubacuje pravilo na bilo kom mestu u lancu, pri čemu se mora navesti broj na kom se pravilo ubacuje.

```
$ iptables –I INPUT 1 --dport 80 –j ACCEPT
```

-L, --list — prikazuje sva pravila u navedenom lancu.

```
$ iptables –L INPUT
```

-F, --flush — briše sva pravila iz navedenog lanca. Ukoliko se lanac ne navede brišu se sva pravila u svim lancima unutar navedene tabele.

```
$ iptables –F INPUT
```

-Z,--zero — postavlja na nulu sve brojače u određenom lancu.

```
$ iptables –Z INPUT
```

-N,--new-chain — kreira novi lanac određenog imena u određenoj tabeli.

```
$ iptables –N externallow
```

-X,--delete-chain — briše određeni lanac iz tabele pod uslovom da su prethodno obrisana pravila koja se odnose na taj lanac.

```
$ iptables –X externallow
```

-P,--policy — kernel postavlja određenu polisu na lancu. Svi paketi koji ne odgovaraju nijednom pravilu moraju da koriste polisu lanca.

```
$ iptables –P INPUT DROP
```

-E,--rename-chain menja ime lanca, što ne utiče na način rada tabele.

```
$ iptables –E allowed disallowed
```

Dodatno, postoji i određeni broj opcija koje se primenjuju na jednu ili više komandi:

-v, --verbose — opcija ide uz komande --list, --append, --insert, --delete, --replace. Ukoliko se navede sa komandom --list, iptables daje detaljan izlaz, odnosno prikazuje adresu interfejsa, pravila i TOS maske, kao i bajtove i brojač paketa za svako pravilo. Ako se opcija -v koristi sa komandama --append ,--insert,--delete ili --replace, program će na izlazu prikazati detaljne informacije o tome kako je pravilo protumačeno, odnosno da li je ispravno navedeno.

-x, --exact — opcija ide uz komandu --list. Ukoliko se navede, komanda --list će prikazati tačan izlaz paketa i brojače bajtova koji će brojati koliko paketa i bajtova odgovaraju navedenom pravilu.

-n, --numeric — opcija ide uz komandu --list. Ukoliko se navede, iptables će na izlazu prikazivati IP adrese i brojeve porta u numeričkom formatu (ne kao imena računara ili servisa).

--line-numbers — opcija ide uz komandu --list. Ukoliko se navede, iptables će na izlazu prikazivati brojeve svakog pravila u lancu.

-c, --set-counters — opcija ide uz komande --insert, --append i --replace. Koristi se za inicijalizaciju brojača paketa i bajtova za svako pravilo. Sintaksa je --set-counters npack nbyte, što govori kernelu da postavi brojač paketa na npack i brojač bajtova na nbyte.

--modprobe — ide uz sve komande i koristi se da ukaže iptables mrežnoj barijeri koji modul treba da koristi kada traži module ili ih dodaje kernelu. Koristi se ako komanda modprobe, koja služi za rad sa modulima kernela, nije u sistemskoj putanji. U tom slučaju, ova opcija naznačuje programu iptables šta da radi u slučaju da potrebni modul nije učitan u jezgro.

A

PREGLED ZNAČAJNIJIH LINUX KOMANDI

U nastavku teksta dat je kraći opis značajnijih komandi Linux sistema. Ovaj dodatak nije zamišljen kao detaljan izvor informacija o svim argumentima i opcijama komandi, već kao podsetnik u kome su ukratko opisane najčešće korišćene komande i operacije koje se njima mogu obaviti. Detaljna uputstva o sintaksi, argumentima i opcijama komandi korisnici mogu dobiti korišćenjem on-line dokumentacije, odnosno odgovarajućih stranica uputstva (man pages).

A

adduser	Kreiranje korisničkih naloga.
anacron	Administrativna komanda koja se normalno pokreće pri podizanju sistema i periodično izvršava komande. Lista poslova se podrazumevano čita iz datoteke /etc/anacrontab.
apropos	Na standardnom izlazu prikazuje ime i opis svih komandi koje u opisu imaju zadati string.
apt-get	Debian Package Management System - alat za rad sa paketima iz komandne linije. Front-end za APT.
aptitude	Debian Package Management System - alat za interaktivni rad sa paketima. Front-end sa sistemom tekstualnih menija za APT.
arch	Prikazuje arhitekturu računara na standardnom izlazu (kao uname -m).
arp	TCP/IP administrativna komanda za rad sa ARP kešom kernela. ARP se koristi za prevođenje IP adresa u MAC adrese mrežnih adaptera.
at	Komanda kojom se zakazuje izvršenje drugih komandi u određeno vreme.
atd	Daemon koji izvršava komande zakazane komandom at. Normalno se pokreće prilikom podizanja sistema.

atq	Prikazuje zakazane at poslove korisnika. U slučaju da komandu zada superuser, prikazuju se svi zakazani poslovi.
atrm	Brisanje zakazanih at poslova

B

badblocks	Administrativna komanda za analizu površine diskova
banner	Na standardnom izlazu prikazuje string kao "poster".
basename	Prikazuje ime direktorijuma bez vodeće apsolutne putanje (na primer: basename $PWD). Komanda je korisna za shell programiranje.
bash	Bash komandni interpreter (Bourne-Again Shell).
bashburn	Front-end za cdrecord (snimanje CD-ROM medijuma).
batch	Slično komandi at, izvršava komande date na standardnom ulazu. Ukoliko se vreme izvršenja ne navede, komande se izvršavaju kada opterećenje sistema dostigne dovoljno nizak nivo (80%).
bc	Interaktivni kalkulator koji ulazne podatke prima sa standardnog ulaza ili iz datoteke.
biod	NFS server.
bzip2	Paket za kompresiju sličan programu gzip koji koristi drugačije algoritme i postiže veći stepen kompresije. Paket uključuje programe za kompresiju i dekompresiju (bzip2, bunzip2), pregledanje sadržaja (bzcat, bzless, bzmore) i oporavak (bzip2recover).

C

cal	Prikazuje kalendar za tekući mesec. Takođe može prikazati godišnji ili mesečni kalendar za godinu i mesec koji se navode kao parametri.
cancel	Uklanjanje poslova iz reda za štampu (System V).
cat	Konkatenacija datoteka i prikazivanje na standardnom izlazu.
cdrecord	Program za snimanje kompakt diskova sa velikim brojem opcija.
cfdisk	Administrativni program sa sistemom menija za particionisanje diskova.
chage	Prikazivanje i promena informacija o isticanju lozinki.
chat	Administrativna komanda za postavljanje i inicijalizaciju dial-up konekcija, koja se koristi zajedno sa PPP daemon procesom pppd.
chattr	Modifikacija specijalnih atributa datoteka, karakterističnih za ext2 i ext3 sisteme datoteka.
chfn	Promena opisnih informacija o korisniku.

chgrp — Promena grupe kojoj objekat sistema datoteka pripada. U opštem slučaju ovu komandu može da izvrši root, a na nekim sistemima može i vlasnik objekta.

chmod — Promena pristupnih prava objekta sistema datoteka. Ovu komandu mogu da izvrše root i vlasnik objekta.

chown — Promena vlasnika objekta sistema datoteka. U opštem slučaju ovu komandu može da izvrši root, a na nekim sistemima može i vlasnik objekta.

chpasswd — Administrativna komanda za promenu lozinke.

chsh — Promena komandnog interpretera koji se pokreće nakon login procesa. Ime komandnog interpretera se navodi sa apsolutnom putanjom.

chvt — Prelazak na virtuelni terminal N. Ukoliko terminal ne postoji biće napravljen. Ekvivalentno kombinaciji tastera <Ctrl-Alt-N>, gde je N broj terminala.

cksum — Računanje CRC ček-sume za datoteku.

clear — Brisanje ekrana.

cmp — Upoređivanje datoteka i prikazivanje prve razlike na standardnom izlazu.

compress — Program za kompresiju datoteka. Gzip i bzip2 se danas koriste umesto ovog programa.

cp — Kopiranje datoteke, grupe datoteka ili delova direktorijumskog stabla. Takođe se može koristiti za kreiranje linkova.

cpio — Arhiviranje i dearhiviranje datoteka (copy-out i copy-in). Takođe se može koristiti za kopiranje datoteka u aktivnom UNIX stablu (copy-pass).

cron — Administrativna komanda koja se normalno pokreće pri podizanju sistema i periodično izvršava komande. Cron proverava korisničke crontab datoteke (nalaze se u direktorijumu /var/spool/cron/crontabs a imenovane su na osnovu korisničkih naloga) svaki minuta i po i pokreće programe koje tada treba izvršiti.

crontab — Zakazivanje periodičnog izvršenja komandi, odnosno izvršenja u specificiranim intervalima. Zavisno od konkretnog UNIX sistema izvršenje mogu zakazati svi ili samo privilegovani korisnici.

csh — C shell. Na Linux sistemima je zamenjen naprednijom varijantom (tsch).

D

date — Prikazuje trenutni datum i vreme.

dd — Konvertuje i kopira datoteku ili određeni deo medijuma. Prilikom pristupa medijumu može zaobići sistem datoteka, čime je omogućeno kopiranje medijuma koji nisu ni u jednom od formata koje UNIX prepoznaje.

debugfs	Administrativna komanda kojom se ostvaruje pristup zaglavlju i meta-data strukturama ext2 sistema datoteka.
depmod	Kreira datoteku u kojoj je opisana međusobna zavisnost programskih modula.
df	Prikazuje iskorišćenost aktiviranih sistema datoteka.
diff	Upoređuje datoteke i prikazuje sve razlike na standardnom izlazu.
dig	Komanda za slanje upita DNS serverima, fleksibilnija od nslookup komande.
dpkg	Debian Package Management System - rad sa paketima iz komandne linije.
dselect	Debian Package Management System - okruženje sa menijima.
du	Prikazuje količinu prostora na sistemu datoteka koju zauzimaju datoteke u poddirektorijumima tekućeg direktorijuma.
dumpe2fs	Administrativna komanda koja na standardnom izlazu prikazuje infomacije iz superbloka sistema datoteka

E

e2fsck	Administrativna komanda za proveru integriteta ext2 i ext3 sistema datoteka.
e2image	Administrativna komanda za kreiranje slike (image) značajnijih delova sistema datoteka (kao što je superblok) na izmenljivom medijumu (disketi).
e2label	Administrativna komanda za prikazivanje i promenu imena sistema datoteka.
echo	Prikazuje niz karaktera ili vrednost promenljive na standardnom izlazu.
edquota	Editor kvota.
egrep	Traži regularne izraze u datoteci.
emacs	Emacs tekst editor.
env	Prikazuje vrednosti promenljivih koje čine okruženje.
expr	Obavlja jednostavne aritmetičke operacije.

F

fdformat	Formatiranje disketa niskog nivoa.
fdisk	Administrativni program za particionisanje hard diskova.

fgconsole	Prikazuje broj trenutno aktivne virtuelne konzole (na primer 2, ukoliko korisnik radi na /dev/tty2).
fgrep	Traži niz karaktera u datoteci.
find	Traži datoteku na osnovu zadatih kriterijuma u aktivnom UNIX stablu.
finger	Prikazuje informacije o korisnicima, uključujući i informacije iz datoteka .plan i .project u home direktorijumu korisnika.
fingerd	Server za informacije o korisnicima.
free	Prikazuje informacije o iskorišćenosti operativne memorije i swap prostora.
fsck	Administrativna komanda za proveru integriteta sistema datoteka.
ftp	Interaktivni program za transfer datoteka između dva udaljena sistema.
ftpd	FTP server.

G

gpm	Server za miša koji obezbeđuje cut-and-paste funkcionalnost na Linux konzoli.
grep	Traži osnovne regularne izraze u datoteci.
groupadd	Administrativna komanda za kreiranje nove korisničke grupe.
groupdel	Administrativna komanda za brisanje postojeće korisničke grupe.
groupmod	Administrativna komanda za modifikovanje parametara grupe.
groups	Prikazuje grupe kojima navedeni korisnik pripada.
grpck	Administrativna komanda koja uklanja duplikate iz datoteke /etc/group.
grpconv	Administrativna komanda za kreiranje /etc/gshadow datoteke, u koju se smeštaju sve grupne lozinke.
grpunconv	Administrativna komanda kojom se uklanja /etc/gshadow datoteka.
gunzip	Dekompresija .gz datoteka.
gzexe	Kreiranje samoraspakujuće izvršne gzip datoteke. Nakon pokretanja takva datoteka će se sama dekompresovati. Ovde se štedi na prostoru, ali gubi na vremenu potrebnom za izvršavanje datoteke.
gzip	Kompesija datoteka u .gz format.

H

halt	Administrativna komanda za zaustavljanje sistema. Ukoliko se sistem nalazi u nivoima izvršenja 0 ili 6 halt zaustavlja sve procese, a inače poziva komandu shutdown -h.
hdparm	Administrativna komanda za pregledanje i postavljanje parametara hard diskova. Koristi se uglavnom na IDE diskovima.
head	Prikazuje početak datoteke.
hexdump	Prikazuje datoteku u heksadecimalnom ili oktalnom formatu.
host	Prikazuje informacije o računarima i zonama u DNS domenu.
hostname	Prikazuje ime računara, pri čemu privilegovani korisnik može dodeliti novo ime računaru.
hwclock	Administrativna komanda kojom privilegovani korisnik može podesiti hardverski časovnik sistema.

I

id	Prikazivanje informacija o korisnicima, uključujući i članstvo u grupama.
ifconfig	TCP/IP administrativna komanda za konfigurisanje mrežnih interfejsa rezidentnih u kernelu. Komandom ifconfig mogu se postaviti parametri poput IP adrese, maske podmreže i broadcast adrese. Komanda ifconfig se koristi prilikom podizanja sistema radi postavljanja parametara mrežnog interfejsa. Nakon toga najčešće se koristi u dijagnostičke svrhe - komanda prikazuje trenutnu konfiguraciju mrežnog interfejsa i MAC adresu mrežne kartice.
inetd	TCP/IP administrativna komanda kojom se može zaustaviti ili pokrenuti inetd wrapper.
init	Osnovni proces i administrativna komanda za inicijalizaciju sistema (forsira se novo čitanje konfiguracione datoteke /etc/inittab) i promenu nivoa izvršenja.
ipchains	Administrativna komanda za konfigurisanje firewalla u Linux kernelu 2.2
iptables	Administrativna komanda za konfigurisanje firewalla u Linux kernelu 2.4

K

kernelversion	Prikazuje verziju tekućeg kernela.
kill	Šalje signale procesu sa poznatim PID

killall	Ubija procese nastale pokretanjem određenog programa (kao argument se navodi ime programa).

L

last	Prikazuje nekoliko poslednjih login procedura, odnosno imena korisnika, terminal, ime udaljenog računara i vreme prijavljivanja na sistem.
lastb	Prikazuje nekoliko poslednjih neuspešnih login procedura, u istom formatu kao i last.
lastlog	Prikazuje sve korisnike sistema i vreme kada su se zadnji put prijavili na sistem.
less	Interaktivni program za pregledanje sadržaja tekstualnih datoteka.
ln	Kreiranje hard i simboličkih linkova.
locale	Štampa izveštaj o regionalnim podešavanjima na standardnom izlazu.
lockfile	Kreira specijalne semafor datoteke koje se koriste za ograničavanje pristupa datotekama.
login	Prijavljivanje na sistem. Treći proces u nizu init-getty-login-shell.
logname	Prikazuje ime korisnika koji je prijavljen na sistem na osnovu podataka u datoteci /var/run/utmp.
look	Prikazuje reči iz datoteke /usr/dict/words koje počinju zadatim nizom karaktera.
lp	Štampanje iz komandne linije (System V).
lpadmin	Administracija CUPS servisa za štampu.
lpc	Administrativna komanda za kontrolu LPRng servisa za štampu.
lpd	LPRng line printer daemon (servis za štampu).
lpinfo	Informacije o štampačima (CUPS).
lpq	Ispitivanje reda za štampu (BSD).
lpr	Štampanje iz komandne linije (BSD).
lprm	Uklanjanje poslova iz reda za štampu (BSD).
lpstat	Ispitivanje reda za štampu (System V).
ls	Prikazuje sadržaj direktorijuma na standardnom izlazu.
lsattr	Prikazuje specijalne atribute datoteka karakteristične za ext2 i ext3 sisteme datoteka.
lsmod	Prikazuje module učitane u tekuće jezgro.

M

mail	Prikazivanje, čitanje i slanje pošte drugim korisnicima sistema.
make	Prevođenje i povezivanje izvornog koda na osnovu datoteke makefile.
man	Prikazuje stranicu uputstva (man page) za određenu komandu.
mtools	Alati za rad sa MS-DOS diskovima. Koriste DOS sintaksu, odnosno korisnici se diskovima obraćaju preko labela, a ne preko nodova.
mesg	Komanda kojom korisnik dozvoljava ili zabranjuje drugim korisnicima da mu šalju poruke komandom write.
mkdir	Kreiranje direktorijuma u aktivnom UNIX stablu.
mkfs	Administrativna komanda, front-end za alate kojima se kreiraju sistemi datoteka.
mkfs.ext2	Administrativna komanda za kreiranje ext2 sistema datoteka (mke2fs).
mkfs.ext3	Administrativna komanda za kreiranje ext3 sistema datoteka.
mkfs.msdos	Administrativna komanda za kreiranje MS-DOS sistema datoteka.
mkfifo	Kreiranje imenovanih FIFO datoteka (imenovani pipe).
mkisofs	Kreiranje ISO9660/Joliet/HFS (Macintosh Hierarchical File System) sistema datoteka radi snimanja CD-ROM medijuma alatom cdrecord.
mklost+found	Kreiranje lost+found direktorijuma na ext2 sistemima datoteka.
mknod	Kreiranje specijalnih datoteka (nodova), odnosno datoteka koje mogu da šalju i primaju podatke (karakter i blok uređaji).
mkraid	Administrativna komanda za kreiranje novog RAID niza diskova definisanog konfiguracionom datotekom /etc/raidtab. Inicijalizacija uništava sve podatke na diskovima koji čine RAID niz.
mkswap	Administrativna komanda za kreiranje logičke strukture swap datoteke ili particije.
modinfo	Štampa na standardnom izlazu informacije o određenom modulu kernela. Informacije se čitaju iz zaglavlja datoteke u kojoj se taj modul nalazi.
modprobe	Administrativna komanda za rad sa modulima kernela. Uz komandu depmod omogućava lakši rad sa modulima.
more	Komanda za pregledanje sadržaja tekstualnih datoteka.
mount	Administrativna komanda za aktiviranje sistema datoteka (montiranje na mount-point direktorijume). Svi korisnici pomoću ove komande mogu utvrditi koji su sistemi datoteka trenutno aktivirani.
mountd	NFS/NIS administrativna komanda. Server koji upravlja zahtevima za montiranje NFS sistema datoteka.

mt	Administrativna komanda za upravljanje magnetnim trakama.
mv	Pomeranje datoteke, grupe datoteka ili direktorijuma sa jedne lokacije na drugu. Promena imena datoteka.

N

named	Server imena (DNS).
nameif	Administrativna komanda za dodelu imena interfejsa sa zadatom MAC adresom.
netstat	TCP/IP dijagnostički alat koji daje izveštaje o mrežnom interfejsu, tabelama rutiranja, mrežnim konekcijama i statistici korišćenja TCP/IP skupa protokola.
newusers	Administrativna komanda za kreiranje korisničkih naloga na osnovu sadržaja datoteke koja se navodi kao parametar (u datoteci se navode korisnička imena i lozinke). Prilikom kreiranja korisničkih naloga, kreiraju se i grupe i home direktorijumi, ukoliko već ne postoje.
nfsd	NFS klijent.
nfsstat	NFS dijagnostički alat koji štampa statistički izveštaj o korišćenju NFS-a.
nice	Izvršavanje komande sa nižim prioritetom ("be nice to other users").
nohup	Pokreće program (komanda se zadaje kao argument) čije se izvršenje nastavlja nakon odjavljivanja korisnika sa sistema.
nslookup	Komanda za ispitivanje DNS servera.
nsupdate	Administrativna komanda za podnošenje zahteva za dinamičko ažuriranje servera imena.

P

passwd	Promena lozinke korisnika.
ping	TCP/IP dijagnostički alat za slanje ICMP ECHO paketa. Ovim alatom se može utvrditi da li je udaljeni računar dostupan.
pppd	PPP (Point-to-Point Protocol) server.
pr	Priprema tekstualnih datoteka za štampanje (podela datoteke na stranice, numerisanje stranica i navođenje datuma i imena datoteke u zaglavlju).
ps	Štampa izveštaj o procesima na standardnom izlazu.
pwck	Provera integriteta passwd datoteke.
pwconv	Administrativni alat za kreiranje datoteke /etc/shadow. Originalne lozinke u datoteci /etc/passwd zamenjuju se znakom x.

pwunconv	Administrativni alat za uklanjanje datoteke /etc/shadow.
pwd	Štampa na standardnom izlazu putanju tekućeg direktorijuma.

Q

quota	Prikazuje zauzeće diska od strane određenog korisnika ili grupe i ograničenja u sistemu datoteka.
repquota	Prikazuje informacije o zauzeću diska i kvotama za navedeni sistem datoteka, trenutnom broju datoteka i zauzeću diska za svakog korisnika kome su dodeljene kvote.
quotacheck	Na osnovu analize potrošnje prostora na odgovarajućem sistemu datoteka kreira odgovarajuće datoteke quota.user i quota.group.
quotaon	Aktiviranje kvote.
quotaoff	Deaktiviranje kvote.

R

rcp	Kopiranje datoteka između udaljenih računara.
readlink	Prikazuje sadržaj simboličkog linka, odnosno putanju i ime objekta na koji link pokazuje.
reboot	Administrativna komanda za zaustavljanje i ponovno podizanje sistema koja odmah ubija sve procese. Ukoliko sistem nije u nivou izvršenja 0 ili 6, reboot poziva komandu shutdown -r.
rename	Promena imena većeg broja datoteka (jedan niz karaktera u imenima se menja drugim).
renice	Promena prioriteta procesa.
resize2fs	Administrativna komanda za promenu veličine sistema datoteka. Komanda zahteva da se najpre programom fdisk obriše particija u kojoj se nalazi sistem datoteka, a zatim kreira nova (veća) počev od istog cilindra.
rev	Štampa datoteku na standardnom izlazu, pri čemu svaku liniju datoteke štampa unazad.
rm	Brisanje datoteke, grupe datoteka i delova direktorijumskog stabla.
rmdir	Brisanje praznih direkorijuma.
rmmod	Uklanjanje modula iz tekućeg jezgra.
route	TCP/IP command za izmenu sadržaja tabele rutiranja (tabele rutiranja održava routed).
routed	Mrežni ruter, podržava Routing Information Protocol.

rpm	Red Hat Package Manager.
runlevel	Prikazuje nivo izvršavanja.
run-parts	Pokreće skriptove u direktorijumu po abecednom redu.

S

sed	Stream editor - modifikacija sadržaja datoteka bez interakcije korisnika.
setfdprm	Konfigurisanje parametara flopi diskova.
sftp	Siguran transfer datoteka između udaljenih računara putem ssh.
showmount	NFS/NIS koja prikazuje informacije o NFS serveru.
shred	Prepisuje slučajni sadržaj preko datoteke, nakon čega briše datoteku. Time se obezbeđuje da se datoteka ne može povratiti.
shutdown	Administrativna komanda za zaustavljanje sistema.
sort	Uređivanje sadržaja datoteka.
split	Deljenje datoteka na segmente jednake veličine.
ssh	Secure Shell - sigurno prijavljivanje na udaljeni sistem (podaci na liniji se šifruju).
sshd	Secure Shell server.
strings	Prikazuje vidljive karaktere u izvršnoj ili binarnoj datoteci.
stty	Podešavanje karakteristika terminala.
su	Privremeno prijavljivanje na sistem sa drugim korisničkim nalogom.
sudo	Izvršavanje komandi sa root privilegijama.
sum	Računa i prikazuje ček-sumu i veličinu datoteka. Komanda je korisna za verifikaciju transfera datoteka.
swapoff	Administrativna komanda za isključivanje swap prostora.
swapon	Administrativna komanda za uključivanje swap prostora.
sync	Administrativna komanda koja upisuje sadržaj keša na disk i prazni keš.
sysklogd	Daemon koji poruke operativnog sistema upisuje u odgovarajuće log datoteke.

T

tac	Štampa sadržaj datoteke na standardnom izlazu počev od poslednje linije ka prvoj.
tail	Prikazuje kraj datoteke (nekoliko poslednjih linija).

tar	Tape Archiver - arhiviranje i dearhiviranje datoteka.
tcpd	TCP/IP wrapper.
tcsh	Poboljšana verzija C shell komandnog interpretera.
tee	Podatke sa standardnog ulaza upisuje u datoteku i šalje ih na standardni izlaz.
telnet	Prijavljivanje na udaljeni sistem. Za razliku od ssh, komunikaciona linija se ne šifruje, tako da nije sigurna.
telnetd	Telnet server.
time	Izvršava komandu i određuje vreme potrebno za izvršenje te komande.
tload	Prikazuje grafik opterećenja sistema.
top	Obezbeđuje informacije o procesima koji su kritični po pitanju potrošnje procesorskog vremena (top processes).
touch	Postavlja vreme zadnjeg pristupa i vreme poslednje modifikacija na tekuće vreme. Ukoliko datoteka ne postoji, kreiraće praznu datoteku.
traceroute	Identifikacija rute do odredišnog računara
tty	Prikazuje ime uređaja koji se koristi kao standardni ulaz.
tune2fs	Administrativna komanda za podešavanje parametara ext2 sistema datoteka.
tunelp	Administrativna komanda za podešavanje parametara štampača.

U

umount	Administrativna komanda za deaktiviranje sistema datoteka.
uname	Prikazuje ime računara, arhitekturu hardvera i ime operativnog sistema.
uncompress	Dekompresija .Z datoteka.
uniq	Uklanja sve duplikate linija iz tekstualne datoteke (izbacuje sve konsekventne identične linije, osim prve u nizu).
uptime	Prikazuje vreme proteklo od poslednjeg podizanja sistema, broj trenutno prijavljenih korisnika i prosečno opterećenje sistema.
useradd	Administrativna komanda za kreiranje korisničkih naloga.
userdel	Administrativna komanda za brisanje korisničkih naloga.
usermod	Modifikacija parametara korisničkog naloga.

V

vdir	Ekvivalentna komandi ls -lb.

vi	Vi tekst editor, prisutan na svim UNIX sistemima.
vim	Vi Improved, poboljšana verzija vi editora.
vmstat	Prikazuje statistički izveštaj o memoriji, swap prostoru, iskorišćenosti procesora i procesima.

W

w	Prikazuje koji su korisnici prijavljeni na sistem i "šta trenutno rade".
wall	Slanje poruke svim korisnicima ("Broadcast Message from...").
watch	Izvršava zadatu komandu repetativno (podrazumevano na svake dve sekunde) tako da korisnik može da prati izlaz komande.
wc	Brojanje karaktera, reči i linija u datoteci.
whatis	Na standardnom izlazu prikazuje opis navedene komande.
whereis	Prikazuje lokaciju izvršnih datoteka, izvornog koda i prateće dokumentacije programa.
which	Prikazuje lokaciju izvršne datoteke.
who	Prikazuje koji su korisnici prijavljeni na sistem.
whoami	Prikazuje korisničko ime korisnika koji je komandu zadao.
write	Slanje poruka određenom korisniku.

X

xinetd	TCP/IP wrapper (extended Internet services daemon). Na nekim sistemima se koristi umesto inetd wrappera.

Y

ypbind	NFS/NIS komanda. Proces kojim se klijentu dodeljuje NIS server.
ypcat	NFS/NIS komanda za prikazivanje informacija u NIS bazi.
ypchfn	NFS/NIS komanda za promenu informacija u datoteci /etc/passwd.
ypinit	NFS/NIS komanda za kreiranje NIS baze na NIS serveru.
yppasswd	NFS/NIS komanda za promenu lozinke za prijavljivanje na NIS domen.
yppasswdd	Rešava zahteve za promenu lozinke korisnika sa udaljenih računara.
yppoll	Zahtev slave servera za ažuriranje baze sa master serverom.
yppush	Prosleđuje tabele sa master servera na slave servere.

ypserv	NIS server.
yptest	NFS/NIS komanda za proveru konfiguracije NIS servisa.
ypwhich	NFS/NIS komanda koja prikazuje ime NIS servera koji opslužuje lokalni računar.
ypxfr	Program koji vrši transfer tabela sa jednog servera na drugi.
ypxfrd	Proces na master serveru koji upravlja prenošenjem ažuriranih podataka na slave server.
ypupdated	Proces na slave serveru koji upravlja prenošenjem ažuriranih podataka sa master servera.
ypset	Zahtev da sistem bude server procesu ypbin na klijent računaru.

Z

zcat	Komanda cat za .Z i .gz datoteke.
zmore	Komanda more za .Z i .gz datoteke.
znew	Dekomprimuje .Z kompresovanu datoteku, a zatim je kompresuje u .gz format.

B

GNU OPŠTA JAVNA LICENCA

Navodimo prevod GNU GPL licence (General Public Licence) na srpski jezik. Da spomenemo, GNU je rekurzivni akronim (GNU = **G**NU IS **N**OT **U**NIX).

GNU-ova bibliotečka opšta javna licenca (BOJL, eng. LGPL/Library General Public License) se sada zove GNU-ova manja opšta javna licenca (MOJL, eng. LGPL/Lesser General Public License).

GNU OPŠTA JAVNA LICENCA

Verzija 2, jun 1991 (1)

UVOD

Za većinu softvera licence su načinjene sa ciljem da vam oduzmu slobodu da ga delite sa drugima i menjate. Nasuprot tome, GNU-ova opšta javna licenca vam garantuje slobodu deljenja i izmene slobodnog softvera da bi osigurala slobodu softvera za sve njegove korisnike. Ova Opšta javna licenca se odnosi na većinu softvera iz Zadužbine za slobodni softver i na svaki drugi program, čiji se autori obavežu na njeno korišćenje. (Drugi softver Zadužbine za slobodni softver je umesto ove licence pokriven GNU-ovom bibliotečkom opštom javnom licencom (2).) I vi je možete primeniti na vaše programe.

Kada govorimo o slobodnom softveru, mislimo na slobodu, a ne na cenu. Naše opšte javne licence su zamišljene da osiguraju vašu slobodu raspodele primeraka slobodnog softvera (i naplaćivanja za tu uslugu po želji), primanja izvornog koda ili njegovog dobijanja po želji, mogućnosti izmene softvera ili korišćenja delova istog u novim slobodnim programima; i da vam daju do znanja da to možete da uradite.

Da bismo zaštitili vaša prava, moramo da postavimo ograničenja koja zabranjuju bilo kome da vam ospori ova prava ili da traži od vas da ih se odreknete. Ako raspodeljujete primerke softvera ili ga izmenite, ova ograničenja su za vas obavezujuća.

Na primer, ako raspodeljujete primerke takvog programa, besplatno ili za određenu novčanu nadoknadu, morate dati i primaocima sva prava koja sami posedujete. Morate se postarati da i oni prime ili mogu da dobiju izvorni kod. Najzad, morate im pokazati ove odredbe da bi bili upoznati sa svojim pravima.

Mi štitimo vaša prava u dva koraka: (1) štitimo softver zaštitom autorskih prava, i (2) nudimo Vam ovu licencu koja Vam daje pravnu dozvolu da umnožavate, raspodeljujete i/ili menjate softver.

Takođe, radi zaštite svakog autora i nas hoćemo da se osiguramo da svako razume da ne postoji garancija za ovaj slobodni softver. Ako je softver neko drugi izmenio i prosledio ga, hoćemo da primaoci znaju da ono što imaju nije original, tako da se bilo kakvi problemi koji su nastali zbog drugih, neće odraziti na ugled autora originala.

Najzad, sve slobodne programe neprestano ugrožavaju softverski patenti. Mi hoćemo da izbegnemo opasnost da raspodeljivači slobodnog programa individualno dobiju patentne licence i na taj način stave program u privatno vlasništvo. Da bismo sprečili ovo, jasno smo istakli da svaki patent mora da bude licenciran za svačiju slobodnu upotrebu ili da uopšte ne bude licenciran.

Precizne odredbe i uslovi umnožavanja, raspodele i izmene slede.

ODREDBE I USLOVI UMNOŽAVANjA, RASPODELE I IZMENE

0. Ova licenca se odnosi na svaki program ili drugo delo koje sadrži saopštenje vlasnika autorskih prava u kom stoji da može biti raspodeljen pod odredbama ove opšte javne licence. „Program“ će nadalje označavati svaki takav program ili rad, a „delo zasnovano na Programu“ će označavati Program ili bilo koji rad proistekao iz njega po Zakonu o autorskim pravima; tj. delo koje sadrži Program ili njegov deo, bilo doslovni ili sa izmenama i/ili preveden na drugi jezik. (Odavde pa nadalje, prevod je uključen bez ograničenja u pogledu izraza „izmena“.) Svaki korisnik licence je označen kao „vi“.

   Druge aktivnosti osim umnožavanja, raspodele i izmene nisu obuhvaćene ovom licencom; one su izvan njenog domena. Čin pokretanja Programa nije ograničen, a dobijeni rezultat izvršavanja Programa je obuhvaćen u slučaju da se njegov sadržaj sastoji od dela zasnovanog na Programu (nezavisno od činjenice da je nastao kao rezultat pokretanja Programa). Ovo neposredno zavisi od toga šta Program radi.

1. Vi možete umnožavati i raspodeliti doslovne primerke izvornog koda Programa čim ga primite, na bilo kojem medijumu, uz uslov da na adekvatan i odgovarajući način označite na svakom primerku odgovarajuće saopštenje o autorskim pravima i objašnjenje garancije; sačuvate nedirnutim saopštenja koja se pozivaju na ovu licencu i odsustvo bilo kakve garancije; i svim drugim primaocima Programa date primerak ove licence zajedno sa Programom.

   Vi možete naplaćivati novčanu nadoknadu za čin fizičkog prenosa primerka, i po vašem izboru možete ponuditi zaštitu garancijom u zamenu za novčanu nadoknadu.

2. Vi možete izmeniti vaš primerak ili primerke Programa ili bilo kog njegovog dela, obrazujući delo zasnovano na Programu, i umnožavati i raspodeliti takve izmene ili dela pod gornjim odredbama člana 1, uz uslov da sami takođe ispunite svaki od sledećih uslova:

   a) morate osigurati da izmenjene datoteke nose uočljiva obaveštenja da ste vi izmenili datoteke kao i datum bilo kakve izmene;

   b) morate osigurati da svako delo koje raspodeljujete ili izdajete, a koje u celini ili delom sadrži ili je izvedeno iz Programa ili bilo kog njegovog dela, bude licencirano u celini bez novčane nadoknade svim trećim licima pod odredbama ove licence;

   c) ako izmenjeni program čita naredbe interaktivno kada je pokrenut, vi morate osigurati da, kada je pokrenut u cilju takve interaktivne upotrebe na uobičajen način, ispiše ili prikaže objavu koja uključuje odgovarajuće saopštenje o autorskim pravima i saopštenje da ne postoji garancija (ili da vi dajete garanciju) da korisnici mogu raspodeliti program pod ovim uslovima, i objašnjenje korisniku kako da prikaže primerak ove licence. (Izuzetak: ako je Program interaktivan ali obično ne ispisuje takvu objavu, vaše delo zasnovano na Programu ne mora da ispiše objavu.)

   Ovi zahtevi se odnose na izmenjeno delo kao celinu. Ako uočljivi delovi takvog dela nisu izvedeni iz Programa, pa se mogu razumno shvatiti kao nezavisna i odvojena dela za sebe, onda se ova licenca i njene odredbe ne odnose na te delove kada ih raspodeljujete kao odvojena dela. Ali kada raspodeljujete iste delove kao deo celine koja je delo zasnovano na Programu, raspodela celine mora biti pod odredbama ove licence, čije dozvole za druge njene nosioce se proširuju na celinu, a samim tim na svaki deo bez obzira ko ga je napisao.

   Namera ovog člana nije da traži prava ili ospori vaša prava na delo koje ste u celini napisali; namera je prvenstveno da se uspostavi pravo na kontrolu raspodele izvedenih ili kolektivnih dela zasnovanih na Programu.

   Samo priključivanje Programu (ili delu zasnovanom na Programu) drugog dela koje na njemu nije zasnovano na jedinici za skladištenje ili medijumu za raspodelu ne dovodi drugo delo pod okvir ove licence.

3. Možete umnožavati i raspodeliti Program (ili delo zasnovano na njemu, po članu 2) u objektnom kodu ili izvršnom obliku pod gornjim odredbama članova 1 i 2 uz uslov da uradite jednu od sledećih stvari:

   a) da priložite uz njega potpuni odgovarajući mašinski čitljiv izvorni kod, koji mora biti raspodeljen prema gornjim odredbama članova 1 i 2 na medijumu uobičajenom za razmenu softvera; ili,

   b) da priložite uz njega pismenu ponudu, važeću bar tri godine, o dostavljanju bilo kojoj trećoj strani, uz novčanu nadoknadu ne veću od cene potrebne da obavite fizičku raspodelu izvora, potpunog mašinski čitljivog primerka odgovarajućeg izvornog koda, za raspodelu prema gornjim odredbama članova 1 i 2 na medijumu uobičajenom za razmenu softvera; ili,

   c) da priložite uz njega informaciju koju ste dobili kao ponudu o raspodeli odgovarajućeg izvornog koda. (Ova mogućnost je dozvoljena samo za

nekomercijalnu raspodelu i samo ako ste dobili program u objektnom kodu ili izvršnom vidu uz takvu ponudu, prema gornjoj tački b.)

Izvorni kod dela podrazumeva oblik dela najpogodniji za pravljenje izmena na njemu. Za delo u izvršnom obliku, potpuni izvorni kod podrazumeva sav izvorni kod za sve module koje ono sadrži, sa dodatkom svih datoteka koji definišu interakciju i spisa za kontrolu prevođenja i instaliranja izvršne verzije. Međutim, kao poseban izuzetak, raspodeljeni izvorni kod ne mora da uključi sve što se obično raspodeljuje (bilo u izvornom ili izvršnom obliku) sa glavnim komponentama (prevodiocem, jezgrom, i tako dalje) operativnog sistema na kom se pokreće izvršna datoteka, osim ukoliko se sama ta komponenta ne isporučuje sa izvršnom datotekom.

Ako se raspodela izvršnog ili objektnog koda vrši ponudom pristupa primerku sa određenog mesta, onda se ponuda istovetnog izvornog koda sa istog mesta računa kao raspodela izvornog koda, čak i u slučaju kada se od treće strane ne zahteva da umnožava izvorni kod zajedno sa objektnim kodom.

4. Program se ne sme umnožavati, menjati, podlicencirati ili raspodeliti drugačije nego što je izričito istaknuto ovom licencom. Svaki drugačiji pokušaj umnožavanja, menjanja, podlicenciranja ili raspodele programa nije punovažan i automatski obustavlja vaša prava pod ovom licencom. Međutim, stranama koje su od vas primile primerke ili prava pod ovom licencom neće se obustaviti njihove licence sve dok se one potpuno pridržavaju njenih odredbi.

5. Od vas se ne traži da prihvatite ovu licencu, pošto je niste potpisali. Međutim, ništa drugo vam ne dozvoljava da menjate ili raspodeljujete Program ili izvedena dela. Takve radnje su zabranjene zakonom ukoliko ne prihvatite ovu licencu. Tako, izmenom ili raspodelom Programa (ili dela zasnovanog na Programu), prihvatate da to uradite pod ovom licencom i pod svim njenim odredbama i uslovima za umnožavanje, raspodelu ili izmenu Programa ili dela zasnovanih na njemu.

6. Svaki put kada raspodeljujete Program (ili bilo koje delo zasnovano na Programu), primalac će automatski primiti licencu od prvobitnog nosioca za umnožavanje, raspodelu ili izmenu Programa prema ovim odredbama i uslovima. Vi ne možete dalje ograničiti primaočevo korišćenje ovde datih prava, kao što vi niste ni odgovorni ako treća strana nametne usklađivanje sa odredbama ove licence.

7. Ako su vam, kao posledica sudske presude ili pod izgovorom kršenja patenta ili bilo kakvog drugog razloga (ne ograničavajući se isključivo na pitanja patenata), nametnuti uslovi (bilo sudskim nalogom, poravnanjem ili drugačije) koji su u suprotnosti sa uslovima ove licence, oni vas ne izuzimaju od uslova ove licence. Ako je raspodela ili bilo koja druga vaša primenljiva obaveza u suprotnosti sa ovom licencom, onda kao posledicu toga ne možete uopšte raspodeliti Program. Na primer, ako patentna licenca ne dozvoljava besplatnu raspodelu Programa od strane svih primalaca primeraka ili indirektno preko vas, onda je jedini način da se zadovolji i ova licenca da se u potpunosti odreknete raspodele Programa.

   Ako se bilo koji deo ovog člana označi kao neodgovarajući ili neprimenljiv pod bilo kojom određenom okolnošću, primenjuje se ostatak člana a član kao celina se primenjuje pod drugim okolnostima.

Cilj ovog člana nije da vas navede da prekršite bilo koje patente ili druga prava na svojinu niti da ospori validnost takvih prava; ovaj član ima za svoj jedini cilj zaštitu integriteta sistema raspodele slobodnog softvera, koji je implementiran primenom javnih licenci. Mnogi ljudi su dali svoj nesebični doprinos u širokom spektru softvera raspodeljenog kroz ovaj sistem, oslanjajući se na njegovu doslednu primenu; na autoru/donatoru je da odluči da li je voljan/voljna da raspodeljuje softver kroz bilo koji sistem, a licenca ne može nametnuti taj izbor.

Ovaj član treba da detaljno razjasni ono što može biti posledica ostatka ove licence.

8. Ako su raspodela i/ili korišćenje Programa zabranjeni u određenim zemljama, bilo patentima ili autorskim pravima, prvobitni nosilac autorskih prava koji stavi Program pod ovu licencu može da priloži eksplicitno geografsko ograničenje raspodele, koje isključuje takve zemlje; tako da je raspodela dozvoljena samo u zemlji ili zemljama koje nisu isključene na takav način. U tom slučaju, ova licenca uključuje ograničenje kao da je deo njenog teksta.

9. Zadužbina za slobodni softver može povremeno objaviti revidirane i/ili nove verzije Opšte javne licence. Takve revizije će biti slične po duhu sadašnjoj verziji, ali se mogu razlikovati u detaljima u cilju razrešenja novih problema ili pitanja.

   Svaka verzija će dobiti različit broj. Ako Program ističe broj verzije Licence koja se primenjuje na njega i tekst „i bilo koja sledeća verzija“, možete primenjivati odredbe i uslove te ili bilo koje sledeće verzije koju objavi Zadužbina za slobodni softver. Ako Program ne ističe broj verzije ove licence, možete izabrati bilo koju verziju koju je objavila Zadužbina za slobodni softver.

10. Ako želite da upotrebite delove Programa u drugim slobodnim programima čiji su uslovi raspodele drugačiji, pišite autoru i zamolite ga za dozvolu. Za softver nad kojim autorska prava ima Zadužbina za slobodni softver, pišite Zadužbini za slobodni softver; mi ponekad načinimo izuzetke za ovakve stvari. Naša odluka će biti motivisana sa dva cilja: čuvanja slobodnog statusa svega što je izvedeno iz našeg slobodnog softvera i promovisanja deljenja i ponovne upotrebe softvera uopšte.

ODSUSTVO GARANCIJE

11. USLED LICENCIRANjA OVOG PROGRAMA BEZ NOVČANE NADOKNADE, NE POSTOJI GARANCIJA ZA PROGRAM U OKVIRU POSTOJEĆIH ZAKONA. AKO NIJE DRUGAČIJE NAPISANO, NOSIOCI AUTORSKIH PRAVA I/ILI DRUGA LICA NUDE PROGRAM „TAKAV KAKAV JE“ BEZ BILO KAKVE GARANCIJE, BILO EKSPLICITNE ILI IMPLICITNE, UKLjUČUJUĆI ALI SE NE OGRANIČAVAJUĆI NA IMPLICITNE GARANCIJE KOMERCIJALNE VREDNOSTI ILI ISPUNjAVANjA ODREĐENE POTREBE. CELOKUPAN RIZIK KVALITETA I PERFORMANSI JE NA VAMA. U SLUČAJU DA SE ISPOSTAVI DA JE PROGRAM DEFEKTAN, VI SNOSITE TROŠKOVE POTREBNOG SERVISIRANjA ILI POPRAVKE.

12. NI U KAKVIM OKOLNOSTIMA, IZUZEV AKO TO ZAHTEVA POSTOJEĆI ZAKON ILI PISMENI DOGOVOR, NOSILAC AUTORSKIH PRAVA ILI BILO KOJE DRUGO LICE KOJE MOŽE IZMENITI I/ILI RASPODELITI PROGRAM

UZ POŠTOVANjE PRETHODNE DOZVOLE, NEĆE VAM BITI ODGOVORNI ZA ŠTETE, KOJE OBUHVATAJU SVE OPŠTE, POSEBNE, SLUČAJNE ILI NAMERNE ŠTETE PROUZROKOVANE UPOTREBOM ILI NEMOGUĆNOŠĆU UPOTREBE PROGRAMA (UKLjUČUJUĆI ALI SE NE OGRANIČAVAJUĆI NA GUBITAK PODATAKA ILI POGREŠAN PRIKAZ PODATAKA ILI GUBITKE KOJE STE IZAZVALI VI ILI TREĆA LICA ILI NEMOGUĆNOST PROGRAMA DA FUNKCIONIŠE SA BILO KOJIM DRUGIM PROGRAMIMA), ČAK I AKO SU TAJ NOSILAC ILI DRUGA LICA BILI UPOZNATI SA MOGUĆNOŠĆU TAKVIH ŠTETA.

KRAJ ODREDBI I USLOVA

Kako da primenite ove odredbe na vaše nove programe

Ako razvijete novi program i želite da bude što korisniji javnosti, najbolji način da to postignete je da ga označite kao slobodni softver koji svako može raspodeliti i menjati pod ovim odredbama.

Da biste to uradili, dodajte sledeća obaveštenja vašem programu. Najsigurnije je da ih dodate na početak svake izvorne datoteke, da biste najefikasnije saopštili odsustvo garancije; svaka datoteka trebalo bi da sadrži najmanje liniju sa autorskim pravima i informaciju gde se može pronaći puno obaveštenje.

```
u jednoj liniji navedite naziv programa i kratak opis onoga šta
radi.
Autorska prava: Copyright (C) gggg ime autora

Ovaj program je slobodni softver; možete ga raspodeliti i/ili
menjati pod odredbama GNU-ove opšte javne licence koju je objavila
Zadužbina za slobodni softver; i to, bilo verzije 2 Licence, ili
(po vašem izboru) bilo koje sledeće verzije.

Ovaj program se raspodeljuje u nameri da bude koristan, ali BEZ
IKAKVE GARANCIJE; čak i bez implicitne garancije KOMERCIJALNE
VREDNOSTI ili ISPUNjAVANjA ODREĐENE POTREBE. Pogledajte GNU-ovu
opštu javnu licencu za više detalja.

Trebalo bi da primite primerak GNU-ove opšte javne licence zajedno
sa ovim programom; ako to nije slučaj, pišite Zadužbini za slobodni
softver na adresu:
Free Software Foundation, Inc., 59 Temple Place - Suite 330,
Boston, MA  02111-1307, USA.
```

Takođe dodajte obaveštenje kako vam se može javiti preko elektronske i obične pošte.

Ako je program interaktivan, treba da ispisuje kratko saopštenje slično ovom pri pokretanju u interaktivnom režimu:

```
Gnomovizija verzija 69, Autorska prava: Copyright (C) godina ime
autora
Gnomovizija se isporučuje BEZ IKAKVE GARANCIJE; za detalje
otkucajte 'prikaži g'.
Ovo je slobodni softver, a vi ste pozvani da ga raspodelite
pod izvesnim uslovima; otkucajte 'prikaži u' za detalje.
```

Hipotetičke naredbe ‘prikaži g’ i ‘prikaži u’ bi trebalo da prikažu odgovarajuće delove Opšte javne licence. Naravno, naredbe koje vi koristite mogu se razlikovati od ‘prikaži g’ i ‘prikaži u’; to čak mogu biti i klikovi mišem ili stavke menija ili nešto što najviše odgovara vašem programu.

Takođe bi trebalo da zatražite od vašeg poslodavca (ako ste zaposleni kao programer) ili vaše škole (ako ste u školi) da potpiše „objašnjenje autorskih prava“ za program, u slučaju da je to potrebno. Na primer (izmenite imena):

```
Jojodin, d.d., se ovim odriče svih autorskih prava za program
‘Gnomovizija’ (koji prolazi kroz prevodioce) koji je napisao
Petar Haker.

potpis Taj Kuna, 1. april 1989.
Taj Kun, predsednik poda
```

Ova opšta javna licenca ne dozvoljava uključenje vašeg programa u programe u privatnom vlasništvu. Ako je vaš program biblioteka rutina, može vam biti korisnije da dozvolite povezivanje vlasničkih aplikacija sa bibliotekom. Ako je to ono što želite, koristite GNU-ovu bibliotečku opštu javnu licencu (2) umesto ove licence.

C

LABORATORIJSKE VEŽBE

Ovaj dodatak je namenjen studentima Više elektrotehničke škole u Beogradu. Knjiga se u celini izlaže na laboratorijskim vežbama iz predmeta Operativni sistemi i predstavlja materijal za pripremu prvog dela ispita. Laboratorijske vežbe su organizovane u četiri tematske celine:

I. Sistemi datoteka

II. Vlasnički odnosi i prava pristupa

III. Rad sa datotekama iz komandne linije i uvod u shell programiranje

IV. Administracija UNIX i Linux sistema

Prvu tematsku celinu (sistemi datoteka) čine sledeće vežbe:

Vežba 1. Uvod u diskove i sisteme datoteka. Administracija blok uređaja.

Vežba 2. Administracija sistema datoteka - kreiranje, aktiviranje, provera i oporavak sistema datoteka.

Vežba 3. Rad sa diskovima bez sistema datoteka. Swap prostor. Značajni direktorijumi u aktivnom UNIX stablu.

Drugu tematsku celinu (vlasnički odnosi i prava pristupa) čine sledeće vežbe:

Vežba 4. Administracija korisnika i korisničkih grupa.

Vežba 5. Vlasnički odnosi i prava pristupa.

Treću tematsku celinu (rad sa datotekama iz komandne linije i uvod u shell programiranje) čine sledeće vežbe:

Vežba 6. Funkcije komandnog interpretera. Opšti pregled UNIX komandi. Dobijanje pomoći i informacija o sistemu.

Vežba 7. Kopiranje, pomeranje i brisanje datoteka. Linkovi. Rad sa direktorijumima.

Vežba 8. Rad sa tekstualnim datotekama. Tekst editori.

Vežba 9. Uvod u shell programiranje.

Četvrtu tematsku celinu (rad sa datotekama iz komandne linije i uvod u shell programiranje) čine sledeće vežbe:

Vežba 10. Arhiviranje podataka. Instalacija softverskih paketa.

Vežba 11. Administracija mrežnog okruženja.

Vežba 12. Rad sa štampačima. Administracija CUPS paketa.

Vežba 13. Administracija procesa. Podizanje i zaustavljanje sistema.

Vežba 14. Konfigurisanje jezgra. Disk kvote.

Vežba 15. Sigurnost UNIX i Linux operativnih sistema.

Ovaj dodatak sadrži kraći opis vežbi, kao pitanja i zadatke za proveru znanja studenata nakon svake odrađene vežbe.

Vežba 1

Uvod u diskove i sisteme datoteka. Administracija blok uređaja.

U okviru ove vežbe studenti se upoznaju sa osnovama administracije blok uređaja na UNIX i Linux sistemima. Vežba obuhvata:

- upoznavanje sa blok uređajima (diskovima, disketama, CD i DVD uređajima i magnetnim trakama),
- podelu diskova na particije,
- administraciju blok uređaja,
- upoznavanje sa sistema datoteka na UNIX i Linux sistemima.

Pitanja i zadaci

1. Koliko se uređaja može vezati na IDE, a koliko na SCSI kontroler?

2. U čemu je osnovna razlika u ponašanju između uređaja vezanih na isti kanal IDE kontrolera (na primer, Primary Master i Primary Slaver) i uređaja vezanih na različite kanale (na primer, Primary Master i Secondary Master).

3. Kako se određuju prioriteti uređaja vezanih na SCSI kontroler? Koji prioritet treba dodeliti sistemskom disku?

4. Objasniti čime su prouzrokovani limiti od 512MB i 8.4GB.

5. Kojom specijalnom datotekom je prestavljen IDE Primary Slave, a kojom četvrti SCSI disk?

6. Na jednom disku treba kreirati šest sistema datoteka. Koliko se najmanje logičkih particija mora kreirati, pod uslovom da se na disku više ne može kreirati ni jedna primarna particija?

7. Navedite dva programa kojima se mogu kreirati particije na disku.

8. Koju funkciju obavlja program fdformat?

9. Ako se razmatraju magnetne trake, šta predstavljaju rewind a šta non-rewind nodovi?

10. Kojim programom se proverava ispravnost površine magnetnog medijuma?

Vežba 2

Administracija sistema datoteka - kreiranje, aktiviranje, provera i oporavak sistema datoteka.

U okviru ove vežbe studenti se upoznaju sa osnovama administracije sistema datoteka na UNIX i Linux sistemima. Vežba obuhvata:

- kreiranje sistema datoteka,
- aktiviranje i deaktiviranje sistema datoteka,
- proveru ispravnosti i oporavak sistema datoteka od grešaka.

Pitanja i zadaci

11. Čemu služi program mkfs?

12. Navesti komandu za aktiviranje prve primarne particije četvrtog SCSI diska na direktorijum /data.

13. Šta se dešava sa trenutnim sadržajem direktorijuma koji nije prazan ukoliko se isti iskoristi kao mount-point za aktiviranje novog sistema datoteka?

14. Koji sistem datoteka se najpre aktivira prilikom podizanja UNIX operativnog sistema? Da li se taj sistem datoteka može deaktivirati?

15. Šta je opisano datotekama /etc/fstab i /etc/mtab?

16. Kako se svim korisnicima sistema može omogućiti da aktiviraju konkretan sistem datoteka (na primer, /dev/hda2)? Mogućnost davanja lozinke superusera je isključena).

17. Navesti dve varijante komande za deaktiviranje sistema datoteka koji se nalazi u prvoj logičkoj particiji Primary Master diska, a montiran je na direktorijum /data.

18. Zašto pri radu sa programom fsck sistem datoteka čiji se integritet proverava mora biti deaktiviran? Koja je funkcija lost+found direktorijuma?

19. U čemu je smisao tehnike vođenja dnevnika transakcija (journaling)? Da li se integritet journaling sistema datoteka proverava nakon nasilnog obaranja sistema?

20. Čemu služe programi df i du?

Vežba 3

Rad sa diskovima bez sistema datoteka. Swap prostor. Značajni direktorijumi u aktivnom UNIX stablu.

U okviru ove vežbe studenti se upoznaju sa radom sa diskovima bez sistema datoteka, virtuelnom memorijom, i značajnijim direktorijumima aktivnog UNIX stabla. Vežba obuhvata:

- rad sa programom dd,
- kreiranje i aktiviranje swap prostora u formi swap particije i datoteke,
- upoznavanje sa FHS standardom (Filesystem Hierarchy Standard v2.1.) i značajnim direktorijumima i datotekama koje se nalaze u aktivnom UNIX stablu.

Pitanja i zadaci

21. Navesti komandu kojom se prva dva bloka diska (master boot record) mogu zapisati na disketu (montirana na direktorijum /mnt/floppy) kao datoteka oldboot.rec.

22. Navesti komande za kreiranje i aktiviranje swap prostora u formi datoteke /swap_file veličine 250MB.

23. Navesti komande za kreiranje i aktiviranje swap prostora u formi particije /dev/hda3. Pretpostaviti da je particija veličine 250MB već kreirana programom fdisk.

24. Koji način realizacije swap prostora daje bolje performanse i zašto?

25. Objasniti sekvencu aktiviranja sistema datoteka i formiranja aktivnog UNIX stabla prilikom podizanja sistema.

26. Šta se nalazi na direktorijumu /dev?

27. Šta se nalazi na direktorijumima /etc i /var?

28. Šta se nalazi na direktorijumima /bin, /sbin, /usr/bin i /usr/sbin?

29. Šta je /proc sistem datoteka?

30. Kako se na realnom sistemu datoteka može snimiti slika operativne memorije (memory dump)?

Vežba 4

Administracija korisničkih naloga i grupa.

U okviru ove vežbe studenti se upoznaju sa korisničkim nalozima i grupama. Vežba obuhvata:

- upoznavanje sa pojmovima korisničkog naloga i grupe,
- upoznavanje sa značajnim datotekama (/etc/passwd, /etc/group, /etc/shadow),
- administraciju korisničkih naloga,
- identifikaciju korisnika i dobijanje informacija o korisničkom nalogu,
- privremeno prijavljivanje na sistem pod drugim imenom i određivanje stvarnog i efektivnog identifikatora korisnika (RUID i EUID).

Pitanja i zadaci

31. Objasniti razliku između sistemskih i regularnih korisnika. Koji sistemski korisnik može da se prijavi na sistem?

32. U kojoj su datoteci opisani svi korisnici sistema? Navesti polja koja opisuju jednog korisnika.

33. Koji korisnik može kreirati druge korisnike?

34. Šta je to primarna grupa korisnika i gde se definiše? Gde se definišu ostale grupe kojima korisnik pripada?

35. Šta je datoteka /etc/shadow i pomoću kog programa se može kreirati tako da bude funkcionalna?

36. Odakle se kopiraju datoteke koje predstavljaju inicijalno okruženje novokreiranog korisnika?

37. Koja datoteka predstavlja globalnu konfiguracionu datoteku za sve korisnike?

38. Zašto se datoteke /etc/passwd i /etc/group modifikuju komandama vipw i vigr, a ne, na primer običnim editorima teksta, kao što su joe ili jed?

39. U čemu je razlika između RUID i EUID. Šta prikazuje komanda id, a šta komanda whoami?

40. Na koji način se regularni korisnik može privremeno prijaviti na sistem kao root (pod predpostavkom da je već prijavljen na sistem kao regularan korisnik i da zna lozinku superuser-a)?

Vežba 5

Vlasnički odnosi i prava pristupa.

U okviru ove vežbe studenti se upoznaju sa kontrolom pristupa na nivou sistema datoteka. Vežba obuhvata:

- upoznavanje sa vlasničkim kategorijama (vlasnik, grupa i ostali korisnici) i pravima pristupa (pravo čitanja, upisa i modifikacije),
- određivanje pristupnih prava za različite vlasničke kategorije,
- promenu vlasništva i pristupnih prava,
- postavljanje specijalnih atributa datoteka (sticky bit, SUID, SGID),
- postavljanje specijalnih atributa datoteka na ext2 sistemu datoteka.

Pitanja i zadaci

41. Definisati vlasničke kategorije vlasnika, grupe i ostatka sveta.

42. Objasniti problem unije vlasničkih kategorija.

43. Šta znači pravo upisa u odnosu na datoteku?

44. Koja su prava neophodna korisniku da bi mogao da obriše datoteku i nad kojim objektom ta prava moraju biti definisana?

45. Šta znači pravo izvršavanja u odnosu na direktorijum?

46. Ko može promeniti prava pristupa datoteke ili direktorijuma?

47. Ko može promeniti vlasništvo datoteke ili direktorijuma?

48. Gde se čuvaju informacije o vlasniku, grupi i pristupnim pravima objekata sistema datoteka?

49. Šta su SUID i SGID flegovi?

50. Objasniti značenje specijalnog atributa i (immutable).

Vežba 6

Funkcije komandnog interpretera. Opšti pregled UNIX komandi. Dobijanje pomoći i informacija o sistemu.

U okviru ove vežbe studenti se upoznaju sa komandnim interpreterom i komandama koje se mogu iskoristiti za dobijanje pomoći. Vežba obuhvata:

- upoznavanje sa funkcijama komandnog interpretera,
- upoznavanje sa dodatnim mogućnostima karakterističnim za Bourne-again shell,
- uvođenje opšteg oblika komande,
- određivanje tipa datoteke,
- dobijanje pomoći i nekih informacija o sistemu.

Pitanja i zadaci

51. Navesti osnovne funkcije komandnog interpretera.

52. Šta je redirekcija, a šta pipe?

53. Čemu služi <Ctrl-D> kontrolni karakter u Bourne-again shellu?

54. Šta je alias i kako se definiše?

55. Kako se prilagođava odziv komandnog interpretera?

56. Koje su korisničke inicijalizacione datoteke i gde se nalaze?

57. Navesti tri načina za dobijanje pomoći.

58. Kako se može najjednostavnije pronaći lokacija neke komande, odnosno programa, u aktivnom UNIX stablu?

59. Da li UNIX prepoznaje datoteke na osnovu ekstenzija? Kako se može odrediti tip datoteke?

60. Čemu služi test magičnih brojeva?

Vežba 7

Rad sa datotekama i direktorijumima. Linkovi.

U okviru ove vežbe studenti se upoznaju sa komandama za rad sa datotekama i direktorijumima. Vežba obuhvata:

- kopiranje, pomeranje, promena imena i brisanje datoteka,
- kreiranje hard i simboličkih linkova,
- navigaciju po direktorijumskom stablu i rad sa direktorijumima.

Pitanja i zadaci

61. Navesti komandu za kopiranje kompletnog sadržaja direktorijuma /home u direktorijum /backup/home (tako da se datoteka /home/jsmith/1.txt kopira u /backup/home/jsmith/1.txt).

62. Šta UNIX radi kada korisnik zada sledeću komandu cp /tmp/a* /home/jsmith/b* ?

63. Navesti komandu za pomeranje grupe datoteka čije ime počinje dvocifrenim brojem sa direktorijuma /tmp na direktorijum /backup/tmp.

64. Koji su uslovi potrebni za kopiranje, a koji za pomeranje datoteke dat1 sa direktorijuma dir1 na direktorijum dir2?

65. Objasniti značenje promenljive umask? Koja prava pristupa ima kopija datoteke, ako su pristupna prava izvorišne datoteke 666, a promenljiva umask ima vrednost 027?

66. U čemu je razlika između hard i simboličkih linkova?

67. Da li korisnik može obrisati datoteku ako postoji hard link na nju? Da li korisnik može obrisati datoteku ako postoji simbolički link na nju?

68. Navesti sintaksu komande za brisanje direktorijuma /oldfiles koji nije prazan.

69. Navesti komandu kojom se traže sve datoteke koje pripadaju korisniku jsmith veličine najmanje 512 bajtova na celom aktivnom stablu.

70. U kom obliku treba zadati komandu ls da bi se na ekranu prikazale i skrivene datoteke?

Vežba 8

Rad sa tekstualnim datotekama. Tekst editori.

U okviru ove vežbe studenti se upoznaju sa osnovnim komandama za rad sa tekstualnim datotekama i značajnijim editorima teksta na UNIX i Linux sistemima. Vežba obuhvata:

- pregledanje sadržaja tekstualne datoteke,
- brojanje karaktera, reči i linija u datoteci,
- upoređivanje i uređivanje sadržaja datoteka,
- traženje teksta u datoteci,
- rad sa ekranskim editorima teksta vi, joe i jed.

Pitanja i zadaci

71. U čemu je razlika između komandi cat i less?

72. Navesti komande za prikazivanje prve tri i poslednje tri linije datoteke /etc/passwd.

73. Navesti komandu za brojanje reči i karaktera u datoteci /tmp/tekst1.

74. Navesti komandu koja pronalazi i na ekranu prikazuje sve razlike između datoteka /tmp/dat1 i /tmp/dat2, a pri tome ne uzima u obzir razlike između malih i velikih slova i suvišne blanko karaktere.

75. Navesti komandu koja uređuje datoteku /tmp/spisak1.txt po mesecima u drugoj koloni u opadajućem poretku, a rezultat upisuje u datoteku /tmp/spisak2.txt.

76. Navesti komandu kojom se u datoteku /tmp/spisak3.txt izdvajaju sve linije datoteke /tmp/spisak1.txt koje počinju nizom karaktera the ili The.

77. Navesti komandu kojom se u datoteku /tmp/spisak4.txt izdvajaju sve linije datoteke /tmp/spisak1.txt koje u sebi sadrže najmanje jedan od sledećih nizova karaktera: run1, run2, fun1, fun2.

78. Navesti komandu kojom se u datoteku /tmp/spisak4.txt izdvajaju sve linije datoteke /tmp/spisak1.txt koje sadrže jedno ili više uzastopnih pojavljivanja slova p.

79. Navesti sekvencu karaktera kojom se napušta vi editor, ako se korisnik koji u njemu radi trenutno nalazi u režimu za unos teksta.

80. Navesti sekvencu karaktera kojom se u vi editoru traži niz karaktera jsmith počev od tekuće pozicije do kraja datoteke.

Vežba 9

Uvod u shell programiranje.

U okviru ove vežbe studenti se upoznaju sa osnovama programiranja u Bash komandnom interpreteru. Vežba obuhvata:

- pripremu i pokretanje jednostavnog shell programa,
- korišćenje sistemskih i korisničkih promenljivih, argumenata i izlaznog statusa komande,
- korišćenje shell proširenja,
- rad sa uslovnim konstrukcijama, petljama i funkcijama.

Pitanja i zadaci

81. Navesti dve komande kojima se može pokrenuti shell program myscript iz tekućeg direktorijuma.

82. Čemu služi linija #!/bin/bash na početku shell programa?

83. Kako se na ekranu može prikazati vrednost promenljive PS1?

84. Navesti komandu kojom se podatak unet sa tastature upisuje u promenljivu prom1.

85. Kako se prenose argumenti sa komandne linije u shell program? Čemu služi promenljiva $# ?

86. Šta je izlazni status komande i čemu služi?

87. Objasnite upotrebu tilda proširenja. Šta radi komanda cd ~\doc ?

88. Šta na ekranu ispisuje sledeća komanda: echo $`who am i;pwd` ?

89. Neka je promenljivoj ime dodeljena vrednost jsmith. Šta na ekranu ispisuje sledeća komanda: echo $(($ime!=jsmith)) ?

90. U šta se proširuje sledeća komanda: $ mkdir /home/jsmith/{data,video,mp3} ?

Vežba 10

Arhiviranje podataka. Instalacija softverskih paketa.

U okviru ove vežbe studenti se upoznaju sa arhiviranjem i instalacijom softverskih paketa na Linux sistemima. Vežba obuhvata:

- upoznavanje sa pojmovima arhive i rezervne kopije podataka,
- rad sa programima tar i cpio,
- upoznavanje sa tar, RPM i deb paketima,
- instaliranje novih paketa, nadogradnju paketa novijim verzijama i uklanjanje postojećih paketa.

Pitanja i zadaci

91. U čemu je razlika između rezervne kopije podataka (backup) i arhive?

92. Šta radi komanda tar cvfz proba.tar a.a b.b c.c ?

93. Kako se može prikazati sadržaj arhive kreirane u prethodnom pitanju ?

94. Navesti komandu koja bezuslovno dodaje datoteku b.b u arhivu proba.tar. Gde se smešta datoteka b.b?

95. Objasnite režime rada komande cpio.

96. Navesti komandu koja programom cpio arhivira sve datoteke iz tekućeg direktorijuma u datoteku /tmp/arch1.

97. Šta tačno radi komanda find . –mount –print | cpio –pduvm /newdisk ? Pretpostaviti da je na direktorijum /mnt/newdisk montiran prazan sistem datoteka veličine veće od root sistema datoteka.

98. Navesti sintaksu komande rpm za instaliranje, uklanjanje i nadoradnju paketa.

99. Navesti sintaksu komande apt-get za instaliranje, uklanjanje i nadoradnju paketa.

100. Ukratko opisati postupak instalacije .tar.gz paketa.

Vežba 11

Administracija mrežnog okruženja.

U okviru ove vežbe studenti se upoznaju sa mrežnim okruženjem Linux sistema. Vežba obuhvata:

- upoznavanje sa mrežnim okruženjem, TCP/IP skupom protokola i IP adresiranjem,
- upoznavanje sa konfiguracionim datotekama, programima za administraciju TCP/IP skupa protokola i mrežnim servisima,

- administraciju mrežnog sistema datoteka i NIS servera,
- administraciju Apache web servera.

Pitanja i zadaci

101. Navesti primer IP adrese u klasi A, B i C. Šta predstavljaju adrese 127.0.0.1 i 255.255.255.255 ?

102. Šta je opisano datotekama /etc/services i /etc/protocols?

103. Šta znači linija " hosts: dns [!UNAVAIL=return] files" navedena u datoteci /etc/nssswitch.conf?

104. Čemu služi program ifconfig?

105. Pomoću kog programa se može proveriti da li je mreža preopterećena? Na osnovu čega se izvodi taj zaključak?

106. Šta su inetd i xinetd? Šta je vezivanje servisa za IP adresu, a šta redirekcija?

107. Šta je određeno linijom /share/doc -ro -access=ws1:ws2:ws3 u datoteci /etc/exports NFS servera?

108. Navesti komandu za montiranje NFS sistema datoteka /share/doc sa servera fserver1 na direktorijum /doc.

109. Koje sistemske datoteke ulaze u sastav NIS baze?

110. Navesti dva načina za pokretanje i zaustavljanje Apache web servera. Koji je od ta dva načina bolji i zašto?

Vežba 12

Rad sa štampačima. Administracija CUPS paketa.

U okviru ove vežbe studenti se upoznaju sa štampanjem pod Linux sistemom. Vežba obuhvata:

- upoznavanje sa štampačima, redovima za štampu i print serverima,
- upoznavanje sa CUPS programskim paketom,
- štampanje dokumenata iz komandne linije,
- administraciju reda za štampu.

Pitanja i zadaci

111. U čemu je razlika između lokalnih i udaljenih štampača? Šta su mrežni štampači?

112. Šta je red za štampač i u kom se direktorijumu obično nalazi? Šta predstavlja ime štampača?

113. Navesti osnovne komande za štampanje po System V i BSD standardu.

114. Šta prikazuje komanda lpq –Pnewprinter@nicotine ?

115. Šta radi komanda lprm –Pprinter1 –user jsmith ?

116. Navesti primer komande kojom se pokreće direktno štampanje redirekcijom izlaza.

117. Navesti komandu za dodavanje dodavanje HP DeskJet štampača koji je povezan na paralelni port računara.

118. Navesti komandu za brisanje štampača printer2.

119. Navesti komandu koja dozvoljava korisnicima user1 i user2 da štampaju na štampaču printer2, a svim ostalim zabranjuje da štampaju na tom štampaču.

120. Navesti komandu kojom se svim korisnicima sistema dozvoljava da na štampaču printer2 štampaju najviše 100 stranica nedeljno.

Vežba 13

Administracija procesa. Podizanje i zaustavljanje sistema.

U okviru ove vežbe studenti se upoznaju sa procesima na UNIX i Linux sistemima. Vežba obuhvata:

- osnovne tehnike upravljanja procesima,
- slanje signala procesima,
- upravljanje poslovima i prioritetima,
- zakazivanje i periodično izvršavanje komandi,
- upoznavanje sa procedurama podizanja i zaustavljanja sistema,
- upoznavanje sa procesom init i inicijalizacionim rc datotekama.

Pitanja i zadaci

121. Objasniti šta je daemon proces. Dati primer dva daemon procesa i objasniti šta predstavljaju.

122. Navesti komandu kojom se može najlakše odrediti PID procesa tuxracer.

123. U čemu je razlika između signala TERM i KILL.

124. Šta radi komanda nohup sort lista1 > lista2 & ?

125. Navesti komandu kojom se može promeniti prioritet procesa čiji je PID 1500 sa 20 na 30. Da li će se nakon toga proces brže ili sporije izvršavati?

126. Čemu služe komande fg i bg?

127. U čemu je osnovna razlika između korišćenja komande at i cron daemon procesa?

128. Navesti dva načina za konfigurisanje LILO boot loadera.

129. Šta je init? Kojim je nivoom izvršenja predstavljen jednokorisnički režim rada?

130. Šta su inicijalizacione rc datoteke? Kada se shell programi u rc direktorijumima izvršavaju?

Vežba 14

Konfigurisanje jezgra. Disk kvote.

U okviru ove vežbe studenti se upoznaju sa modulima i prevođenjem jezgra i disk kvotama. Vežba obuhvata:

- rad sa modulima jezgra,
- prevođenje jezgra,
- administraciju disk kvota.

Pitanja i zadaci

131. Šta su moduli jezgra i šta se njima omogućava?

132. Napisati komande za učitavanje i uklanjanje modula za podršku ReiserFS sistema datoteka.

133. Koji se problemi ne mogu rešiti modulima jezgra?

134. Kako se pokreće okruženje sa menijima iz koga se može konfigurisati izvorni kod novog jezgra?

135. Napisati komande za prevođenje izvornog koda jezgra Linux sistema.

136. Šta je /vmlinuz ?

137. Gde je implementirana podrška za disk kvote?

138. Šta se treba dodati u liniju LABEL=/home /home ext2 defaults,nosuid 1 2 datoteke /etc/fstab da bi kvote bile postavljene za korisnike, a šta za grupe?

139. U čemu je razlika između soft i hard limita? Šta je grace period?

140. Čemu služi program qoutacheck?

Vežba 15

Sigurnost UNIX i Linux sistema.

U okviru ove vežbe studenti se upoznaju sa osnovnim pojmovima vezanim za sigurnost UNIX i Linux sistema. Vežba obuhvata:

- primenu postupaka za povećanje opšte sigurnosti sistema,
- rad sa paketom GnuPG,
- administraciju iptables mrežne barijere.

Pitanja i zadaci

141. Navesti dve osnovne mere povećanja opšte sigurnosti sistema na nivou BIOS-a.

142. Čemu služe direktive password i restricted u datoteci /etc/lilo.conf?

143. Šta je opisano datotekom /etc/securetty?

144. Kako se može zabraniti korišćenje kombinacije tastera Ctrl-Alt-Del?

145. Kako se poruka digitalno potpisuje?

146. Šta radi komanda gpg -sear jsmith /backup/dat1?

147. Šta je demilitarizovana zona?

148. Čemu služi NAT tabela?

149. Čemu služi mangle tabela?

150. Kako se vrši filtriranje paketa kroz lance?

Student:	
Broj indeksa:	

Vežba	Naziv vežbe	Datum	Nastavnik/saradnik
1.	Uvod u diskove i sisteme datoteka. Administracija blok uređaja.		
2.	Administracija sistema datoteka - kreiranje, aktiviranje, provera i oporavak sistema datoteka.		
3.	Rad sa diskovima bez sistema datoteka. Swap prostor. Značajni direktorijumi u aktivnom UNIX stablu.		
4.	Administracija korisnika i korisničkih grupa.		
5.	Vlasnički odnosi i prava pristupa.		
6.	Funkcije komandnog interpretera. Opšti pregled UNIX komandi. Dobijanje pomoći i informacija o sistemu.		
7.	Kopiranje, pomeranje i brisanje datoteka. Linkovi. Rad sa direktorijumima.		
8.	Rad sa tekstualnim datotekama. Tekst editori.		
9.	Uvod u shell programiranje.		
10.	Arhiviranje podataka. Instalacija softverskih paketa.		
11.	Administracija mrežnog okruženja.		
12.	Rad sa štampačima. Administracija CUPS paketa.		
13.	Administracija procesa. Podizanje i zaustavljanje sistema.		
14.	Konfigurisanje jezgra. Disk kvote.		
15.	Sigurnost UNIX i Linux operativnih sistema.		

LITERATURA

[1] *The Linux System Administrator's Guide*

[2] *The Linux Network Administrator's Guide*

[3] G. Mourani, *Securing and Optimizing Linux: The Ultimate Solution*, Open Network Architecture Inc. (2001)

[4] A. Danesh, *Red Hat Linux*, Mikro knjiga (2000)

[5] S. Figgins, E. Siever, A. Weber, *Linux in a Nutshell, 4th Edition*, O'Reilly & Associates, Inc. (2003)

[6] W. Stanfield, R. D. Smith, *Linux System Administration, Second Edition*, Sybex (2001)

[7] T. Collings, K. Wall, *Red Hat Linux Networking & System Administration*, Hungry Minds Inc. (2002)

[8] R. W. Smith, *Linux+ Study Guide*, Sybex (2001)

[9] *UNIX Unleashed*, Sams Publishing (1994)

[10] S. Jovanović, *UNIX, uvod u sistem, mrežno okruženje i administraciju*, Institut za nuklearne nauke "Vinča" (1996)

[11] M. J. Bach, *The Design of the UNIX Operating System*, Prentice-Hall (1987)

[12] A. Silberschatz, P. B. Galvin, G. Gagne, *Operating System Concepts*, John Wiley & Sons, Inc. (2003)

[13] S. McClure, J. Scambray, G. Kurtz, *Sigurnost na mreži*, Kompjuter biblioteka (2001)

[14] B. Schneier, *Applied Cryptography, Second Edition*, John Wiley & Sons, Inc. (1996)