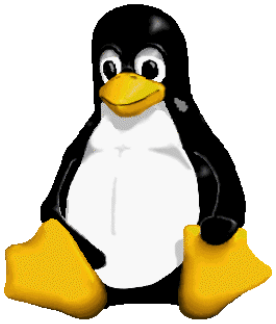
VEŽBA 2: LINUX

Linux je operativni sistem koji je nastao kao posledica hobija mladog studenta, Linusa Torvaldsa, na helsinškom univerzitetu u Finskoj. Linus je prvobitno bio zainteresovan za Minix, skromnu varijantu Unix operativnog sistema, pa je rešio da razvije sistem koji nadopunjuje nedostatke Minix-a. Počeo je sa radom 1991. godine kada je nastala verzija 0.02 i nastavio je predano da radi sve dok 1994. godine nije izašla verzija 1.0 Linux kernela. Kernel, samo srce operativnog sistema, je razvijen i objavljen pod GNU General Public Licence licencom. To znači da je, samim tim, source kod kernela dostupan svima. Imajući u vidu da je kernel zapravo jezgro operativnog sistema oko kojeg je razvijen ceo operativni sistem Linux, danas postoji na stotine i hiljade kompanija i organizacija, ali i jednak broj individualnih korisnika, koji su do sada objavili svoju verziju operativnog sistema baziranog na linux kernelu.

Osim toga što je Linux operativni sistem koji je besplatan, njegova funkcionalnost, adaptivnost i robustnost su ga učinili alternativnim operativnim sistemom Unix-u i svim Microsoft-ovim operativnim sistemima. IBM, Hewlett-Packard i mnogi drugi giganti u svetu računarstva su prihvatili Linux i maksimalno podržavaju njegov dalji razvoj. Danas, ne toliko od ulaska u treću deceniju svoga postojanja, Linux je postao operativni sistem koji je rasprostranjen širom sveta, pretežno kao serverska platforma. I pored toga, njegova upotreba kao operativnog sistema na personalnim računarima je u stalnom porastu i danas već i sa tog aspekta apsolutno je ravnopravan do skoro superiornom Microsoft-ovom Windows-u. Osim toga, Linux ima mogućnost implementacije na mikročipovima što omogućava nastajanje takozvanih „embedded“ sistema, čija upotreba postaje dominantna poslednjih godina.

Tokom 90-tih godina, Linux je uglavnom smatran za ništa drugo nego običan hobi mladih entuzijasta, koji nikada neće biti korišćen od strane velikog broja ostalih korisnika. Ipak, zahvaljujući projektima kao što su desktop okruženja KDE i GNOME, OpenOffice (kao alternativa Microsoft Office-u), Mozzila web browser, i mnoštvo drugih, danas postoji čitav spektar aplikacija koju omogućavaju korisniku da koristi Linux bez obzira na njegovo znanje Linux-a i računara uopšte.



Slika 1: Tux, oficijalna maskota Linuxa

Linux ima oficijalnu maskotu po kojoj je postao prepoznatljiv širom sveta. U pitanju je Tux, čuveni Linuxov pingvin, koji je odabran od strane Linusa Torvaldsa.

Ako se planira da računar na kome je instaliran Linux služi kao server, onda nema potrebe za instalacijom grafičkog desktopa. Razlog leži u činjenici da svaki grafički interfejs zahteve mnogo resursa, pre svega memorijskih, zatim procesorskih, pa na kraju krajeva prilično su zahtevni i što se tiče zauzeća sekundarne memorije (hard diskovi i sl.). Dakle, ako sistem ne zahteva monitor, instaliranje bilo kakvog grafičkog okruženja predstavlja samo razbacivanje resursa. Ipak, za sve druge sisteme, KDE i GNOME čine Linux pristupačnim svima. Oni omogućuju ono što prosečan korisnik i očekuje od svoga računara kao na primer:

- Prikaz mnogih različitih sadržaja automatski kada se klikne na odgovarajuću ikonicu, bez potrebe da se specificira program koji će se koristiti u tu svrhu;
- Cut i Paste operacije kako na tekstu tako i na slikama od jednog prozora do drugog, čak i kada su u tim prozorima pokrenute potpuno različite aplikacije koje koriste podatke u različitim formatima;
- Čuvanje i ponovo pokretanje sesija, tako da korisnik može prilično jednostavno da se ponovo uloguje na sistem i nastavi sa radom tamo gde je stao;
- Pomoć korisniku u obliku saveta, malih prikaza slika (thumbnails), kao i pomoći prilikom korišćenja alata;
- Omogučava široku lepezu lepih pozadina, screen-savera i tema;
- Dozvoljava u širokoj meri podešavanje parametara po ukusu, ali na suptilan način tako da većina korisnika ipak bude zadovoljna sa podrazumevanim (default-nim) podešavanjima.

I KDE i GNOME su dizajnirani tako da budu intuitivni i pozajmili su dosta ideja od drugih popularnih grafičkih okruženja, tako da je njihovo korišćenje logično i intuitivno za većinu korisnika. Ipak nabrojaćemo neke osnovne detalje i interesantne karakteristike oba okruženja koja bi možda bilo malo teže zapaziti svakodnevnim korišćenjem.

K Desktop Environment – KDE

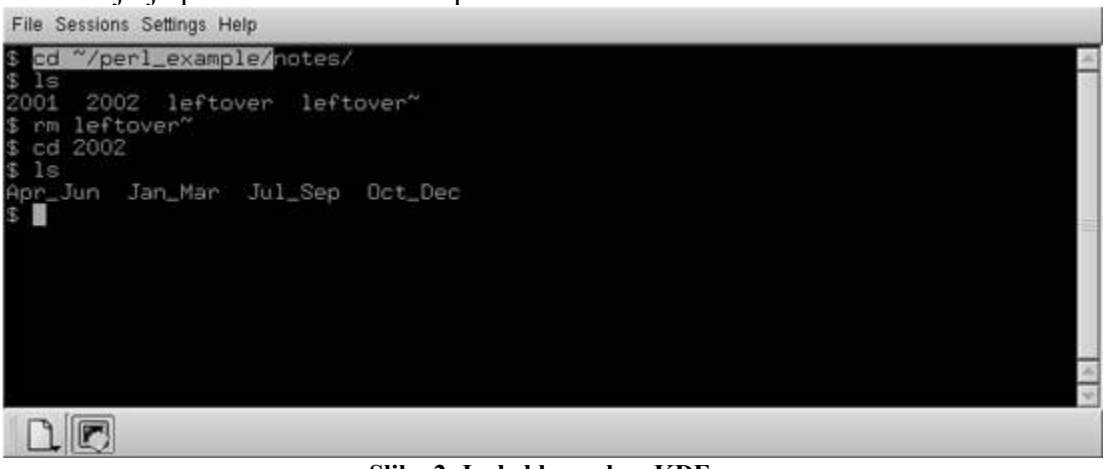
KDE je naravno open-source softverski projekat koji omogućava moderno, lako za korišćenje desktop okruženje za Unix i, samim tim, Linux sisteme. Od kada je počeo da se razvija, u oktobru 1996. godine, napravio je veliki napredak, najviše zahvaljujući izuzetno kvalitetnim GUI alatima od kojih je napravljen, kao i činjenicom da je za samu implementaciju korišćen C++ i njegove objektno-orijentisane metode.

KDE koristi tehnologiju poznatu kao Kparts koje omogućavaju da se jedna aplikacije umetne u drugu transparentno, pa tako, na primer, web pretraživač Konqueror može da prikazuje PDF dokumente unutar svojih „browser“ prozora koristeći pri tome program za otvaranje PDF dokumenata – KPDF. Pri tome, Konqueror ne mora da ima svoje komponente za prikaz PDF dokumenata. Isto važi i za KOffice pakete gde, na primer, tekst procesor KWord može da umeće tabele iz KSpread aplikacije (KWord odgovara – Microsoft Word-u, a KSpread Excelu).

Jedan od ciljeva razvojnog KDE tima jeste da se sve napravi konfigurabilno koristeći GUI dijaloge. Ispod konfiguracionog sistema se krije set tekstualnih fajlova sa prilično jednostavnim *parameter=value* formatom. Korisnik bi mogao njih da edituje, iako nikada nema potrebe za tim. Čak i najiskusniji korisnici priznaju da je za neke jednostavne stvari, kao što je na primer promena boje pozadine desktopa, brže i

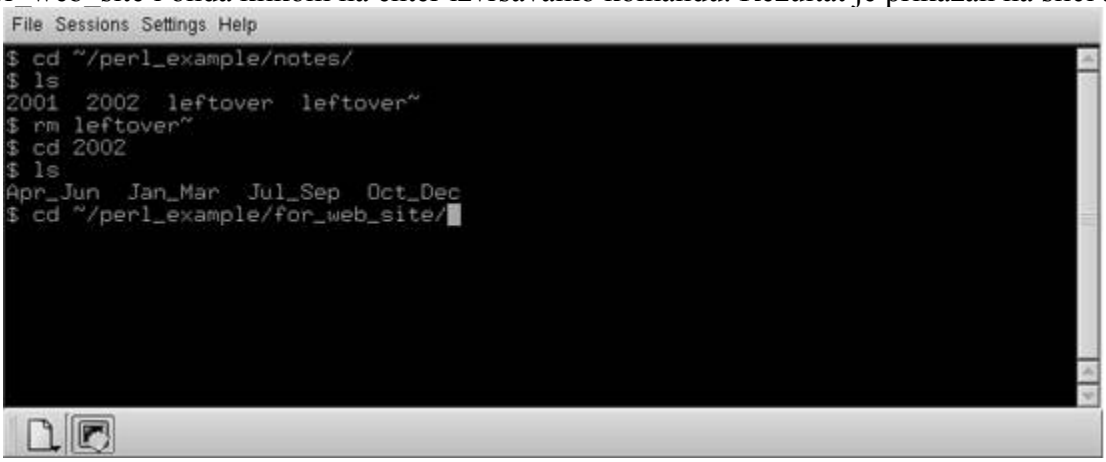
jednostavnije kliknuti nekoliko puta nego da se čita uputstvo, pronađe sintaksa koja definiše boju pozadine, otvara konfiguracioni fajl, edituje se i na kraju restartuje window manager.
Osim jednostavne konfiguracije, KDE prezentuje i dosta drugih metoda koje su ranije jednostavno bile strane za Linux operativni sistem. Na primer, KDE integriše pristup internetu direktno sa desktopa. Sve se to dešava putem fajl menadžera koji se ponaša kao web browser, ili se može to posmatrati i obrnuto. Pretraživanje fajlova preko određenih FTP sajtova je potpuno identično kao i pretraživanje fajlova na lokalnom hard-disku. Korisnik čak može da koristi drag-and-drop metodu i da ikonice koje predstavljaju linkove ostavlja po desktopu, te ih kasnije koristi. KDE čak integriše i alate za pretraživanje po internetu, kao i ostale internet resurse u desktop okruženje i čak dozvoljava korisniku da definiše svoje sopstvene omiljene internet pretraživače i da ih jednostavno koristi. Kao dodatak, skoro sve KDE aplikacije su u stanju da otvaraju fajlove i da ih čuvaju na udaljenim lokacijama, ne samo korišćenjem FTP, ili HTTP protokola, već i ka i od digitalnog fotoaparata, korišćenjem SSH enkripcije, ili na druge načine.
Sada ćemo samo još pomenuti neke od aplikacija koje se mogu pronaći u okviru KDE-a. Hiljade programa su dostupni za KDE. Neke od njih su osnovne aplikacije (kao što su konzola, terminal emulator, ...), zatim editori, alati za programiranje, igre i aplikacije za multimedijalne sadržaje. Ovde ćemo nabrojati samo neke od njih.

Konzola
Konzola je, ništa drugo nego prozor koju sadrži Unix shell. Na njemu je prikazan prompt, on prihvata komande, i može da se skroluje kao terminal. Na početku, Unix-ov terminal emulator se zvao *xterm*, ali je vremenom taj naziv uglavnom svuda promenjen u ***konzola*** (***console***). Na prvi pogled deluje ironično da najsavremeniji računari sa modernim i skupim monitorima, gigabajtima operativne memorije, i najsavremenijim grafičkim karticama, koriste softver koji emulira nekadašnji VT100 terminal. Ali, Linux je mnogo više od jednog operativnog sistema baziranog na principu point-and-click. Iako sadrži mnogo lepih grafičkih aplikacija, dosta vremena se provodi u obavljanju administrativnih poslova, i za takve poslove su command-line interfejsi neuporedivo moćniji alat. Kasnije ćemo detaljnije pričati o shell-u i skriptama.



Slika 2: Izgled konzole u KDE-u

Zapravo, konzola omogućava mnogo više nego običan terminal program. Jedna od moćnih performansi je cut-and-paste osobina. Ako pogledamo još jednom sliku 2 videćemo u prvom redu komandu koja otvara folder ~/perl_example/notes. Recimo da nismo hteli folder notes već umesto njega folder for_web_site. Prvo treba da odeberemo deo cd komande koja nas interesuje. Nakon toga, mišem dođemo sa leve strane od c u komandi cd. Pritisne se levi taster miša i povuče na desno sve dok se ne označi cela reč perl_example i karakter / koji sledi iza nje. Zapravo ovo je i prikazano na slici 2. Kada označena oblast obuhvata sve karaktere koji su nam od značaja, klikne se srednje dugme miša što će izazvati da se označeni karakteri prekopiraju u sledeću komandnu liniju. Sada je moguće samo dopuniti liniju sa direktorijumom koji zapravo želimo da otvorimo: for_web_site i onda klikom na enter izvršavamo komandu. Rezultat je prikazan na slici 3.



Slika 3: Kopirana izmenjena komanda

Na ovaj način može da se selektuje sve što se u prozoru nalazi kao ulaz ali i kao izlaz. Da bi se selektovala cela reč, u dvoklik levim mišem će rešiti problem. Ako treba kopirati ceo red treba tro-kliknuti levim dugmetom. Moguće je i selektovati više redova odjednom, ali ova komanda nije baš najzgodnija ako se radi sa komandama. Ipak, prilično je korisno kada se radi sa nekim tekst editorom (npr. vi) kada želimo da kopiramo veći deo teksta sa jednog prozora na drugi.

KGhostview

KGhostview je program koji se koristi za prikazivanje PostScript fajlova, kao i PDF fajlova. Adobe PostScript, je postao jedan od najpopularnijih formata za razmenu podataka u svetu. Korišćenje ovog programa je više nego jednostavno: poziva se zajedno sa imenom fajla koji želimo da otvorimo. Na primer:

```
$ kghostview article.ps
```

ili naravno klikom na ikonicu tipa ps ili pdf bilo gde unutar KDE okruženja.
KGhostview nije idealan za pregledanje PDF fajlova, iako je više nego dovoljan za veliki broj dokumenata. Ako ipak bude nekih problema sa gledanjem pdf fajlova, može se probati sa kpdf programom koji se nalazi u istom paketu programa zajedno sa KGhostview.

Konqueror

Konqueror nije samo prvoklasan web browser i fajl menadžer već služi i kao čitač dokumentacije. KDE dokumentacija je najčešće prikazana u HTML formatu ali Konqueror je u mogućnosti da prikazuje i druge formate kao što su Info i manpages. Na primer da bi se prikazao manpage za ls komandu, potrebno je samo otvoriti malu komandnu liniju pritiskom na Alt-F2 i kucanjem:

man:ls

KDE će prepoznati da korisnik želi da vidi manpages za komandu ls i otvoriće Konqueror prozor, nakon čega će prikazati manpage.
Na sličan način Konqueror može da koristi alate za pretraživanje po internetu. Ako, na primer, želimo da pretražimo stranice na internetu tražeći linuxovu maskotu, pri tome koristeći pretraživač AltaVista, uradićemo to na sledeći način:

av:tux

naravno, ovo radi i sa ostalim alatima za pretraživanje. Sledeća tabela prikazuje neke od najpopularnijih pretraživača sa svojim prefiksima.

Pretraživač	Prefiks
AltaVista	av:
SourceForge	sf:
Excite	ex:
Google	gg:
Meriam-Webster Dictionary	dict:

Još neki programi koji su popularni i karakteristični za KDE su:

- Kmail – KDEov standardni čitač pošte
- Kedit i Kwrite – standardni tekstualni editori. Kwrite je prefinjeniji ali ni jedan nije tako moćan kao Emacs
- Kpaint – je jednostavan grafički paket. Mnogo je primitivniji od GIMP-a, najpopularnijeg grafičkog programa na linux platformama
- Koffice – je kolekcija programa koja sadrži uobičajene office alate

GNU Network Object Model Environment – GNOME

GNOME desktop okruženje, kao i KDE, predstavlja kompletan paket: od pozadina destopa pa do aplikacija. Isto kao i KDE, GNOME može da pokrene svaku X aplikaciju. Zapravo, razlike između ova dva desktop okruženja su više vezana za programere koji razvijaju okruženja, nego za krajnje korisnike koji najčešće bez većih problema koriste aplikacije iz obe grupe.

Primarni cilj GNOME projekta jeste jednostavnost i lakoća upotrebe. Aplikacija mora da zadovolji brojne zahteve vezane za interfejs prema korisniku da bi postala deo oficijalne GNOME distribucije. Zbog činjenice da GNOME predstavlja odličnu platformu za razvoj u programskim jezicima kao što su C, C++, Python, Java, i C#, neoficijalne i nestandardne aplikacije su brojne.
Najveći broj Linux distribucija sadrži GNOME.
GNOME je dizajniran tako da bude jednostavan za korišćenje svima onima koji su ikada koristili računar. Iako izgled može da se menja na gotovo sve načine, standardna instalacija će sadržati desktop sa ikonicama na sebi i panelima sa gornje i donje strane. Ovi paneli spadaju među najvažnije GNOME alate jer su jako sadržajni i omogućavaju širok spektar interakcija sa sistemom. Oni mogu da budu pozicionirani samo duž jedne ivice ekrana, ili samo duž dela ivice ekrana, a mogu da sadrže dugmiće koji pokreću aplikacije, kao i male aplikacije koje se nazivaju applet-i među koje spadaju sat, sistem monitori, pa čak i manje igrice.
Nautilus – desktop i fajl menadžer

Nautilus je ime za GNOME desktop i fajl menadžer. On kontroliše prikaz slike sa pozadine, kao i fajlove sa desktopa, dozvoljava korisniku da koristi fajlove bez korišćenja konzole i vodi računa o obrisanim fajlovima umesto korisnika. Drugim rečima, GNOME je ekvivalent windows explorer-u, macintosh-ovom Finderu i KDE-ovom Konquerer-u. Kao i ove nabrojane aplikacije, Nautilus dozvoljava prevlačenje objekata sa jednog na drugo mesto. Neke prilično zgodne karakteristike Nautilusa su:

- Umesto prikazivanja ikonica za fajlove koji predstavljaju slike, prikazuje se umanjen thumbnail slike što se pokazuje kao veoma zgodno prilikom uređivanja direktorijuma sa slikama;
- Ako mišem korisnik zastane iznad mp3 fajlova, fajl se automatski pušta;
- Za tekstualne fajlove ikonica koja ga predstavlja se sastoji od stvarnog sadržaja fajla. Na ovaj način korisnik može da primeti da li je u pitanju pravi tekstualni fajl bez potrebe da ga prethodno otvori;
- Ako se desnim klikom odabere opcija Stretch Icon, moguće je uvećati ikonicu. Ako se dovoljno uveća može se videti kompletan sadržaj fajla, nešto slično kao desktop notepad;

GNOME zapravo sačinjavaju dva dela: prvi deo sačinjava samo desktop okruženje, vrlo atraktivno i lako za korišćenje i drugi deo koji predstavlja razvojno okruženje za pravljenje GUI aplikacija koje će moći da se integrišu u takvo grafičko desktop okruženje. Ceo GNOME je baziran na GTK+ (GIMP ToolKit-u). GTK+ je projekat koji je imao za cilj razvoj GIMP-a, ubedljivo najpopularnijeg programa za obradu slika na Linux platformama. Međutim, kasnije je od njega nastala biblioteka napisana u C programskom jeziku tako da je C, samim tim, normalno razvojno okruženje za GTK+ aplikacije. Vremenom je, dodavanjem odgovarajućih interfejsa, GTK+ postalo moguće koristiti i iz aplikacija pisanih u drugim programskim jezicima kao što su C++, Python, Java, pa čak i C#.
Neke popularnoje i češće korišćene aplikacije pod GNOME desktop okruženjem su:

- gEdit – GNOM-ov tekst editor koji je postao prilično popularan iako je relativno jednostavan i primitivan;
- Evolution – jedan od mnogih email paketa zasnovanih na GTK+;

- GIMP – najmoćniji bitmap grafički paket za Linux, čvrsto povezan sa GNOM-om;
- GNOM Office – predstavlja pokušaj da se kreira koherentna celina od grupe različitih kancelarijskih alata.

Za sada, dosta je priče o GNOME desktop okruženju. Ubuntu, Linux distribucija sa kojom ćemo raditi implementira GNOME razvojno okruženje tako da ćemo se sa njim detaljnije upoznati tokom ovih vežbi.

Sada ćemo se malo detaljnije upoznati sa konzolom, koja po mnogima predstavlja i najveću snagu Linux operativnih sistema.

Osnovne Linux Komande

Broj komandi na tipičnom Unix sistemu je dovoljan da popuni nekoliko stotina strana dokumentacije. Osim toga, moguće je i dodavati nove komande. Sada ćemo samo reći par reči o onim najpopularnijim i najčešće korišćenim.
Direktorijumi
Kao i kod Windowsa i bilo kojeg modernog računarskog sistema, pod Linuxom su fajlovi organizovani po hijerarhijskoj strukturi direktorijuma. Ne postoji zakonitost koja govori gde bi se koji fajl trebao nalaziti ali se vremenom pojavila određena konvencija na tu temu. Tako, na Linux platformama može se pronaći direktorijum koji se zove /home i gde se nalaze fajlovi svih korisnika koji rade na toj mašini. Pri tome svako korisnik ima svoj direktorijum unutar direktorijuma /home. Tako ako je korisnikovo ime za logovanje mdw svi njegovi fajlovi su smešteni unutar direktorijuma /home/mdw. Ovo se zove home direktorijum za tog korisnika. Naravno, unutar tog direktorijuma se mogu kreirati novi poddirektorijumi.

Za korisniki Windows operativnih sistema oznaka / može delovati prilično čudno i strano imajući u vidu da su oni verovatno navikli na \. Dakle, prva stvar na koju se treba navići jeste drugačija oznaka za prelazak na drugi nivo hijerarhije u organizaciji direktorijuma.
Gde se onda nalazi /home? Odgovor je, naravno, u direktorijumu koji se označava sa / i zove se root direktorijum. Kada se korisnik uloguje sistem ga stavlja u njegov home direktorijum. Da bi se to proverilo, koristi se komanda print working directory ili pwd:

```
$ pwd
/home/mdw
```

Sistem je na taj način potvrdio da se nalazimo u direktorijumu /home/mdw.
Svakako da je korisna komanda koja omogućava prelazak u neki drugi direktorijum. U tu svrhu se koristi komanda cd:

```
$ cd /usr/bin
$ pwd
/usr/bin
$ cd
```

Postavlja se pitanje gde se sada nalazimo? Ako se koristi komanda cd bez ikakvog argumenta sistem nas vraća u home direktorijum. Inače, home direktorijum se često označava sa ~. Dakle, string ~/programs znači da se programs nalazi unutar home direktorijuma. Da bi se kreirao direktorijum programs potrebno je unutar home direktorijuma otkucati:

$ mkdir programs

Ili sa bilo kog mesta:

$ mkdir /home/mdw/programs

Sada se možemo prebaciti u taj kreirani direktorijum:

$ cd programs
$ pwd
/home/mdw/programs

Rezervisana sekvenca karaktera .. odnosi se na direktirijum koji je neposredno iznad trenutno aktivnog u hijerarhiji. U našem slučaju, pošto se trenutno nalazimo u home/mdw/programs/ možemo se vratiti u home direktorijum tako što ćemo otkucati komandu:

$ cd ..

Suprotna komanda od mkdir je rmdir koja briše direktorijum:

$ rmdir programs

Slično tome, komanda rm briše fajlove. Nećemo je pokazati na primeru jer još nismo rekli ni kako se prave fajlovi. Najlakše se pravi fajl korišćenjem vi ili Emacs tekst editora, ali i neke od komandi koje ćemo nabrojati mogu da se koriste za kreiranje fajlova. Ako se specificira i opcija –r prilikom brisanja fajlova, rm briše ceo direktorijum sa svim svojim sadržajem, tako da treba biti prilično oprezan sa korišćenjem te opcije.

Do sada smo se već uverili da grafička desktop okruženja za Linux, kao što su KDE i GNOME, dolaze sa svojim fajl menadžerima, koji obavljaju dosta posla vezanog za rad sa fajlovima i direktorijumima, kao što su listanje i brisanje fajlova, pravljenje direktorijuma i tako dalje. Rekli smo već da neki od njih imaju integrisan web browser u sebe kao i neke druge izuzetno moćne alate. Ipak, kada treba izvršiti komandu na više fajlova, koji zadovoljavaju određene specifikacije, konzola i njena komandna linija se pokazuju kao ubedljivo najefikasniji alat, iako su prilično teže za učenje i savladavanje od GUI aplikacija. Na primer, ako treba obrisati sve fajlove u trenutnom direktorijumu ali i svim direktorijumima ispod koji počinju na slovo r i završavaju se sa .txt, u Z shell-u (zsh) bi se to uradilo na sledeći način:

```
$ rm **/r*.txt
```

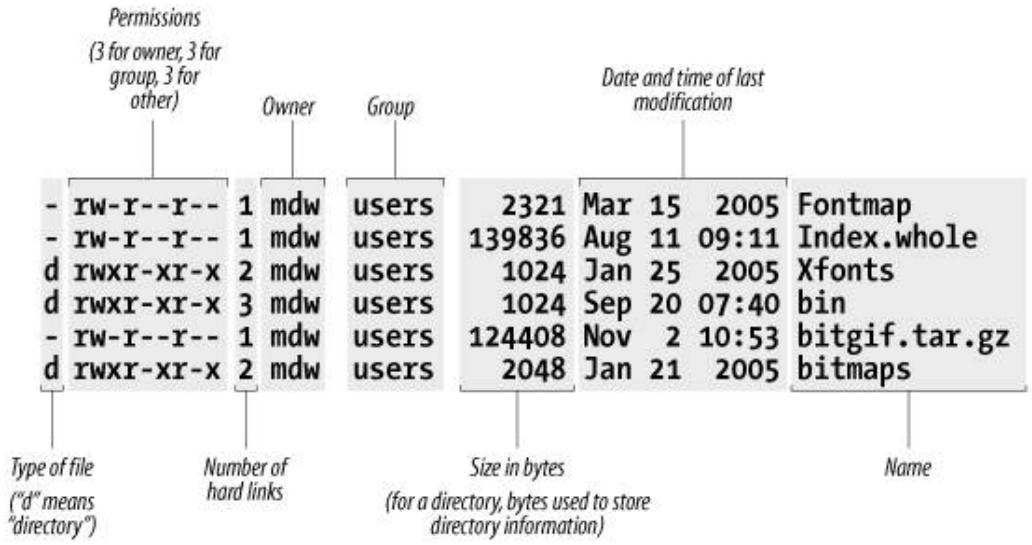
Više o ovim tehnikama skriptovanja će biti reči kasnije.

Za listanje direktorijuma se koristi komanda ls. Ako se samo pozove ls bez ikakvih parametara, izlistava se kompletan sadržaj direktorijuma. Najčešće se koristi ls sa argumentima:

```
$ ls /home
```

Kao i većina Linux komandi, i ls može da se kontroliše sa specijalnim opcijama koje počinju sa -. Treba samo obratiti pažnju da ispred – treba otkucati spacebar. Neke od korisnih opcija su:

- ls –a koja izlistava sve fajlove iz datog direktorijuma (čak i skrivene koji počinju sa tačkom);
- ls –l koja izlistava dodatne informacije o fajlovima i direktorijumima i
- ls –h koja izlistava fajlove sa veličinom koja je zaokružena i razumljivija korisniku



Slika 4: Prikaz komandom ls –l

Kada je potrebno pogledati sadržaj fajla to se može uraditi pozivanjem nekog tekst editora, npr. $ vi adrese.txt, ali ako se treba pregledati fajl na brzinu onda se to može uraditi komandom:

```
$ cat .bashrc
```

Međutim, veliki fajlovi će biti brzo skrolovani pa je onda korisnija komanda:

$ more .bashrc

Ona prikazuje ceo ekran i čeka korisnika da pritisne spacebar kako bi nastavio sa prikazom. More ima još korisnih opcija. Na primer, moguće je pretražiti fajl u potrazi za određenim stringom. Samo treba otkucati */string*, gde je *string* string koji se traži i pritisnuti enter.

Nekada korisnik ima potrebu da zadrži fajl na određenom mestu i da se pretvara kao da je na drugom. Ovo je najčešće korisno administratoru a ne samom korisniku. Na primer, programer može da ima nekoliko verzija programa prog.0.9, prog.1.1 itd, ali ime prog treba uvek da se odnosi na onaj program koji se trenutno koristi. Linux je za ovakve slučajeve rezervisao link-ove. Ovde ćemo spomenuti simboličke linkove, koji su najfleksibilniji i najpopularniji tip. Simbolički link je samo pokazivački fajl-fajl koji pokazuje na drugi fajl. Ako se edituje, čita ili izvršava simbolički link, sistem je dovoljno inteligentan da sve radnje obavlja na pravom fajlu. Simbolički linkovi su dosta slični shortcut-ima pod Windows operativnim sistemom, ali su mnogo moćniji.
Uzmimo za primer prog od malopre. Ako korisnik želi da napravi link prog na konkretan fajl, koji se zove prog.1.1 sve što treba da uradi jeste da pozove sledeću komandu:

$ ln –s prog.1.1 prog

Na ovaj način je kreiran novi fajl koji se zove prog koji je tzv. dummy fajl. Ako se on pokrene pokrenuće se u stvari prog.1.1. Ako pozovemo ls komandu sa fajlom prog dobićemo sledeći ispis:

$ ls –l prog
lrwxrwxrwx 2 mdw users 8 Nov 17 14:35 prog->prog.1.1

oznaka l na početku izlazne linije pokazuje da je fajl zapravo link, a poslednja strelica označava pravi fajl na koga pokazuje prog.

SHELLS

Kao što smo ranije već jednom spomenuli, logovanje na sistem preko konzole nas automatski dovodi u shell. Ako sistem omogućava grafički login onda nakon logovanja pokretanje terminal programa, tj. konzole dovodi korisnika u shell. Shell je program koji prihvata i izvršava korisničke komande. Postoji više različitih shell-ova koji se razlikuju ali su u suštini vrlo slični. Iako može da deluje zbunjujuće zašto postoji više tipova shell-a to bi trebalo shvatiti kao neku vrstu evolucije. Prvi razvijen shell za Linux je bio Bourne shell, koji bi danas bio izuzetni komplikovan i nepraktičan u poređenju sa shell-ovima koji su došli kasnije. Nabrojaćemo shell-ove koji su danas prisutni u Linux sistemima, tako da korisnik može da odabere shell koji njemu najviše odgovara.

- Bash – Bourne Again Shell je trenutno najviše korišćeni shell (ujedno i najmoćniji). Kompatibilan je sa Bournovim shell-om i kreiran i distribuiran u okviru GNU projekta.

- Csh – C shell. Razvijen na Berkeley-ju. Prilično je kompatibilan sa Bournovim shellom ali ima veoma različit interfejs za programiranje. Ne dozvoljava editovanje iz komandne linije.
- Ksh – Korn shell. Možda i najpopularniji shell na Unix sistemima generalno, i prvi koji je uveo moderno i shell tehnike, od kojih je neke nasledio od C shell-a. Kompatibilan je sa Bournovim shellom i omogućava editovanje iz komandne linije.
- Sh – Bourne shell, originalni shell. Dozvoljava editovanje iz komandne linije.
- Tsch – unapređeni C shell. Omogućava editovanje iz komandne linije.
- Zsh – Z shell. Najnoviji od svih shell-ova. Kompatibilan je sa Bournovim shellom i omogućava editovanje iz komandne linije.

Ako korisnik želi da proveri koji shell koristi trenutno to može da uradi sledećom komandom:

```
$ echo $SHELL
/bin/bash
```

Najveća je verovatnoća da je pokrenut upravo bash shell jer je on ubedljivo najpopularniji Linux shell.
Elementarna upotreba shell-a je prilično nedvosmislena: ukucate komandu i shell će odgovoriti izvršavanjem te komande. Međutim, da biste izvukli maksimum iz tekstualne interakcije sa Linuxom, morate da znate i naprednije funkcije shell-a. Na kraju krajeva, većinu interaktivnih shell funkcija čine one koje vam štede vreme, a neke vam čak omogućavaju da promenite izgled shell-a i nazive njegovih komandi.
Neki od trikova koji će vam svakako uštedeti vreme:

1) PARCIJALNO KUCANJE KOMANDI

Kucanje dugačkih komandi ili imena fajlova može da uspori operacije u komandnoj liniji, produžavajući vreme kucanja, kao i povećavajući verovatnoću da se pojavi greška u kucanju, dovodeći tako do neizvršenja komande ili nekih drugih neželjenih posledica. Zbog toga bash i neki drugi shell-ovi podržavaju funkciju poznatu kao završavanje komandi. Ukucajte prvih nekoliko slova komande ili imena fajla i pritisnite taster Tab. Shell će da locira sve komande na putanju ili imena fajlova koja počinju sa unetim karakterima. Ako samo jedna komanda ili ime fajla odgovara kriterijumu, shell će da ispiše ostatak komande, odnosno ime fajla. Ako više od jedne komande ili imena fajla odgovara kriterijumu, shell će da bipne, prikaže sve komande ili imena koja dolaze u obzir, ili i jedno i drugo u zavisnosti od konfiguracije shella. Ako nema nijednog fajla koji odgovara definisanim slovima, shell će samo da bipne.

2) PONOVNO POKRETANJE KOMANDI, ISTORIJA

Dok radite u shellu postoje velike verovatnoće da ćete hteti da primenite istu komandu, ili neku njenu varijantu, ponovo i ponovo. Na primer, možda ćete probati da instalirate paket, samo da biste otkrili da pre njega morate da instalirate drugi paket. Nakon što instalirate neophodni paket, želećete da instalirate prvi. U cilju bržeg izvršenja zadatka možete da upotrebite popularnu shell funkciju: istoriju.

Najmoderniji shellovi čuvaju informacije o nedavno otkucanim komandama. Možete da se krećete kroz ove komande tako što ćete pritiskati strelice tastera za gore i dole.

3) MODIFIKOVANJE KOMANDI

Shell-ovi poseduju jednostavne opcije za editovanje komandi koje kucate. Možete da upotrebite ove opcije za ispravljanje grešaka koje nastaju kucanjem komandi, ili možete da ih upotrebite za modifikovanje komandi koje ste našli pretraživanjem istorije, dodatno proširujući korisnost funkcije istorije. Najvažnije od ovih komandi su:

- Ctrl + A pomera kursor na početak linije
- Ctrl + E pomera kursor na kraj linije
- Esc + F pomera kursor za jednu reč u desno
- Esc + B pomera kursor za jednu reč u levo
- Ctrl + T menja mesto karaktera na kojem se nalazi kursor i prethodnog
- Esc + T menja mesto reči na kojoj se nalazi kursor i prethodne reči
- Ctrl + K briše od kursora do kraja linije
- Ctrl + X Backspace briše od kursora do početka linije

4) POKRETANJE VIŠE PROGRAMA: POZADINSKE OPERACIJE

Čak i u tekstualnom režimu rada, u Linuxu možete da pokrenete više programa. Jedna alatka kojom ovo možete da radite je ampersand operater (&). Dodajte ovaj karakter na kraj komande i Linux će pokrenuti program u pozadini; program će da radi, ali vi zadržavate kontrolu nad terminalom sa kojeg ste pokrenuli program. Naravno, ova akcija ima više smisla u nekim programima nego u drugim – pokretanje tekstualnog editora u pozadini baš i nema mnogo smisla, pošto morate da stupite u interakciju sa programom. Međutim, program koji samo obrađuje podatke i ne kreira nikakav konzolni rezultat, ima smisla pokrenuti u pozadini.

5) MANIPULISANJE SA ULAZOM I IZLAZOM: PREUSMERAVANJE

Linux programi koji rade u tekstualnom režimu rada rade na tri ulazno/izlazna toka, pri čemu se svaki od njih tretira kao fajl. Standardni input (stdin) je obično vezan za tastaturu i predstavlja metod pomoću kojeg program prima ulaz od korisnika. Standard output (stdout) je obično vezan za ekran, xterm prozor ili drugu alatku za prikaz u tekstualnom režimu rada i služi za prikazivanje normalnog programskog izlaza. Standard error (srderr) je takođe obično vezan za prikaz u tekstualnom režimu rada, ali je on zadužen za poruke visokog prioriteta, kao što su izveštaji o greškama. Linux može bez problema da preusmerava ove ulazne i izlazne tokove. To znači da korisnik može da uhvati izlaz programa koji radi u tekstualnom modu u fajl, ili može poslati programu sadržaj fajla u obliku ulaza. Ovo se radi koristeći operatere za preusmeravanje, koji su prikazani u tebeli koja sledi.

operater	akcija
<	preusmerava stdin da koristi specifikovani fajl
>	preusmerava stdout u specifikovani fajl, prepisujući postojeći sadržaj fajla
>>	preusmerava stdout u specifikovani fajl, pripajajući ga postojećem sadržaju fajla
2>	preusmerava stderr u specifikovani fajl, prepisujući postojeći sadržaj fajla
2>>	preusmerava stderr u specifikovani fajl, pripajajući ga postojećem sadržaju fajla
&>	preusmerava i stdout i stderr u specifikovani fajl, prepisujući postojeći sadržaj fajla

Svaki operater uzima ime fajla kao parametar. Možete da kombinujete veći broj operacija. Na primer, da biste sadržaj data.txt koristili kao ulaz za izvršnu datoteku *numcrunch*, a snimili izlaz programa u out.txt možete da ukucate:

$ numcrunch < data.txt >out.txt

6) KOMBINOVANJE KOMANDI: PIPES

Pipes nude način za povezivanje programa. Standardni izlaz prvog programa se preusmerava u drugi program, kao standardni ulaz. Lanac može da se nastavlja na onoliko programa koliko želite. Pipe operater je vertikalna crta (|), što znači da će povezana serija komandi izgledati ovako:

$ ps ax | grep gdm

Ovaj primer uzima izlaz od ps ax i „pipuje“ ga u grep, koji traga za linijama koje sadrže niz gdm. Upotrebom grepa na ovaj način možete da smanjite veličinu izlaza redundantnih programa. Možda ćete takođe želeti da „pipujete“ tekstualni izlaz kroz less pager, omogućavajući vam da pregledate izlaz brzinom koja vama odgovara.

SHELL SKRIPTOVI

I obični korisnici i sistem administratori mogu da koriste shell skriptove za lakši rad sa Linuxom. Shell skriptovi vam omogućavaju da kombinujete više programa u cilju kreiranja potencijalno mnogo složenije celine. Takođe, možete da upotrebite funkcije kao što su varijable, uslovni iskazi i petlje za kreiranje složenih programa. Većina Linuxovih sistemskih skriptova za startovanje su u stvari shell skriptovi; zbog toga, sposobnost razumevanja i modifikovanja shell skriptova je od suštinske važnosti za efikasnu administraciju linux sistema.

Shell skriptovi su jedna specifična podklasa interpretiranih programa, što znači da specijalni program (interpreter – u ovom slučaju shell program) čita fajl programa (shell skript) i implementira operacije koje on definiše svaki put kada se program pokrene. Ova operacija je u suprotnosti sa kompajliranim programima, u kojima kompajler čita fajl programa (izvorni kod) i generiše novi fajl (objektni kod ili binarni fajl) koji računar može da pokreće bez dalje pomoći kompajlera. Interpretirani programi, a samim tim i

shell skriptovi, se lakše razvijaju, pošto možete da napravite izmenu u programu i odmah ga pokrenete. Interpretirani programi se takođe mogu izvršavati na bilo kom procesoru, pod uslovom da postoju odgovarajući shell i da program ne zavisi od specifičnog hardvera, ili karakteristika procesora. Za razliku od njih, kompajlirani programi su brži, pošto kompajler može da generiše objektni kod koji je optimizovan za procesor. Međutim, dobijeni kod će se moći izvršavati samo na datom procesoru ili familiji procesora.
Shell skriptovi obično počinju linijom koja ih identifikuje kao takve. Ova identifikaciona linija počinje karakterom #, praćenim putanjom ka interpreteru (samom shell-u). Linuxov kernel zna da interpretira ove karaktere kao identifikaciju skripta i svi jezici skriptovanja koriste kardinalni znak (#) kao karakter komentara, zato shell ignoriše ovu liniju. Mnogi shell skriptovi se identifikuju kao da ih pokreće /bin/sh, ali neki specifikuju /bin/bash, ili neki drugi shell. Neki jednostavni skriptovi mogu da se izvršavaju u širokom varijetetu shellova, dok drugi koriste funkcije karakteristične za određene shellove. Na primer, bash i tcsh koriste različite metode za dodeljivanje vrednosti varijablama, što znači da će bilo koji skript koji ovo radi moći da se pokrene na jednom ili drugom, ali ne u oba. Da biste mogli da pokrenete shell skript, morate ili eksplicitno da ga usmerite ka shellu (kao u */bin/bash imeskripta*), ili da promenite dozvole na fajlu skripta, kako bi bio izvršni. Nakon toga možete da ga pokrenete isto onako kako pokrećete bilo koji drugi izvršni fajl, kucanjem njegovog imena (moguće sa celom putanjom ka fajlu pre njegovog imena). Da biste fajl napravili izvršnim traba da upotrebite chmod komandu:

$ chmod a+x imeskripta

Ova komanda dodeljuje svima (zbog slova a – all) pravo da izvršavaju (zbog slova x) skriptu koja se zove *imeskripta*.

Jedna od najosnovnijih upotreba shell skriptova jeste za pokretanje eksternih programa. To se radi tako što se postavlja ime eksternog programa, po mogućstvu sa kompletnom njegovom putanjom, pre imena, u skript, isto kao što biste ukucali komandu u shellovom promptu. Ako uključite ampersand (&), komanda će pokrenuti program u pozadini i procesiranje se nastavlja do sledeće komande, u suprotnom prva komanda se mora završiti pre nego što se izvrši sledeća. Takođe, možete da kombinujete programe u pipe, ili upotrebite preusmeravanje. Ako često uhvatite sebe kako kucate dugačke linije komandi, možda ima smisla da unesete tu komandu u skript i date joj kraće ime. Možete čak da upotrebite parametre poslate skriptu kao varijable, za upravljanje sa varijabilnim podacima koje komanda traži.
Mnoge linux komande su najkorisnije kada se pokreću u skriptama. Tabela koja sledi daje listu ovih komandi. Neke od ovih komandi su veoma složene, tako da bi trebalo da konsultujete stranice sa uputstvom za njih (man pages), za dodatne detalje. Naravno, možete da koristite i komande koje obično kucate u shellovom promptu, kao što su cd, cp, dd i tako dalje.

KOMANDA	EFEKAT
awk	jednostavni programski jezik koji daje rezultate na osnovu poklapanja patterna. Linux se isporučuje sa GNU awk-om ili gawk-om; on odgovara na oba imena
cut	uklanja delove liniji ulaza
echo	šalje komandu ili varijablu stdout-u
false	ne radi ništa, već vraća neuspešan kod
find	traga za fajlovima koji odgovaraju specifikovanom kriterijumu, kao što je ime fajla ili datum kreiranja.
grep	traga za fajlovima koji sadrže definisani niz. U normalnoj situaciji, šalje linije koje se poklapaju stdout-u.
pidof	vraća ID procesa (PID) koji odgovara specifikovanom aktivnom programu
read	traži ulaz od korisnika ili od fajla
sed	editor niza; edituje fajlove na osnovu komandi poslatih sed-u u komandnoj liniji, ili u skriptu
sort	sortira linije stdin-a, ili fajla i šalje sortiranu verziju stdout-u
uniq	ukljanja duplikate linija iz sortiranog fajla, ili iz stdin-a
true	ne radi ništa, već vraća uspešan kod

UPOTREBA VARIJABLI

Programi, uključujući shell skriptove, često moraju da rade sa podacima koji su nepoznati programeru. Na primer, možete da napišete skript za konvertovanje fajla sa podacima iz jednog formata u drugi, putem pozivanja serije pomoćnih programa za konverziju. Kada pišete skript, vi ne znate ime originalnog fajla, zato skript koristi jednu ili više varijabli koje će da drže ovu informaciju. Varijable okruženja su jedan tip varijabli koje možete da koristite i definišete u shell skriptovima. U bash-u vi definišete varijablu sa operatorom u obliku znaka jednakosti (=), kao što je ovde prikazano:

Varijabla = vrednost

Pazite da ne bude razmaka oko operatora znaka jednakosti. Vrednost može da bude alfanumerički niz, kao hitchhiker.txt ili numerička vrednost, kao broj 42. Ako postavite znake navodnika ono tekstualnog niza, možete da uključite i razmake u vrednost varijable.
Da biste koristili varijablu, treba da upotrebite ime varijable sa znakom dolara ($) ispred nje, kao kada biste koristili konstantnu vrednost. Na primer, listing koji sledi prikazuje skript koji manipuliše sa varijablom. Ovaj skript traži i prima ulaze (koristeći echo i read komande), a onda prikazuje rezultat manipulacije. Ova manipulacija šalje varijablu komandi cut koristeći pipe, a onda ekstrahuje drugo polje (-f2), onako kako je definisano sa razmacima u ulazu (-d’’ ’’):

```
#! /bin/bash
echo "Molim vas unesite tri reči razdvojene space-om: "
read inputline
```

```
echo -n "Druga reč je: "
echo $inputline | cut -d " " -f 2
```

ako se ovaj skript zapamti kao vardemo, pa potom napravi izvršnim (sa *chmod a+x vardemo*), pokrećemo ga i dobijamo rezultat:

```
$ ./vardemo
Molim vas unesite tri reči razdvojene space-om:
jedan dva tri
Druga reč je: dva
```

Takođe, mogu se koristiti varijable $1, $2 i tako dalje, kako bi se označili parametri poslati skriptu prilikom pokretanja programa. Na primer, ako ukucate ./vardemo *aparam*, $1 preuzima vrednost *aparam*. Možete da koristite ovu funkciju za lako slanje podataka skriptu. Na primer, ako hoćete da napišete skript koji će da konvertuje jedan format fajla u drugi, pomoću različitih pomoćnih programa za konverziju, možete da pošaljete ime originalnog fajla kao parametar i pozivate ga kao $1 kroz ceo skript. $0 varijabla se odnosi na ime koje ste koristili za pokretanje skripta, koje može biti korisno za štampanje poruka o greškama, ili za promenu ponašanja skripta, u zavisnosti od toga kako se poziva.

UPOTREBA USLOVNIH ISKAZA

Uslovni iskaz je iskaz koji određuje da li je nešto istinito ili neistinito. Na primer, iskaz može da testira da li fajl postoji, ili da li su dve varijable jednake. Ovi iskazi se nalaze unutar pravouglih zagrada ([]) i koristi ih nekoliko komandi koje ćemo sada opisati (if, while i until). Upotreba uslovnih iskaza omogućuje shell skriptovima da izvode akcije samo onda kada određeni uslov postoji, ili da izvode akcije više puta, sve dok završni uslov postoji. Tabela koja sledi daje pregled uslovnih iskaza. Pošto ovi iskazi nisu korisni osim u kombinaciji sa drugim komandama, specifični primeri njihove primene će biti prikazani u nastavku.

uslovni iskaz	značenje
-a file	istina ako fajl postoji
-b file	istina ako fajl postoji i ako je blok fajl
-c file	istina ako fajl postoji i ako je character fajl
-d file	istina ako fajl postoji i ako je direktorijum
-e file	istina ako fajl postoji
-f file	istina ako fajl postoji i ako je običan fajl
-r file	istina ako fajl postoji i ako se može pročitati
-s file	istina ako fajl postoji i ako je nije prazan
-w file	istina ako fajl postoji i ako se može pisati u njega
-x file	istina ako fajl postoji i ako se može izvršiti
-L file	istina ako fajl postoji i ako je simbolični link
-N file	istina ako fajl postoji i ako je modifikovan od njegovog poslednjeg pristupa
file1 –nt file2	istina ako file1 ima noviji datum modifikacije od file2

file1 –ot file2	istina ako file1 ima stariji datum modifikacije od file2
-o optname	istina ako je opcija optname omogućena u shell-u
-z string	istina ako je dužina stringa nula
-n string	istina ako dužina stringa nije nula
string1=string2	istina ako su stringovi jednaki
string1!=string2	istina ako stringovi nisu jednaki
string1<string2	istina ako string1 dolazi leksikografski pre stringa2
string1>string2	istina ako string1 dolazi leksikografski posle string2
arg1 –eq arg2	istina ako je arg1 jednak sa arg2. Oba moraju biti celi brojevi.
arg1 –ne arg2	istina ako arg1 nije jednak sa arg2
arg1 –lt arg2	istina ako je arg1 manji od arg2
arg1 –gt arg2	istina ako je arg1 veći od arg2
arg1 –le arg2	istina ako je arg1 manji ili jednak sa arg2
arg1 –ge arg2	istina ako je arg1 veći ili jednak sa arg2

UPOTREBE IF I CASE

Skriptovi često moraju da izvode različite akcije u zavisnosti od određenih uslova. Na primer, možda ćete hteti da skript prekine da se izvršava ako ne može da nađe fajl koji je korisnik specifikovao, ali da nastavi sa procesiranjem ako fajl postoji; ili ćete možda želeti da prikažete informaciju koja je prilagođena za svaki od nekoliko mogućih ulaza. If i case instrukcije su zadužene za ove situacije.

PRAVLJENJE BINARNIH IZBORA: IF

If je ključna reč i koristi uslovni iskaz da bi odredila da li da izvrši ili ne određenu akciju. Njena osnovna sintaksa je :

```
if [uslov]
then
        action
else
        other action
fi
```

možete da izostavite else klauzulu i other action ako ne želite da skript radi bilo šta drugo, u slučaju da uslovi nisu ispunjeni. Za primer korišćenja ključne reči if, pogledajte listing koji sledi. Ovaj skript proverava postojanje /proc/ide/ide0/hdb direktorijuma, koji bi trebalo da postoji samo ako postoji slave hard disk na priključenom ATA kontroleru. Ako taj direktorijum postoji, skript prikazuje informaciju na priključenom hard disku. Ako ne postoji, skript će vam reći da nema drajva.

```
#!/bin/bash
if [ -d /proc/ide/ide0/hdb]
then
        echo "The drive's model is:"
        cat /proc/ide/ide0/hdb/model
else
        echo "Sorry, there is no primary slave disk."
fi
```

Verovatno će identifikovani model drajva da varira od jednog do drugog, a u listingu koji sledi dat je primer jednog mogućeg izlaza programa.

$./demo (pod pretpostavkom da je skript sačuvan kao demo.sh)
The drive's model is:
Maxtor 91000DB

PRAVLJENJE VIŠE IZBORA: CASE

If možete da koristite za pravljenje binarnih odluka – da preduzmete akciju ako je uslov ispunjen ili neku drugu akciju ako uslov nije ispunjen. Takođe, možete da ugnjezdite if instrukcije jedne unutar drugih – ustvari, opciona elif instukcija je kreirana za tu situaciju: koristite je posle then ali pre else, kako biste izvršili drugi test. Međutim, ponekad je potreban fleksibilniji test višestrukih uslova. Tu case stupa na scenu. Vi šaljete varijablu ovoj instrukciji, a onda specifikujete seriju akcija koje će biti preduzete u zavisnosti od vrednosti varijable. Case sintaksa izgleda ovako:

```
case variable in
        vrednost1) komande1
                ;;
        vrednost2) komande2
                ;;
        [...]
esac
```

možete da uključite proizvoljan broj vrednosti koje se mogu poklopiti sa varijablom. Shell izvršava komande koje su asocirane sa prvom vrednošću koja se poklapa sa varijablom. Vrednosti koje određujete mogu da sadrže džokere, slično džokerima u fajlovima koji se poklapaju sa njima. Jedna od najkorisnijih vrednosti jeste ona koja se sastoji samo od jednog asteriska (*). Ova vrednost se poklapa sa bilo kojom varijablom, što znači da kada je ona poslednja opcija u case instrukciji, ona radi veoma slično kao else klauzula if instrukcije. Listing koji sledi prikazuje case u akciji. Ovaj skript pokreće jedan od tri programa kao odgovor na korisnikov ulaz. Kada korisnik pokrene ovaj skript, tekst definisan u prve dve echo linijese pojavljuje na ekranu i skript prima ulazno slovo. Ako je taj ulaz bilo šta osim e, l ili m, skript prikazuje standardnu poruku o grešci i izlazi; u suprotnom, skript pokreće specifikovani korisnički program.

```
#/bin/bash
echo "type the first letter of a program name:"
echo "emacs, lynx, or mutt"
read progletter
case $progletter in
        e) emacs
                ;;
        l) lynx
                ;;
        m) mutt
                ;;
        *) echo "you didn't type e, l or m; exiting.."
esac
```

UPOTREBA PETLJI

Jedna od alatki za skriptovanje je i petlja – način za izvršavanje istog koda ponovo i ponovo. Postoje dve široke klase petlji u shell skriptovima. For petlja će izvršiti kod fiksiran broj puta. Ovaj broj možda nećete znati kada budete pisali skript, ali se može definisati prilikom unošenja petlje. Suprotno od ovoga, while i until petlje se izvršavaju sve dotle dok se neki uslov ne ispuni.

FOR PETLJE
Osnovna sintaksa petlje je :

```
for variable in list
do
        commands
done
```

variable i commands su ono što možete da očekujete. List je varijabla ili izraz koji produkuje listu elemenata, najčešće komandi ili seta komandi zatvorenih unutar unazad okrenutih jednostrukih apostrofa (`), simbola koji se nalazi levo od tastera 1 na većini tastatura. Commands se izvršava jednom za svaki element u list-u, a variable uzima vrednost svakog elementa u list-u kod svakog prolaza kroz petlju. Na primer, u listingu koji sledi, umesto da se identifikuje samo jedan drajv, ovaj skript identifikuje sve ATA uređaje računara:

```
#/bin/bash
for drive in `ls –d /proc/ide/hd?`
do
        echo "The model of drive $drive is:"
        cat $drive/model
done
```

Izlaz, naravno, varira od broja i tipova hard diskova na računaru.

WHILE I UNTIL PETLJE
While i until petlje su veoma slične, ali se razlikuju po svojim uslovima za izlazak: while petlja se izvršava sve dok je specifikovan uslov istinit, until petlja se izvršava sve dotle dok je specifikovani uslov neistinit (to jest, sve dotle dok ne postane istinit). Iz ovoga sledi da ove dve vrste petlji zamenjuju jedna drugu, osim što morate da preokrenete prirodu uslovnih iskaza. Njihova sintaksa je takođe slična:

```
while [uslov]
do
        komande
done
```

ako se zameni while sa until dobija se until petlja. U listingu koji sledi predstavljena je varijanta već viđenog listinga. Kada se pokrene, ovaj skript stalno traži ulazno slovo.

Kada ukucate prihvatljiv odgovor, on pokreće traženi program ili izlazi. Kada napustite ciljni program, videćete drugi prompt za pokretanje programa. Ovaj skript bi mogao da se napiše kako sa while tako i sa until; ali uslovni iskaz bi koristio != umesto = za određivanje trenutka kada će izaći.

```
#/bin/bash
progletter=t
until[$progletter=q]
do
echo "type the first letter of a program name: "
echo "emacs, lyrics or mutt"
echo "(type 'q' to exit) "
read progletter
case $progletter in
        e) emacs
                ;;
        l) lynx
                ;;
        m) mutt
                ;;
        *) echo "you didn't type e, l or m; exiting.. "
esac
done
```

Dodatak

1) Korišćenjem *alias*-a moguće je napraviti prečice za komande koje često koristite. Npr:

alias logisim='java -jar /home/pedja/logisim/logisim-generic-2.7.1.jar'

omogućava jednostavno pokretanje java programa logisim-generic-2.7.1.jar koji inače mora da se poziva sa java -jar /putanja/do/programa. Korišćenjem alias-a logisim ovaj program se pokreće samo kucanjem *logisim* pod pretpostavkom da je ova gore linija dodata u ~/.bashrc konfiguracionu datoteku (koja se automatski učitava svaki put kada se pokreće shell)

Još jedan interesantan primer je definisanje alias-a za komandu koja radi cd .. (prelazi u direkorijum "iznad") više puta (npr umesto kucanja cd .. 6 puta, kucamo samo *updir* 6)

Da bi ovo bilo moguće, treba napraviti komandu updir:

alias updir='. ~/scripts/updir.sh'

Pri čemu je funkcionalnost komande definisana skriptom updir.sh (u mom slučaju ta skripta se nalazi u direktorijumu scripts unutar home direktorijuma).

Da bi sve ovo funkcionisalo kako treba, sadržaj shell skripte updir.sh je:

```
#!/bin/bash

UPDIR_LVL=$1 #parametar prosledjen skripti je broj koji nam treba

if [ $# -lt 1 ]; then #ako nije prosledjen broj kroz argument uzmi 1
 UPDIR_LVL=1
fi

UPDIR=""
for i in `seq 1 $UPDIR_LVL`; #i je ovde samo brojac 1..broj
do
 UPDIR="$UPDIR../"
done

cd $UPDIR #ako je broj 3 ovo ce biti cd ../../../ sto nam i treba
```

2) Shell skripte se korisne i za znatno složenije stvari. Prilično kratka skripta koja sledi radi pretragu i ispisuje direktorijume koji sadrže datoteke sa istim nazivom i iste veličine.

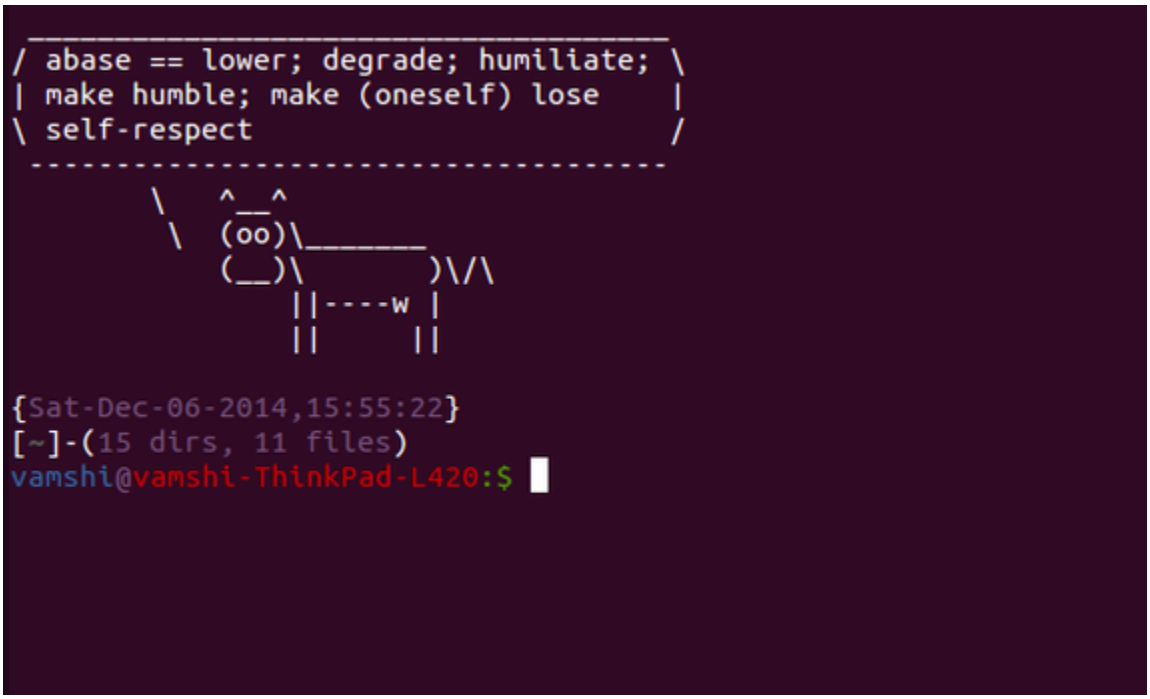
```
for d in `find $PWD -type d -print`; do
   echo `cd $d;ls -1ARs .|cksum|sed 's/ /_/'` $d;
done | awk '{c[$1]++; s[$1]=s[$1] " " $2} END {for (i in c) {if (c[i]>1)
print s[i]}}'
```

Ova skripta je korisna ako želite pronaći duplikate direktorijuma i sa ciljem je napravljena tako da vraća sve direktorijume koji sadrže "slične" datoteke koje ne moraju nužno biti identične (poredi datoteke po veličini, ne po sadržaju), da bi radila mnogo brže. Ako je potrebno proveriti da li su direktorijumi identični (sadrže identične datoteke), u tu svrhu se može koristiti skripta koja je znatno sporija jer proverava svaku pojedinačnu datoteku unutar direktorijuma (ako se ovo pokrene u home direktorijumu to može da potraje poprilično).

```
for d in `find $PWD -type d -print`; do
   echo `cd $d;cksum $(find . -type f -print)|cksum|sed 's/ /_/'` $d;
done|awk '{c[$1]++; s[$1]=s[$1] " " $2} END {for (i in c) {if (c[i]>1)
print s[i]}}'
```

Najbolji način korišćenja ove dve skripte je kada se najpre koristi prva da se dobije "predosećaj" direktorijuma koji su isti, nakon čega se na tim direktorijumima poziva druga skripta koja onda daje precizan rezultat.

3) Postoje takođe primeri gde su korišćeni neki iz "mora" shell komandi koje postoje kako bi se na interesantan način uradilo nešto korisno:



Na primeru iznad je iskorišćen program cowsay (:)) gde kravica izgovara string koji je prosleđen programu. Jedan student je to iskoristio da mu svaki put kada otvara konzolu kravica "izgovori" neki naizmenični red iz tekstualne datoteke koja sadrži čuvene GRE reči (netipične reči engleskog jezika koje se traže na GRE testu). Kaže da mu je ovo pomoglo da položi ispit.
Instalacija programa cowsay u Ubuntu Linux operativnom sistemu obavlja se iz konzole, na sledeći način:

```
sudo apt-get install cowsay
```

kao i većina programa koji se jednostavno instaliraju na UNIX-like sistemima.