ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU

PISANJE SKRIPTI U LINUKSU

– Diplomski rad –

Kandidat:

Darko Nitković 2010/413

Mentor:

doc. dr Zoran Čiča

Beograd, Novembar 2016.

SADRŽAJ

1. UVOD

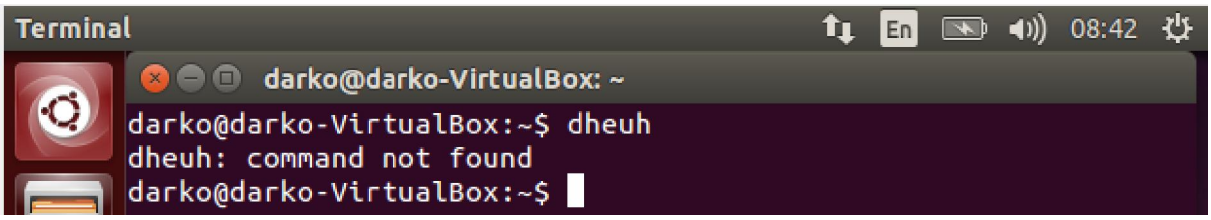
Cilj ovog rada je upoznavanje sa pisanjem komandnih skriptova u Linuks okruženju. Kako komandna linija podrazumijeva intenzivan rad sa datotekama i direktorijumima ova tehnika predstavlja veoma korisno sredstvo za automatizaciju posla. Dato je više konkretnih primjera koji treba da pomognu čitaocu da razumije skriptove i rad sa njima. Skriptovi realizovani u okviru ove teze biće priloženi u elektronskom obliku. Sam rad je organizovan u pet poglavlja.

Prvo poglavlje predstavlja uvod, i u njemu je dat kratak pregled predmeta samog rada, i kratko upoznavanje sa sadržajem ostalih poglavlja. Drugo poglavlje prezentuje ljusku *bash,* kao sredstvo interakcije između korisnika i operativnog sistema. U ovom poglavlju je predstavljen i emulator terminala kojim se pristupa ljusci. Treće poglavlje daje uvid u rad sa datotekama i direktorijumima. Objašnjena je hijerarhija, kao i osnovne komande za rad sa njima. U četvrtom poglavlju je opisano kreiranje komandnih skriptova, uz korišćenje komandi opisanih u trećem poglavlju. Peto poglavlje predstavlja zaključak u kom je dat pregled pozitivnih i negativnih strana ove tehnike.

2. LJUSKA

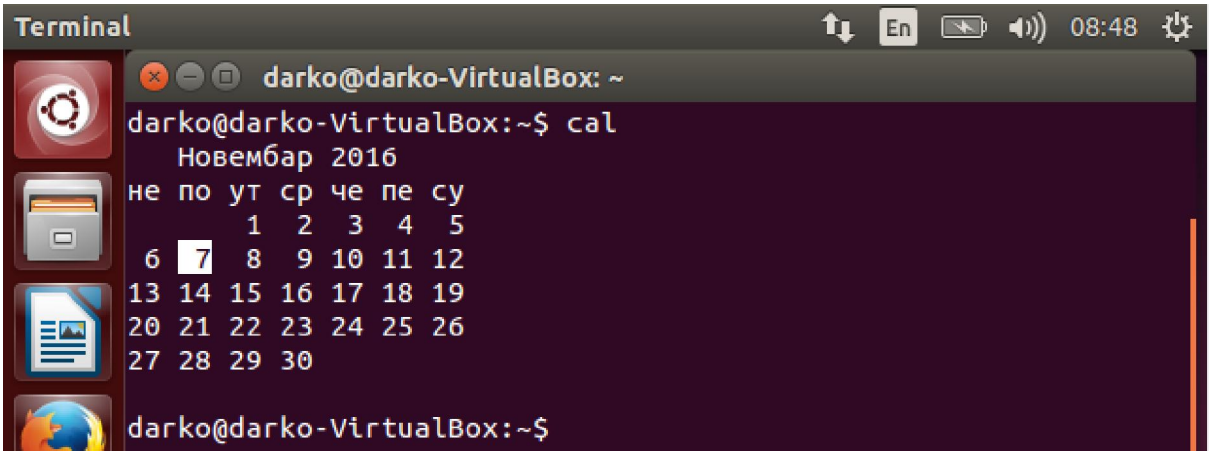
2.1. Pristup ljusci

Ljuska je program – interpreter komandne linije – koji preuzima komande sa tastature i prosleđuje ih operativnom sistemu na izvršavanje [1]. Ljuska koja je prisutna u većini Linuks distribucija je *bash*. *Bash* je skraćenica imena *Bourne Again Shell* i predstavlja unaprijeđenu verziju originalne *Unixove* ljuske *Bourne Shell* čiji je autor Stiv Burn. Za pristup ljusci iz grafičkog okruženja koristi se program koji se zove emulator terminala. Nakon pokretanja emulatora terminala dostupan je odzivnik ljuske koji se pojavljuje kada god je ljuska spremna da prihvati unos. U zavisnosti od distribucije operativnog sistema format odzivnika može da varira ali u najvećem broju slučajeva sadrži: korisničko ime, ime računara na kom je operativni sistem instaliran, tekući radni direktorijum korisnika i znak za dolar. Komande koje se unose počinju iza znaka za dolar i na taj način se razdvajaju od odzivnika. Unosom neke nepostojeće komande ljuska izdaje upozorenje i pruža mogućnost za ponovni unos.



Slika 2.1.1. - Primjer unosa nepostojeće komande

Komanda cal prikazuje kalendar za tekući mjesec.



Slika 2.1.2. - Rezultat komande cal

Ljuska pruža mogućnost kretanja kroz istoriju unijetih komandi. Pritiskom na taster strelica gore moguće je vidjeti prethodno unijetu komandu. Pritiskom na taster strelica dole prethodno

unijeta komanda nestaje. Na ovaj način moguće je pregledati i do 500 prethodno unijetih komandi što zavisi od odgovarajućih podešavanja u operativnom sitemu. Tekuća sesija završava se zatvaranjem prozora klikom na x ili unosom komande exit.

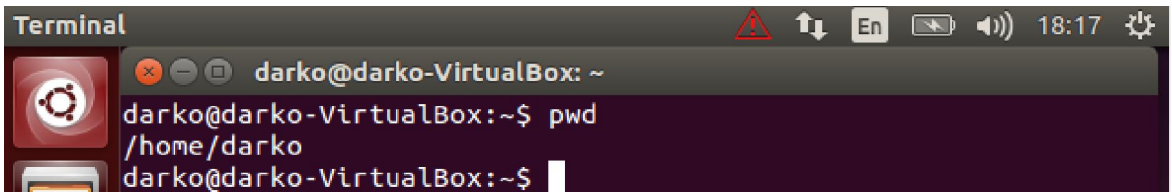
3. RAD SA DATOTEKAMA I DIREKTORIJUMIMA

3.1. Hijerarhija direktorijuma

Direktorijumi u Linuksu su organizovani u hijerarhijsku strukturu. Ovo podrazumijeva organizaciju direktorijuma u obliku stabla. Direktorijum koji je na samom vrhu ove strukture je korjenski. On se označava sa /. Unutar njega se nalaze direktorijumi u kojima su smješteni druge datoteke i direktorijumi. Korisnik u jednom trenutku može da se nalazi samo u jednom direktorijumu. Direktorijum u kome se korisnik trenutno nalazi naziva se tekući radni direktorijum. Direktorijum neposredno iznad je roditeljski direktorijum tekućem radnom direktorijumu.

3.2. pwd

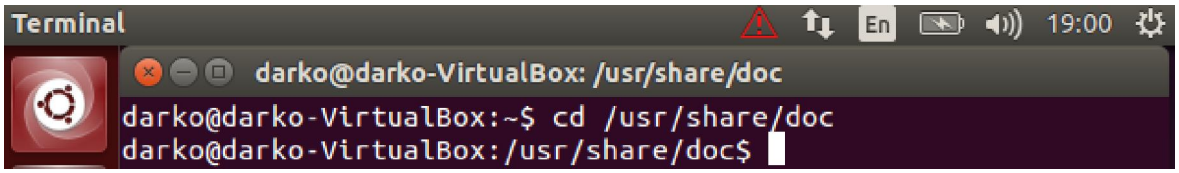
Za prikazivanje tekućeg radnog direktorijuma koristi se komanda pwd.



Slika 3.2.1. - Prikazivanje tekućeg radnog direktorijuma

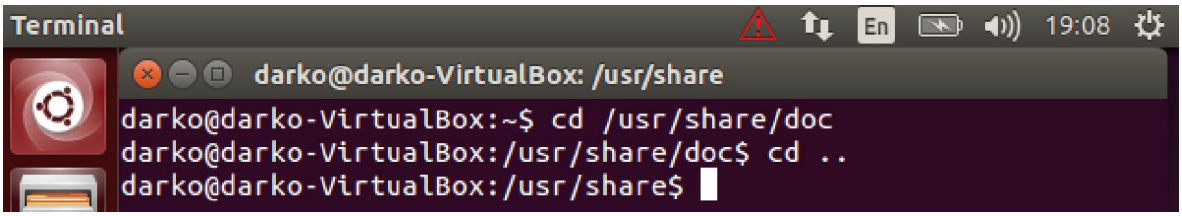
3.3. Apsolutne i relativne putanje

Referencirati neku datoteku ili direktorijum znači zadati putanju. Postoje dvije vrste putanja. To su apsolutne i relativne putanje. Kada putanja počinje znakom /, to je puna ili apsolutna putanja [2]. Putanja koja ne počinje ovim znakom je relativna. Za promjenu tekućeg radnog direktorijuma koristi se komanda cd. Postavljanje direktorijuma /usr/share/doc za tekući radni prikazano je na slici 3.3.1.. Ovim postupkom je iskorišćeno zadavanje apsolutne putanje. Kreće se od korjenskog direktorijuma. Unutar njega se nalazi folder usr. Unutar ovog foldera nalazi se folder share. Unutar foldera share je folder doc u kome se nalaze razne dokumentacione datoteke.



Slika 3.3.1. - Zadavanje apsolutne putanje

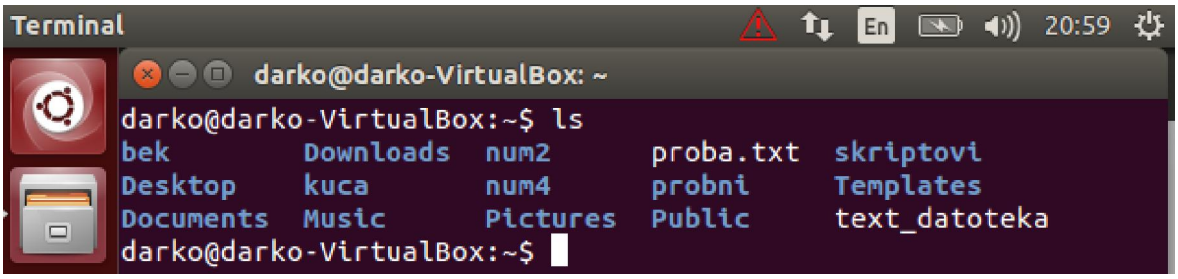
Relativne putanje koriste znakove . i .. . Znak . predstavlja tekući radni direktorijum. Znak .. predstavlja roditeljski direktorijum tekućeg radnog direktorijuma. Za prelazak u roditeljski direktorijum tekućeg radnog direktorijuma koristi se naredba cd .. . Za prelazak iz direktorijuma /usr/share u /usr/share/doc unosi se cd doc jer je /usr/share/doc poddirektorijum direktorijuma /usr/share, u tom slučaju je ovo moguće, jer se radi o poddirektorijumu tekućeg radnog.



Slika 3.3.2. - Primjer upotrebe relativne putanje

3.4. ls

Ova komanda u svom osnovnom obliku služi za izlistavanje direktorijuma. Nije nužno da komanda ls izlistava sadržaj trenutnog direktorijuma. Komanda se može koristiti za izlistavanje bilo kog direktorijuma. Prikazani rezultat moguće je modifikovati dodavanjem opcija. Upotrebom ls –F /etc dodaje se indikatorski znak pored svakog imena (npr. / za direktorijum) . Konstrukcija ls –a /dev izlistava sve datoteke direktorijuma /dev uključujući i skrivene koje počinju tačkom.



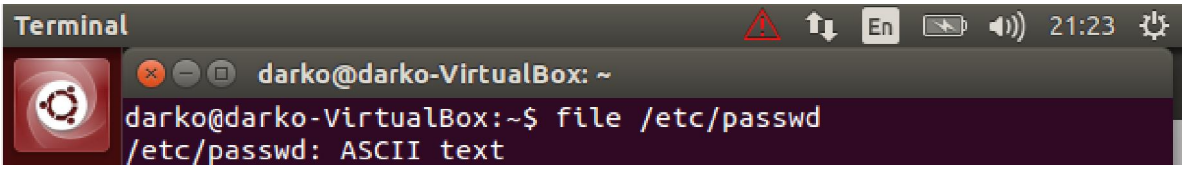
Slika 3.4.1. - Izlistavanje tekućeg radnog direktorijuma

Slika 3.4.2. – Izlistavanje direktorijuma /etc uz dodavanje indikatorskog znaka

Slika 3.4.3. – Izlistavanje direktorijuma /dev uključujući i skrivene datoteke

3.5. file

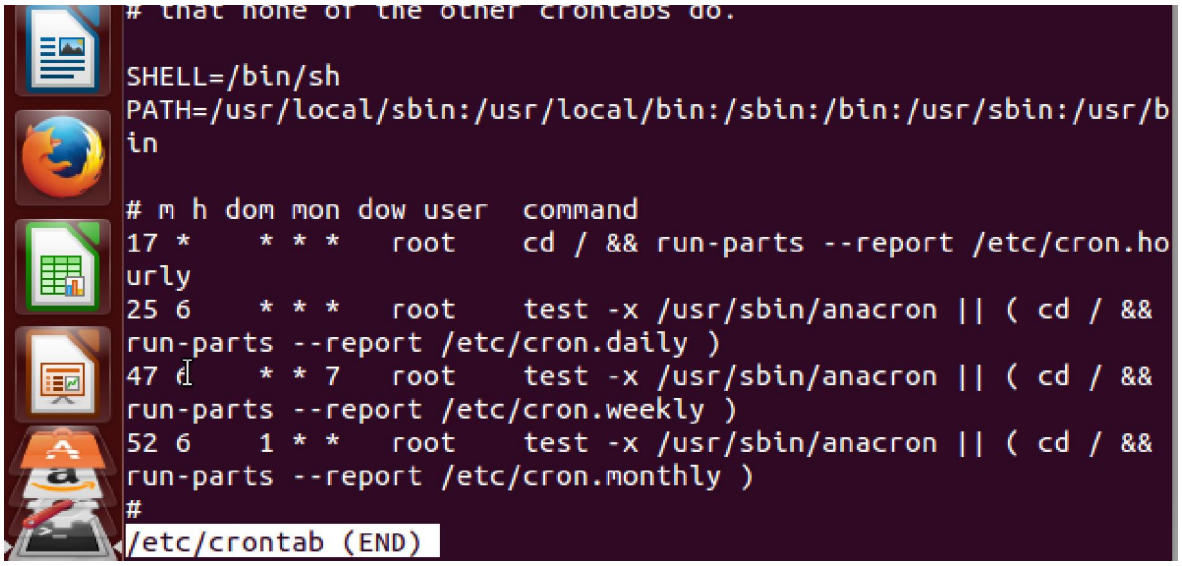
Komanda file se koristi za određivanje tipa datoteke. Unošenjem file *datoteka* određuje se tip datoteke *datoteka.* Rezultat komande file /etc/passwd pokazuje da je datoteka u kojoj se čuva lista korisničkih naloga tekstualnog tipa.



Slika 3.5.1. – Određivanje tipa datoteke /etc/passwd

3.6. less

Komanda se koristi za pregledanje sadržaja tekstualnih datoteka. Formata je: less *datoteka*. Mnoge konfiguracione datoteke su tekstualnog tipa. Skriptovi su takođe datoteke tekstualnog tipa. Zbog toga je ova komanda korisna jer pruža uvid u sadržaj navedenih datoteka. Naredbom less /etc/crontab prikazuje se sadržaj datoteke u kojoj je definisano kada će se izvršavati automatski poslovi.

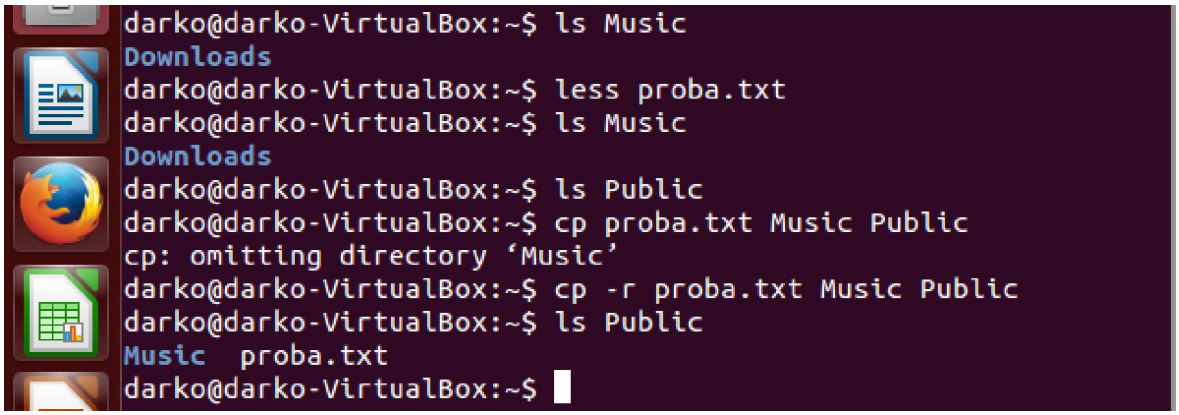


Slika 3.6.1. – Primjer upotrebe komande less

Moguće je kretanje kroz prikazane rezultate ako postoji više stranica. Taster page up vrši pomjeranje za jednu stranu unazad. Taster page down vrši pomjeranje za jednu stranu unaprijed. Strelica na gore vrši pomjeranje za jedan red nagore. Strelica nadole vrši pomjeranje za jedan red nadole. Taster G pomjera na kraj tekstualne datoteke. Izlazak iz rezultata prikazanih na ekranu vrši se pritiskom tastera Q.

3.7. cp

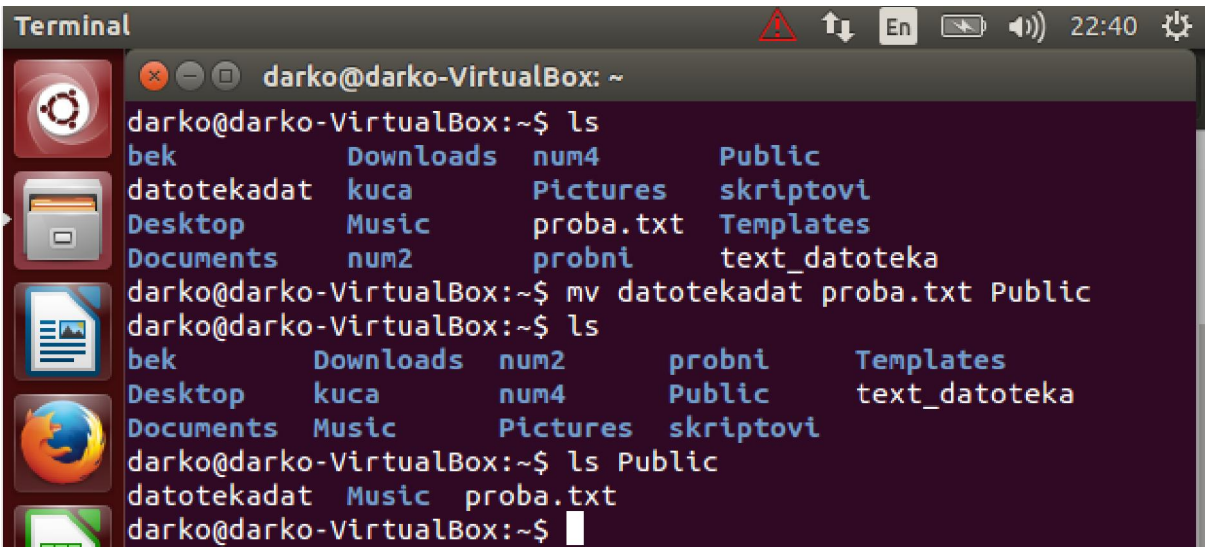
U svom osnovnom obliku vrši kopiranje datoteka. Za kopiranje datoteke1 u datoteku2 potrebno je unijeti sledeće : cp datoteka1 datoteka2. Ovom naredbom moguće je kopirati više stavki u jedan direktorijum. Unošenjem cp -r proba.txt Music Public vršimo kopiranje datoteke proba.txt i direktorijuma Music u prazan direktorijum Public. Kada se vrši kopiranje direktorijuma potrebno je zadati opciju –r jer time vršimo rekurzivno kopiranje. To znači da se vrši i kopiranje sadržaja datog direktorijuma.



Slika 3.7.1. – Primjer upotrebe komande cp

3.8. mv

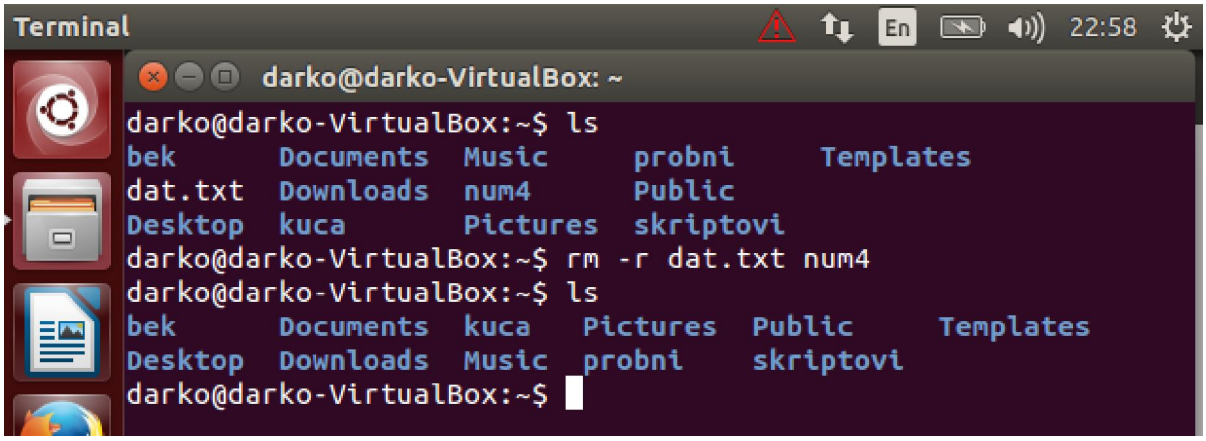
Ova komanda omogućava preimenovanje ili premještanje datoteka u zavisnosti za šta se koristi. Unosom mv dat1 dat2 vrši se preimenovanje datoteke imena dat1 u datoteku dat2, ukoliko dat2 ne postoji. Može se koristiti i za premještanje više datoteka u direktorijum. Ako se žele premjestiti datoteke datotekadat i proba.txt u direktorijum Public onda se to vrši unosom naredbe: mv datotekadat proba.txt Public.



Slika 3.8.1. – Primjer upotrebe komande mv

3.9. rm

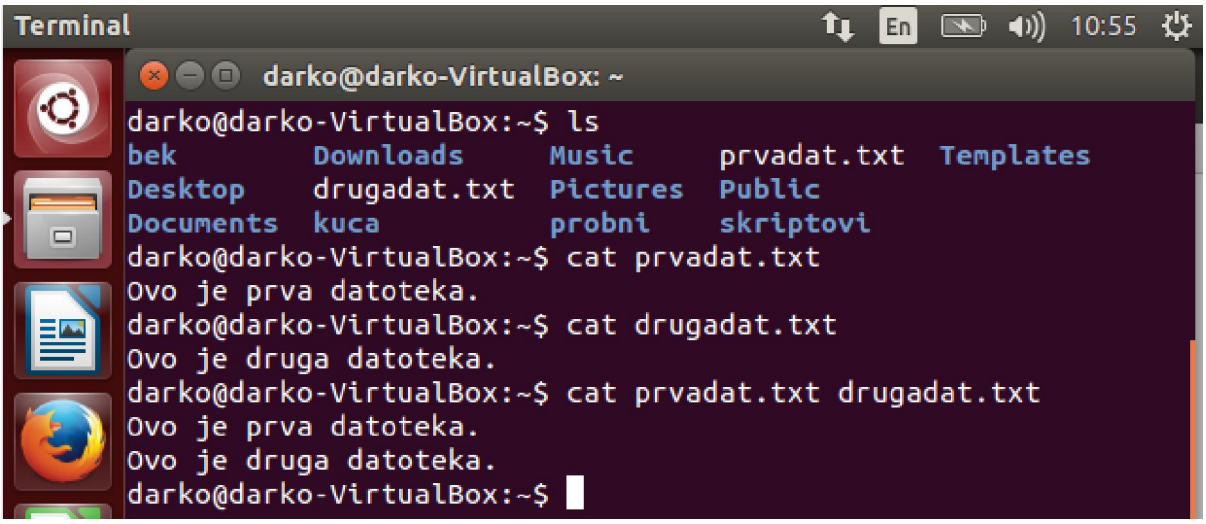
Naredba rm služi za uklanjanje datoteka i direktorijuma. Uklanjanje datoteke dat.txt i direktorijuma num2 vrši se unošenjem rm –r dat.txt num4. Kada se uklanja direktorijum potrebno je unijeti –r. U suprotnom, uklanjanje neće biti uspješno.



Slika 3.9.1. – Primjer upotrebe komande rm

3.10. cat

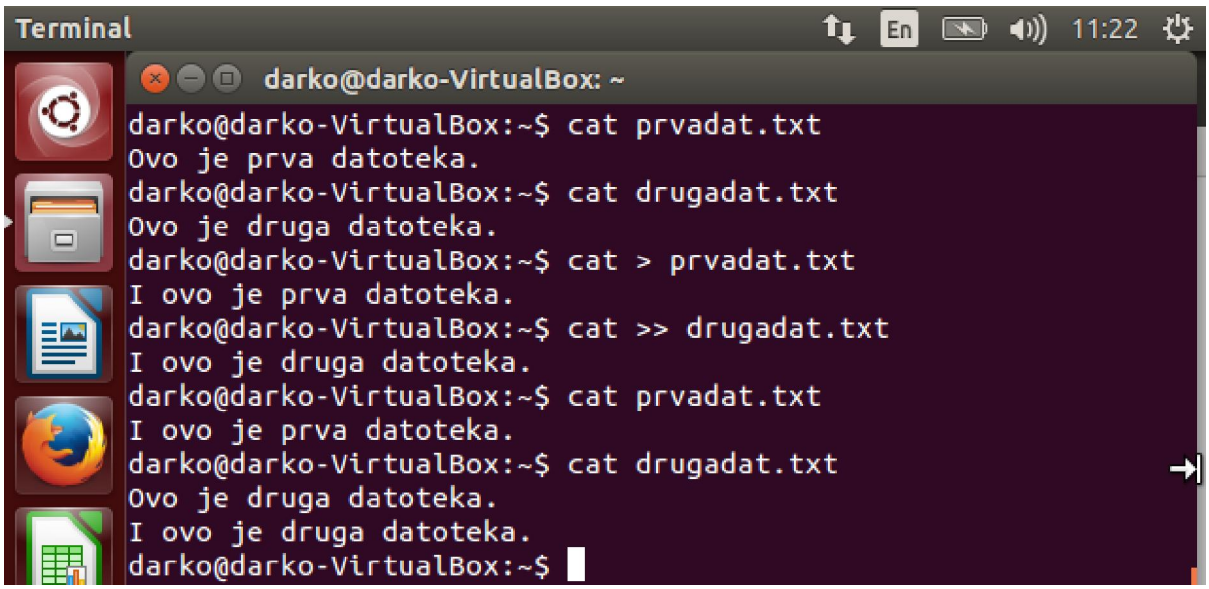
Vrši čitanje jedne ili više datoteka pa rezultate prikazuje na ekran.



Slika 3.10.1. – Primjer upotrebe komande cat

Ako se ne navedu argumenti komande cat i stisne ENTER cat čeka korisnikov unos sa tastature. Nakon što korisnik završi sa unosom i pritisne CTRL+D cat taj unos prikazuje na ekranu. Cat se takođe koristi se operatorima redirekcije > i >>. Za konstrukciju cat > prvadat.txt cat vrši preusmjeravanje. Korisnik unosi željeni tekst i pritiska CTRL+D. Cat ovaj unos ne prikazuje na ekranu već ga preusmjerava u prvadat.txt. Upotrebom operatora redirekcije > preko sadržaja prvadat.txt upisuje se unijeti tekst. U tom slučaju gubi se prvobitni sadržaj datoteke prvadat.txt.

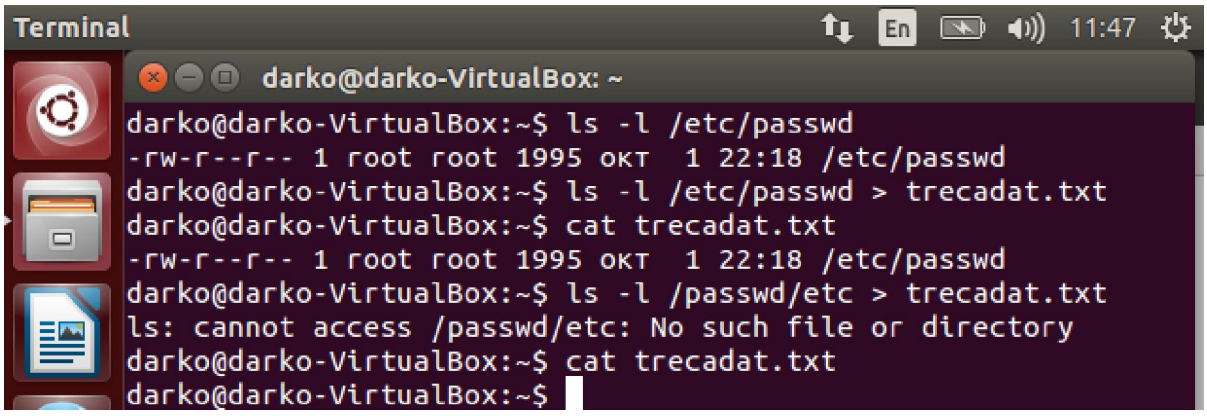
Operator redirekcije >> u kombinaciji sa cat vrši dodavanje na kraj datoteke. Naredba cat >> drugadat.txt neće prebrisati sadržaj drugadat.txt. Unijeti tekst će dodati na kraj ove datoteke.



Slika 3.10.2. – Upotreba komande cat sa operatorima redirekcije

3.11. Redirekcija standardnog izlaza i standardne greške

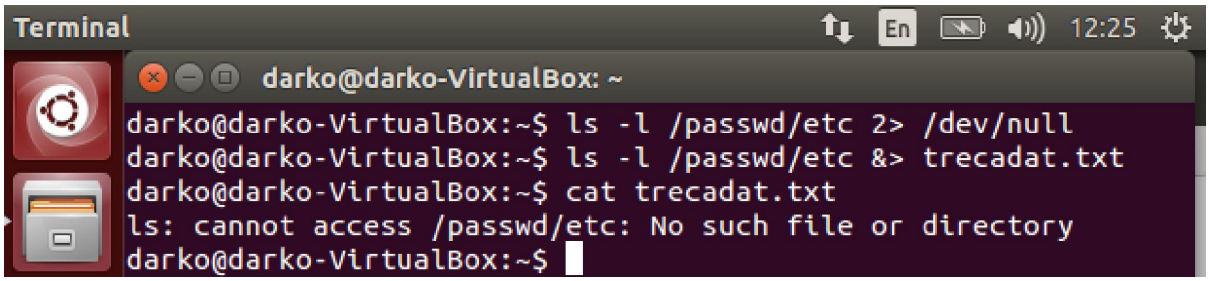
Unosom ls -l /etc/passwd > trecadat.tx vrši se preusmjeravanje izlaza komande ls u trecadat.txt. Ako se sada unese ls -l /passwd/etc > trecadat.txt dobiće se poruka o grešci. Ova greška je prikazana na ekranu jer direktorijum /passwd/etc ne postoji. Operatorom > je zadato da se u trecadat.txt preusmjeri izlaz komande ls a ne greška. Datoteka trecadat.txt je sada prazna. Operator > vrši upisivanje ispočetka. Kako je došlo do greške upisivanje nije uspjelo. Ishod je prazna datoteka trecadat.txt.



Slika 3.11.1. – Ilustracija preusmjeravanja standardnog izlaza

Za redirekciju greške koristi se deskriptor datoteke. Sastoji se od standardnog ulaza, izlaza i greške. Označavaju se brojevima 0, 1, 2. Broj 1 se ne piše jer se podrazumijeva. Preusmjeravanje je moguće izvršiti u tzv "kofu za bitove." To je sistemska datoteka /dev/null. Ono što se tamo

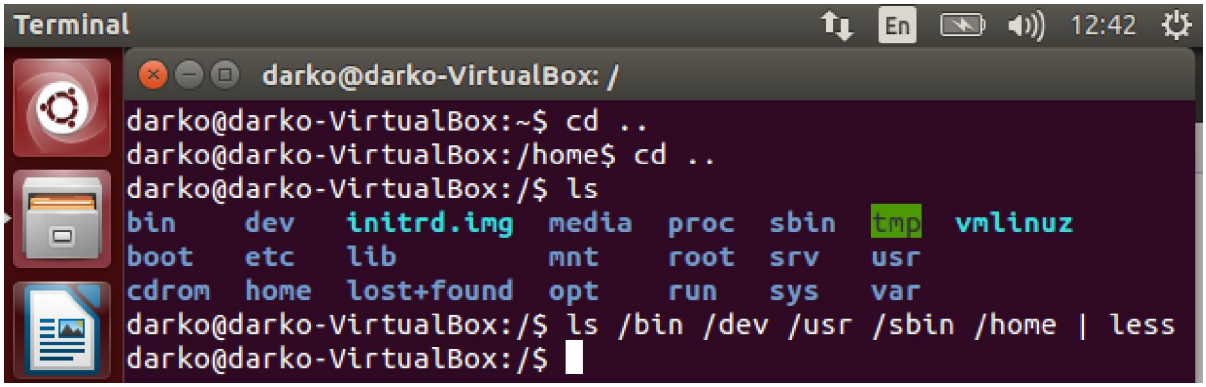
preusmjeri ne može više biti prikazano. Npr. ls -l /passwd/etc 2> /dev/null preusmjerava grešku u /dev/null. Direktorijum /passwd/etc ne postoji. Greška ovom notacijom je tamo preusmjerena. Nije prikazana na ekranu. Naredba ls -l /passwd/etc &> trecadat.txt vrši preusmjeravanje i izlaza i greške ls -l /passwd/etc u trecadat.txt.



Slika 3.11.2. – Preusmjeravanje greške različitim notacijama

3.12. Protočna obrada

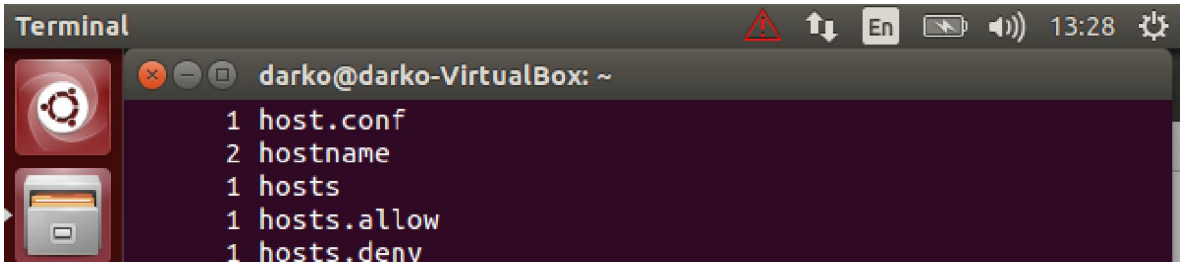
Omogućava dovođenje izlaza jedne naredbe na ulaz druge naredbe. Koristi se sa operatorom |. Ima oblik: naredba1 | naredba2. Kao naredba2 može da se koristi bilo koja komanda koja prihvata izlaz naredbe1 kao svoj ulaz. Na slici 3.12.1. su izlistani neki direktorijumi korjenskog direktorijuma. Taj rezultat je proslijeđen komandi less.



Slika 3.12.1. – Primjer protočne obrade

3.13. uniq i sort

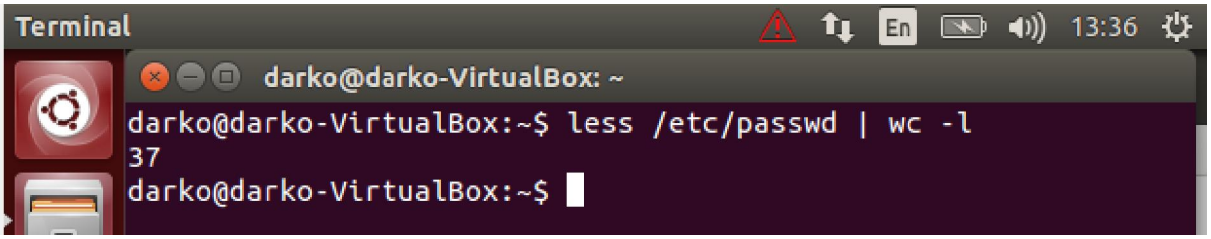
Protočnu obradu je moguće učiniti složenijom. To se postiže uvrštavanjem više naredbi u obradu. Naredba ls /bin /sbin /etc | sort | uniq -c | less je primjer ove primjene. Izlistava se sadržaj /bin /sbin /etc. Taj rezultat se sortira po abecednom redu komandom sort. Naredba uniq -c vrši numeraciju koliko se puta red ponavlja. Sve se to prosleđuje u less. Naredba less prikazuje rezultat na ekranu. Naredba uniq u svom osnovnom obliku eliminiše duplikate redova. Ovu komandu je moguće modifikovati, kao i sve druge. Unosom man komanda prikazuje se detaljno uputstvo za komandu. Takođe, u uputstvu se prikazuje i spisak opcija za modifikaciju.



Slika 3.13.1. - Efekat numerisanja redova pomoću uniq -c

3.14. wc

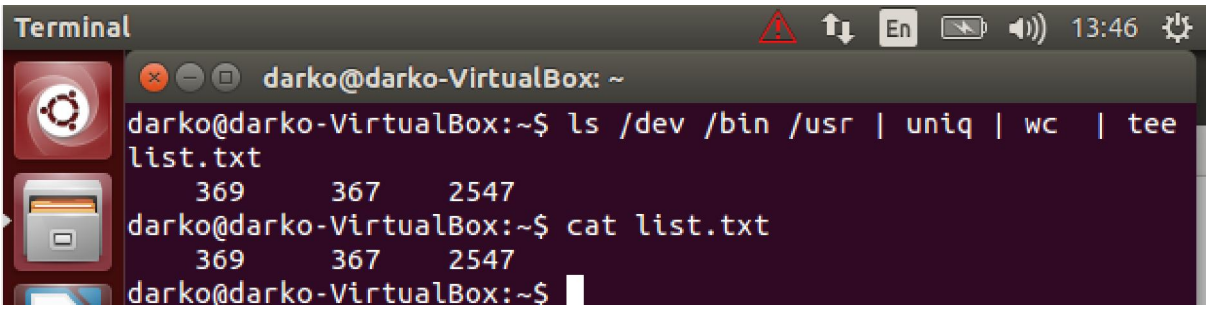
Prikazuje broj redova, riječi i bajtova u datotekama. Modifikacija wc -l prikazuje samo broj redova.



Slika 3.14.1. – Prikazivanje broja redova u datoteci

3.15. tee

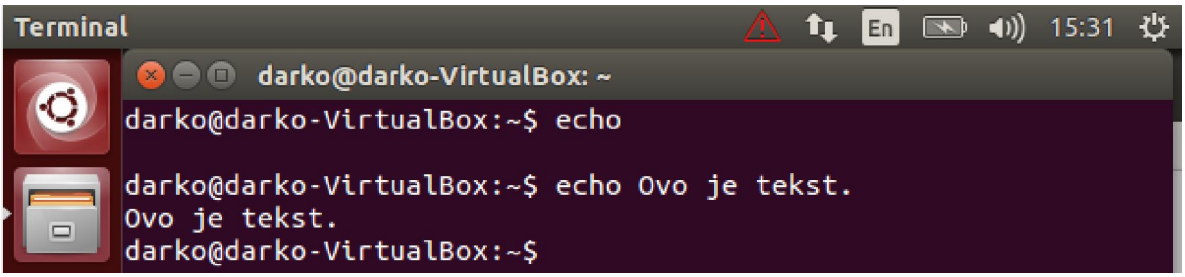
Nekada je potrebno međurezultate obrade sačuvati u tekstualnu datoteku. To se postiže ovom komandom.



Slika 3.15.1. – Smještanje rezultata protočne obrade u tekstualnu datoteku

3.16. echo

Ispisuje parametre komandne linije na standardnom izlazu. Ako parametri nisu zadati ispisuje se prazan red.

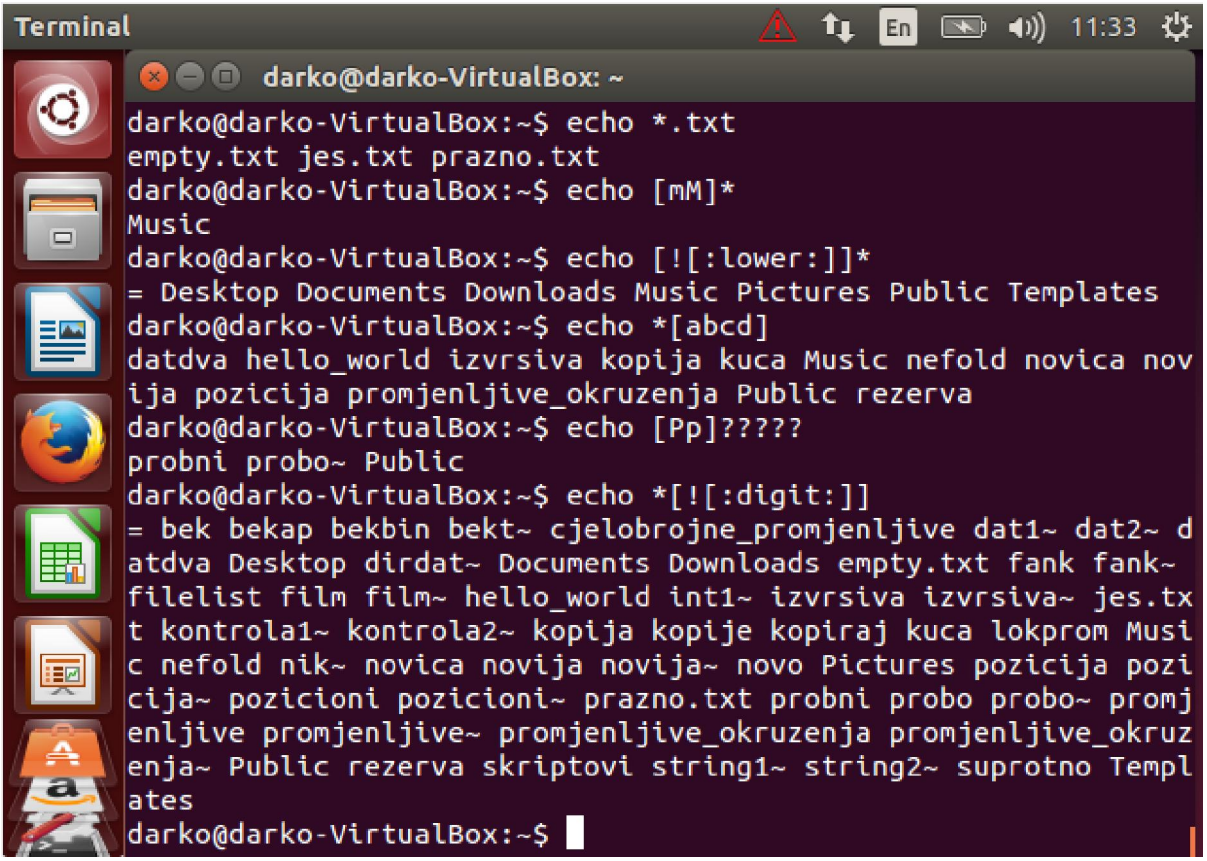


Slika 3.16.1. – Primjer upotrebe komande echo

3.17. Džoker znakovi

Komandna linija između ostalog podrazumijeva intenzivan rad sa datotekama. U tu svrhu često je potrebno zadati imena datoteka grupno. Takav postupak se naziva globiranje. U tu svrhu se koriste džoker znakovi. Neki od načina upotrebe džoker znakova su:

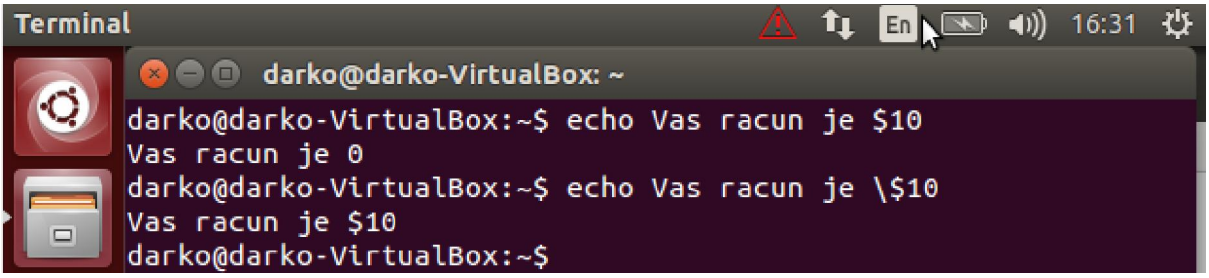
- *.txt Znak * vrši zamjenu bilo kojih znakova koji se nalaze prije .txt. Ovim se omogućava zadavanje imena svih datoteka koji se završavaju na .txt.
- [mM]* Zadaju se imena svih datoteka koje počinju ili sa m ili sa M.
- [![:lower:]]* Zadaju se sve datoteke koje ne počinju malim slovom.
- *[abcd] Sve datoteke koje se završavaju sa a, b , c, d.
- [Pp]????? Sve datoteke koje počinju ili sa velikim ili malim p iza kojeg sledi tačno pet znakova.
- *[![:digit:]] Sve datoke koje se ne završavaju cifrom.



Slika 3.17.1. – Upotreba džoker znakova

3.18. Specijalni znakovi

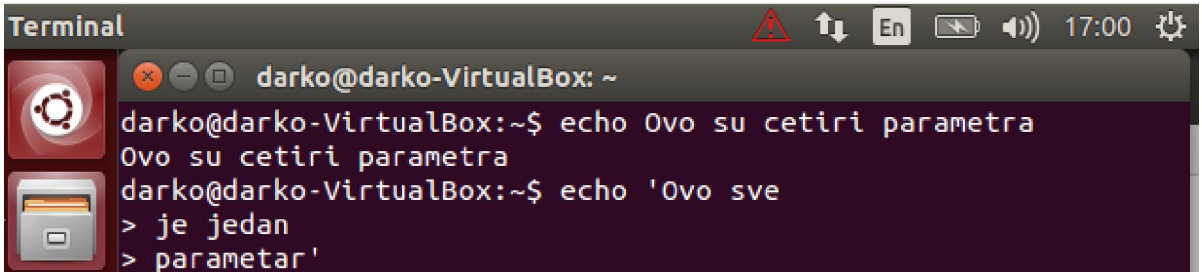
Postoji grupa znakova koja pored svog osnovnog literalnog značenja ima i posebno značenje za ljusku. To je takozvano metaliteralno značenje. U te znakove ubrajaju se: &, *, ?, [ ], < >, | , ( ) , ´ , # , $, ^, ' , " , { } , ; , \ . Znak \ eliminiše svojstvo svih specijalnih znakova. U prvom primjeru $1 se zamjenjuje sa praznim stringom. Znak za dolar se koristi za dohvatanje sadržaja u promjenljivoj. Kako promjenljiva nije definisana ta vrijednost se mijenja praznim stringom. U drugom primjeru specijalno značenje ovog znaka je uklonjeno pomoću \ . U tom slučaju znak za dolar ima svoje literalno značenje.



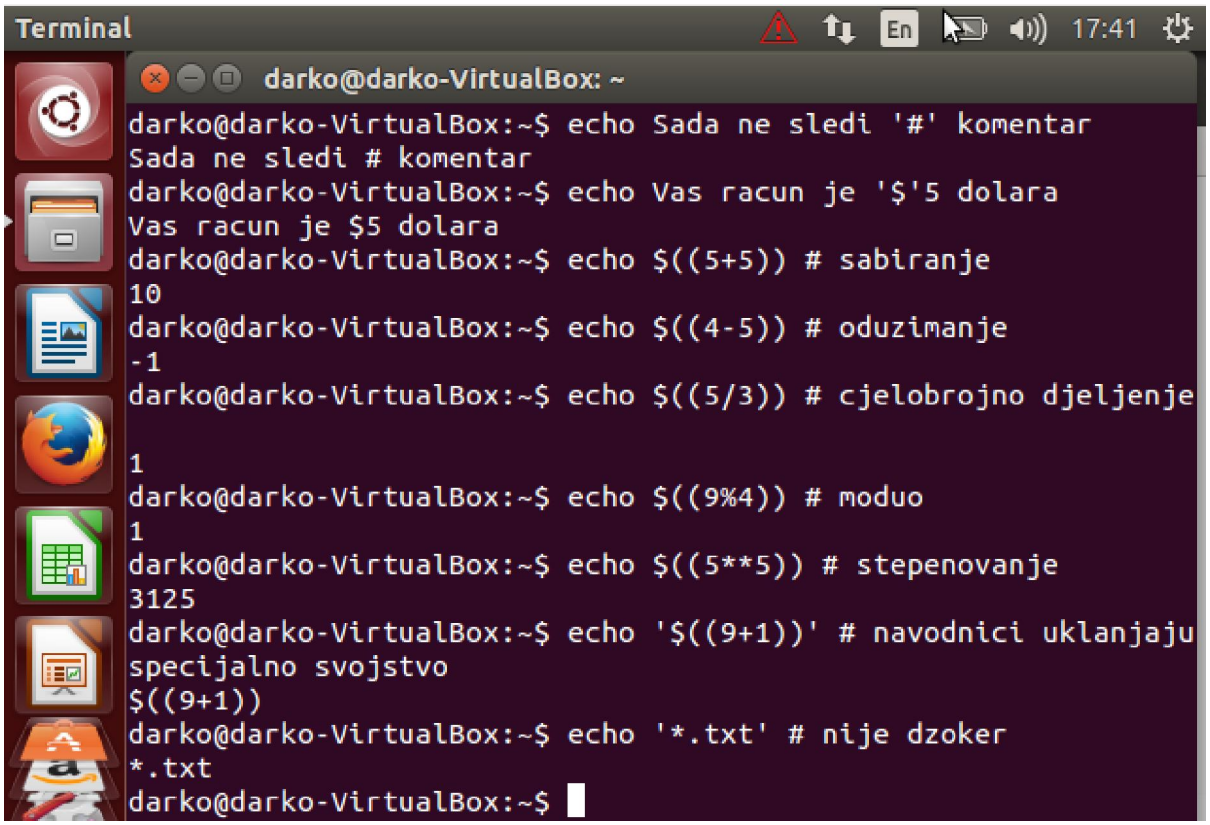
**Slika 3.18.1. – Efekat specijalnog znaka **

3.19. Jednostruki navodnici

Eliminišu specijalno značenje svih specijalnih znakova. Razmak omogućava da se razgraniči početak i kraj parametara. Ako postoji sedam parametara koji su okruženi jednostrukim navodnicima onda se oni tretiraju kao jedan parametar. Znak # predstavlja početak komentara. Sve što se iza njega nalazi ljuska ne čita. Moguće je vršiti određene aritmetičke operacije sa cijelim brojevima. Za ekpsanziju aritmetičkih izraza koristi se sledeći oblik komande: $((*izraz*)) gdje je *izraz* aritmetički izraz koji se sastoji od vrijednosti i aritmetičkih parametara [1].



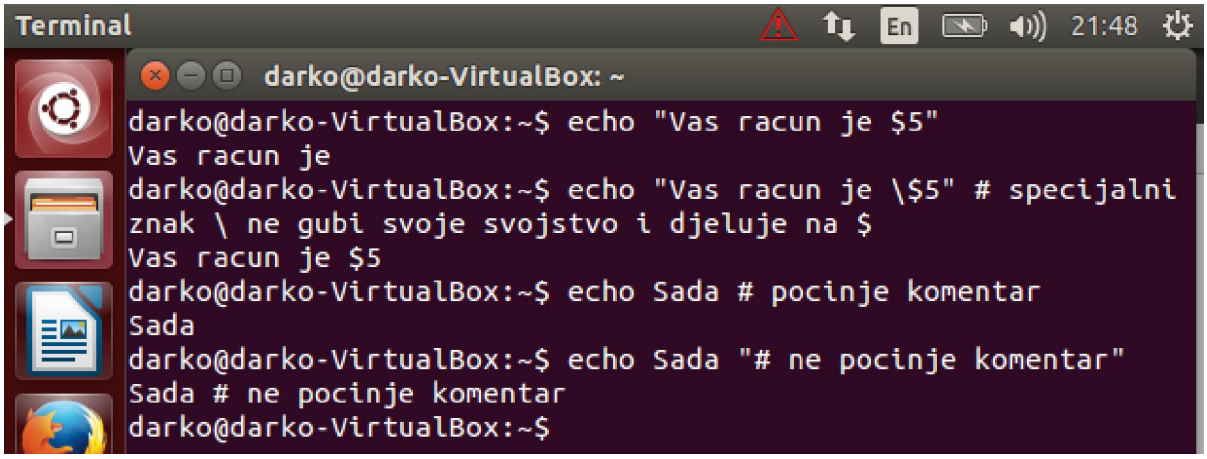
Slika 3.19.1. – Primjer upotrebe jednostrukih navodnika



Slika 3.19.2. – Efekat upotrebe jednostrukih navodnika

3.20. Dvostruki navodnici

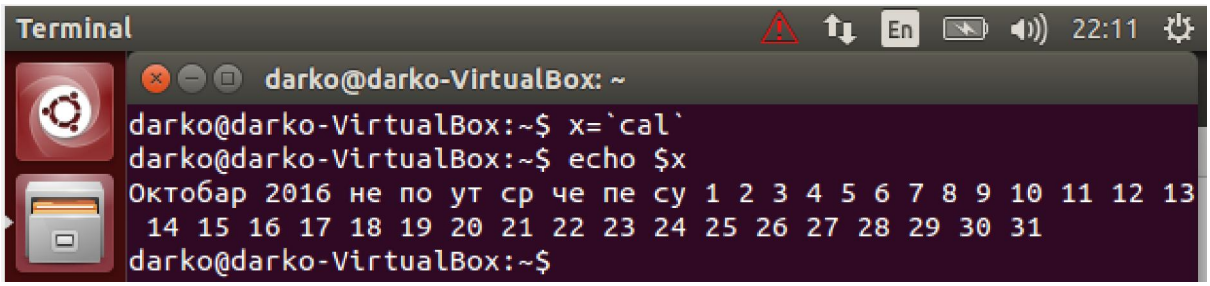
Predstavljaju blaži kriterijum u odnosu na jednostruke. To znači da određeni specijalni znakovi zadržavaju svoje značenje. To su: $, ´ , \.



Slika 3.20.1. – Primjer upotrebe dvostrukih navodnika

3.21. Obrnuti navodnici

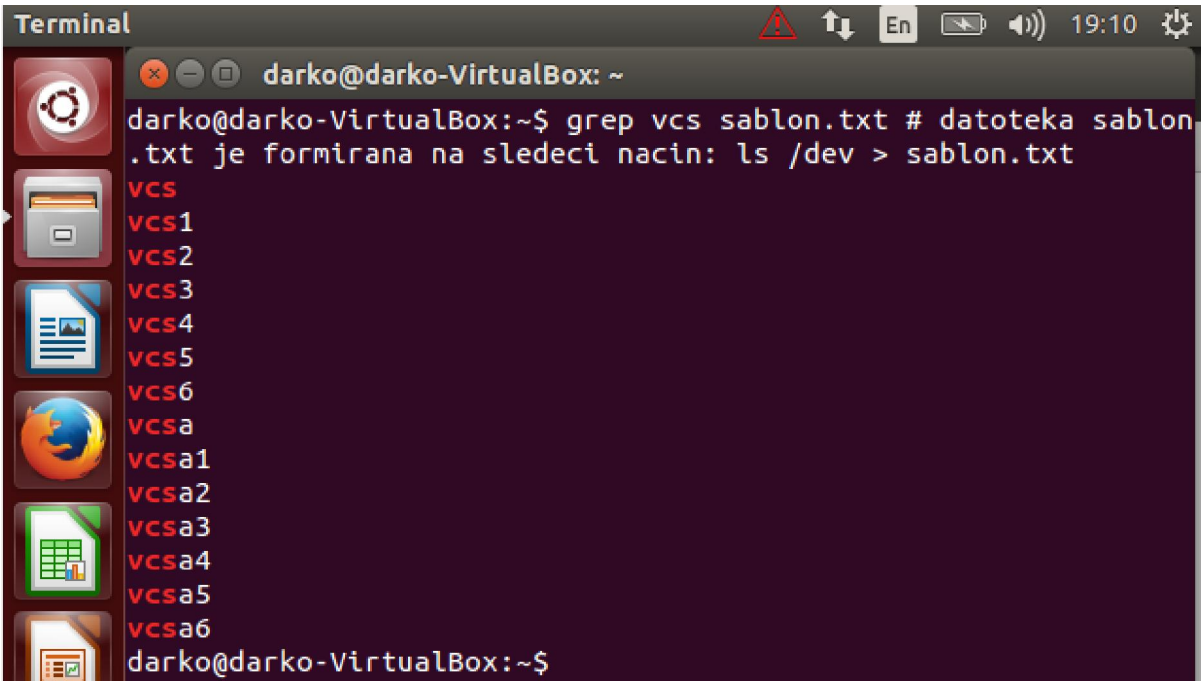
Ne igraju ulogu kod specijalnih znakova. Osnovna njihova uloga je da se eliminišu novi redovi. Takođe sve višestruke razmake i tabulatore zamjenjuju jednostrukim. Takav izlaz se dodjeljuje nekoj promjenljivoj.



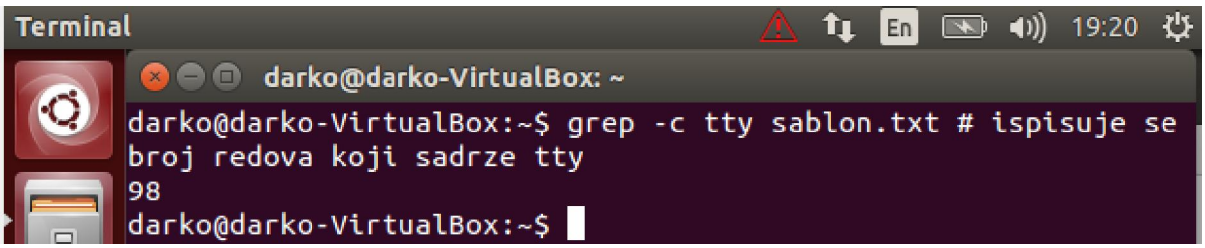
Slika 3.21.1. – Primjer upotrebe obrnutih navodnika

3.22. grep

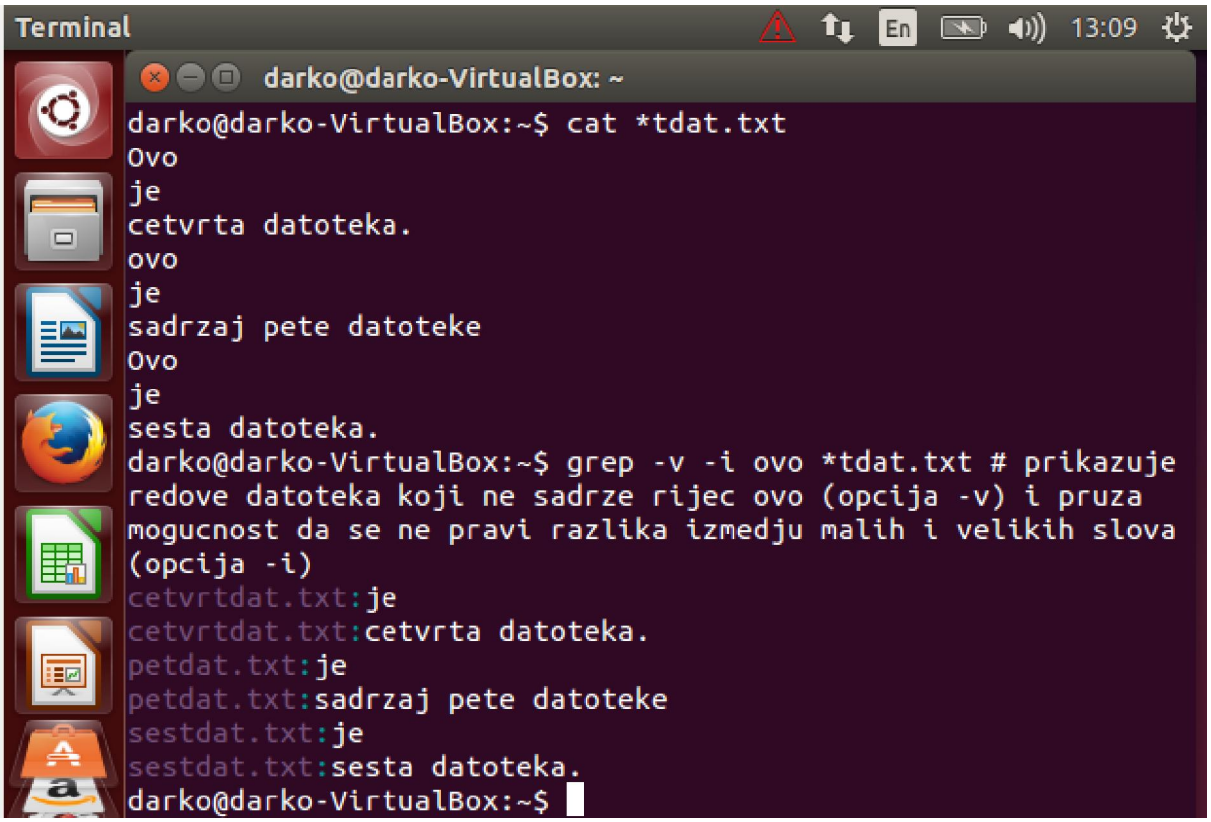
Vrši pretraživanje redova teksta sa svog ulaza koji sadrži određeni šablon. Taj ulaz može biti tastatura ili druga datoteka. Regularni izrazi predstavljaju složenije šablone. Ima oblik grep *šablon.* Unosom ovakve konstrukcije vrši se ispis redova teksta po *šablonu*, pri čemu je tekst koji se pretražuje unijet sa tastature. Unos grep vcs sablon.txt vrši ispis svih redova datoteke sablon.txt koji sadrže vcs. Konstrukcija grep –c tty sablon.txt ispisuje broj redova datoteke sablon.txt koji sadrže tty. Konstrukcija grep –v –i ovo *tdat.txt vrši ispis svih redova datoteka koje se završavaju na *tdat.txt i koji ne sadrže riječ ovo. Dodavanjem opcije –i zadaje se da se ne pravi razlika između malih i velikih slova.



Slika 3.22.1. – Pretraživanje redova teksta uz pomoć grep



Slika 3.22.2. – Primjer modifikovanja rezultata pretrage

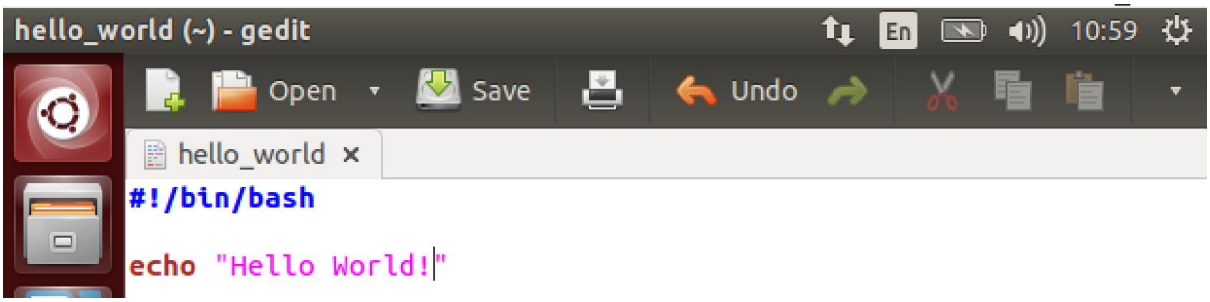


Slika 3.22.3. – Pretražuju se sve datoteke koje se završavaju na tdat.txt

4. PISANJE KOMANDNIH SKRIPTOVA

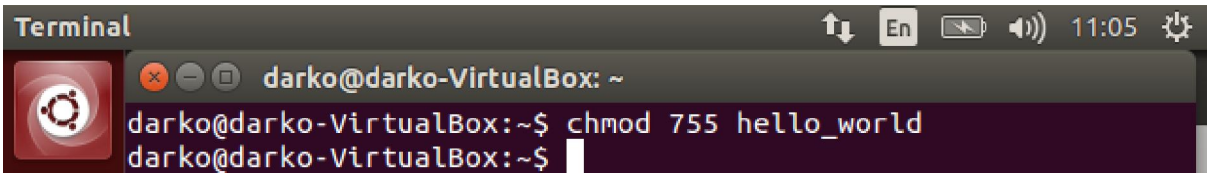
4.1. hello_world

Komandni skript (engl. *shell script*) jeste niz komandi upisanih u datoteku; ljuska učitava te komande iz datoteke isto kao kada su otkucane u prozoru terminala [1]. Ne može se bilo koja tekstualna datoteka smatrati skriptom. Potrebno je podesiti ovlašćenja da bi skript mogao da se izvrši. Svaki komandni skript počinje sa konstrukcijom #! koja se naziva šebeng (engl. *shebang*). Šebeng se koristi da najavi ime programa koji se koristi za pokretanje komandi koje se nalaze u skriptu. U slučaju *bash* ljuske prvi red ima oblik: #!/bin/bash. Ovim se saopštava sistemu da se u direktorijumu bin nalazi program *bash* koji se koristi za pokretanje skripta. Pozdravnu poruku "Hello World!" treba otkucati u editoru teksta i sačuvati kao hello_world.



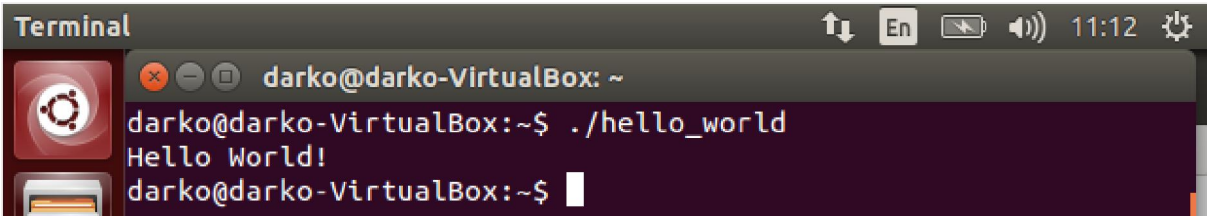
Slika 4.1.1. – Ispisivanje pozdravne poruke na ekranu

Sledeći korak je učiniti skript izvršnim. To se postiže unošenjem na komandnoj liniji chmod 755 hello_world. Komanda chmod 755 omogućava da svi korisnici mogu da izvršavaju skript.



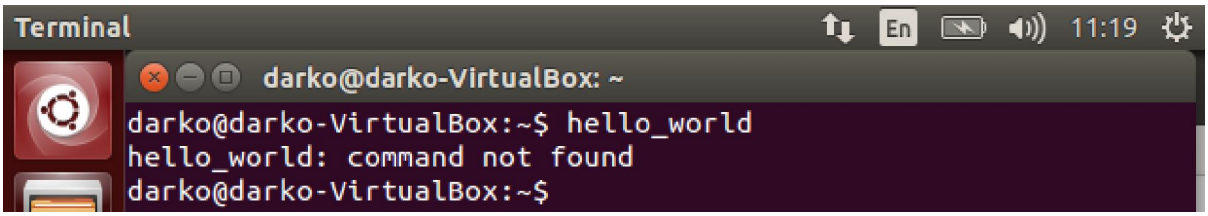
Slika 4.1.2. – Unos koji omogućava da svi korisnici mogu da izvršavaju skript

Pokretanje skripta vrši se unošenjem ./hello_world na komandnoj liniji. Znak . je tekući radni direktorijum u kome se nalazi /hello_world.



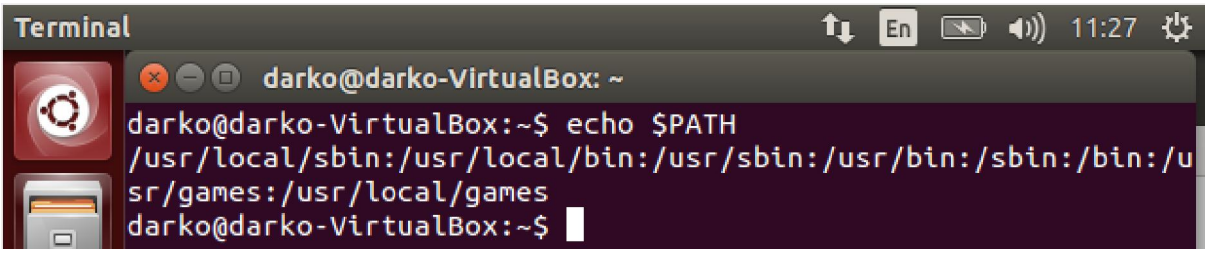
Slika 4.1.3. – Pokretanje skripta

Ako se unese hello_world bez navođenja eksplicitne putanje dobija se sledeća poruka:



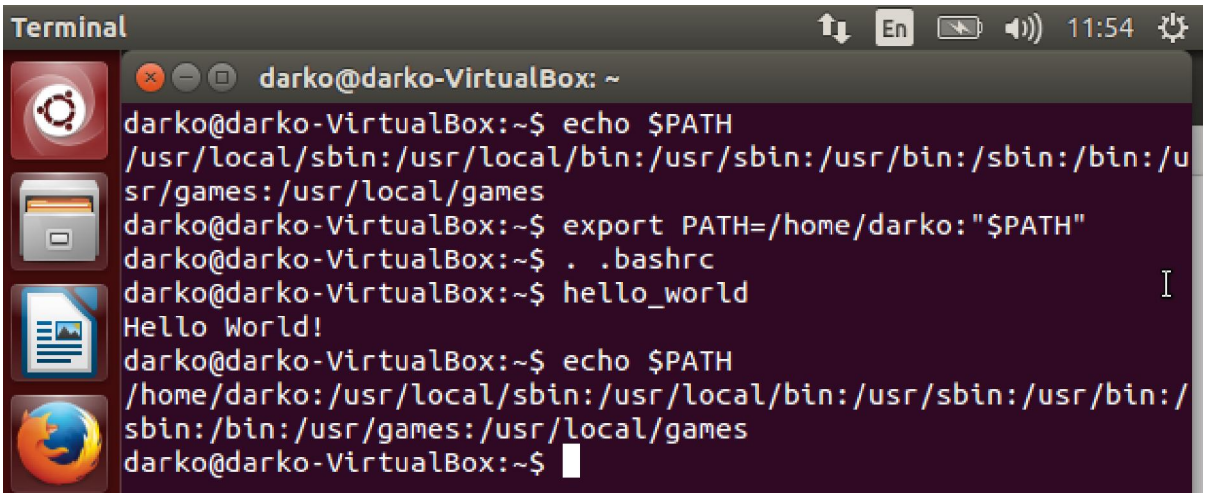
Slika 4.1.4. – Rezultat unosa skripta bez navođenja eksplicitne putanje

Ovo se dešava zato što ljuska unos hello_world doživljava kao i svaki drugi izvršni program. Skripta hello_world je sačuvana u matičnom direktorijumu. Spisak direktorijuma koje ljuska automatski pretražuje u potrazi za izvršnim programima nalazi se u promjenljivoj PATH. Oni su razdvojeni dvotačkama.



Slika 4.1.5. – Ispitivanje sadržaja promjenljive PATH

Da se ne bi navodila eksplicitna putanja moguće je matični direktorijum dodati u promjenljivu okruženja PATH. Unošenje se vrši komandom export na sledeći način: export PATH=/home/darko:"$PATH". Ovo će važiti u sledećoj sesiji terminala. Da bismo omogućili da važi u trenutnoj potrebno je da ljuska ponovo pročita sistemsku datoteku .bashrc. Potrebno je unijeti . .bashrc. Znak . omogućava da se pročita datoteka .bashrc. Nakon toga nije potrebno pokretati terminal ispočetka jer je /home/darko uspješno dodat u promjenljivu PATH i skript hello_world će moći da se izvrši bez unošenja eksplicitne putanje.

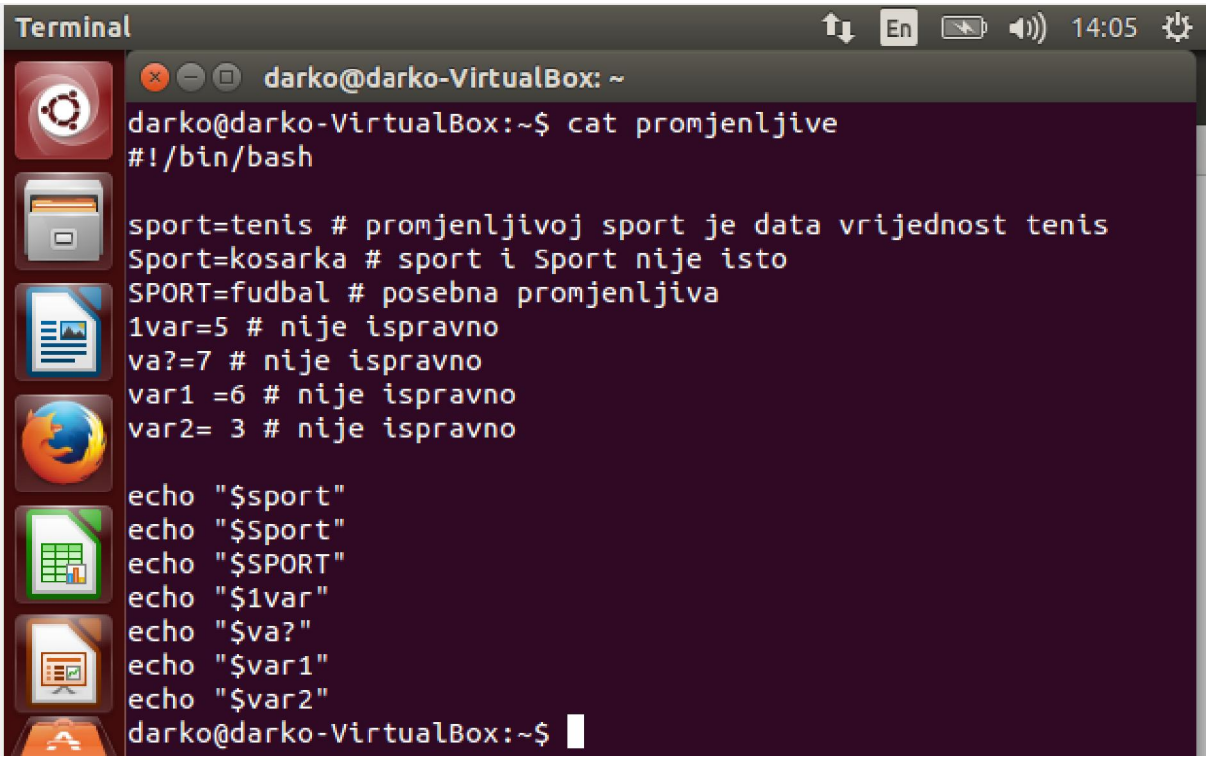


Slika 4.1.6. – Dodavanje matičnog direktorijuma u promjenljivu okruženja PATH

4.2. Promjenljive

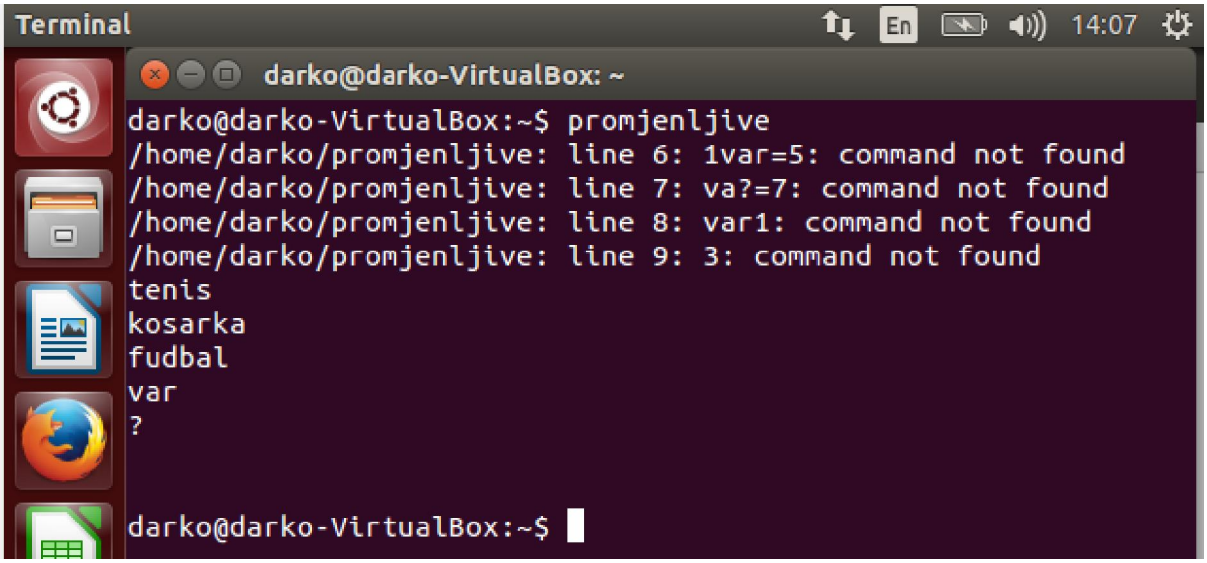
Promjenljiva predstavlja mali, imenovani dio memorije, kome može biti dodjeljena vrijednost. Ne postoji klasifikacija promjenljivih po tipovima jer sve promjenljive ljuska doživljava kao stringove. Kao što im samo ime ukazuje, njihova vrijednost se može mijenjati. Imena promjenljivih su "case sensitive". To znači da ime promjenljive *sport* nije isto što i ime *Sport*. Ako se na primjer nekoj promjenljivoj dodijeli vrijednost 5, ljuska tu promjenljivu doživljava kao string koji sadrži znak 5 a ne integer 5 kao što je to u nekim drugim programskim jezicima. Ime promjenljive mora biti jedinstveno i ne mogu dvije promjenljive da imaju isto ime. Postoje određena pravila za davanja imena promjenljivim. Ime promjenljive može sadržati sledeće znakove:

- a-z mala slova
- A-Z velika slova
- 0-9 cifre
- _ znak za podvlačenje

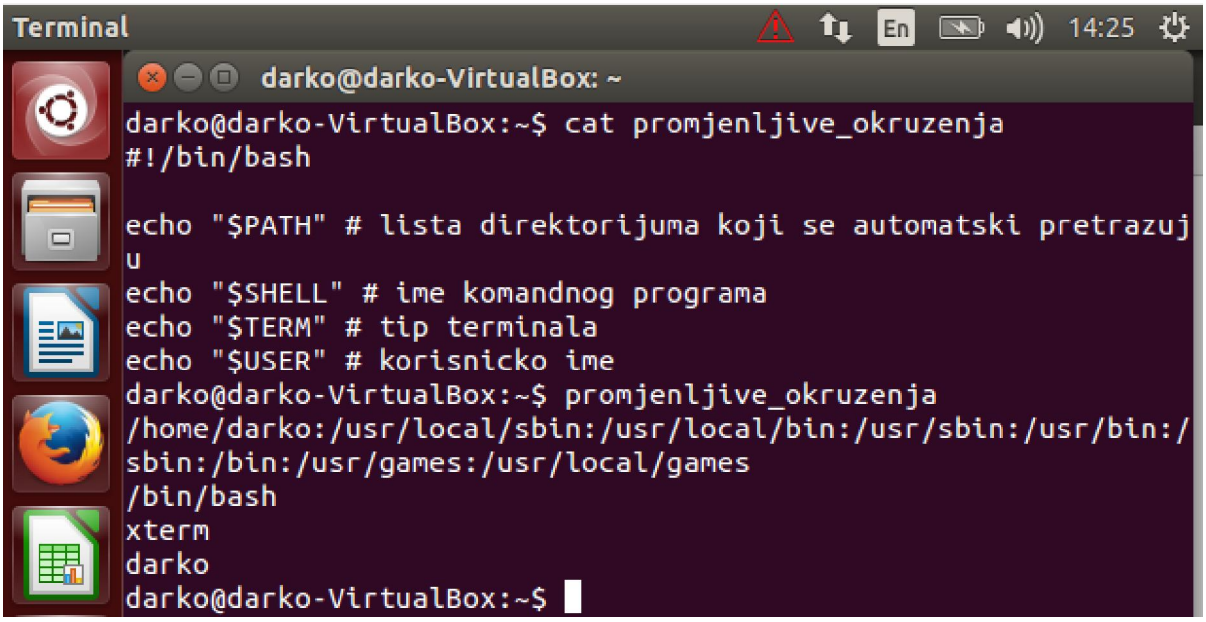


Slika 4.2.1. – Kod skripta koji ilustruje pravilnu i nepravilnu dodjelu vrijednosti promjenljivima

Ne smije biti razmaka sa bilo koje strane znaka = prilikom dodjele vrijednosti promjenljivoj. Ime promjenljive ne smije počinjati cifrom. Za dohvatanje vrijednosti koja je dodjeljena promjenljivoj koristi se znak $. Ako se promjenljivoj dodijeli vrijednost koja sadrži razmake onda je potrebno tu vrijednost okružiti navodnicima. Promjenljive okruženja su specifičan tip promjenljivih i one su unaprijed postavljenje.



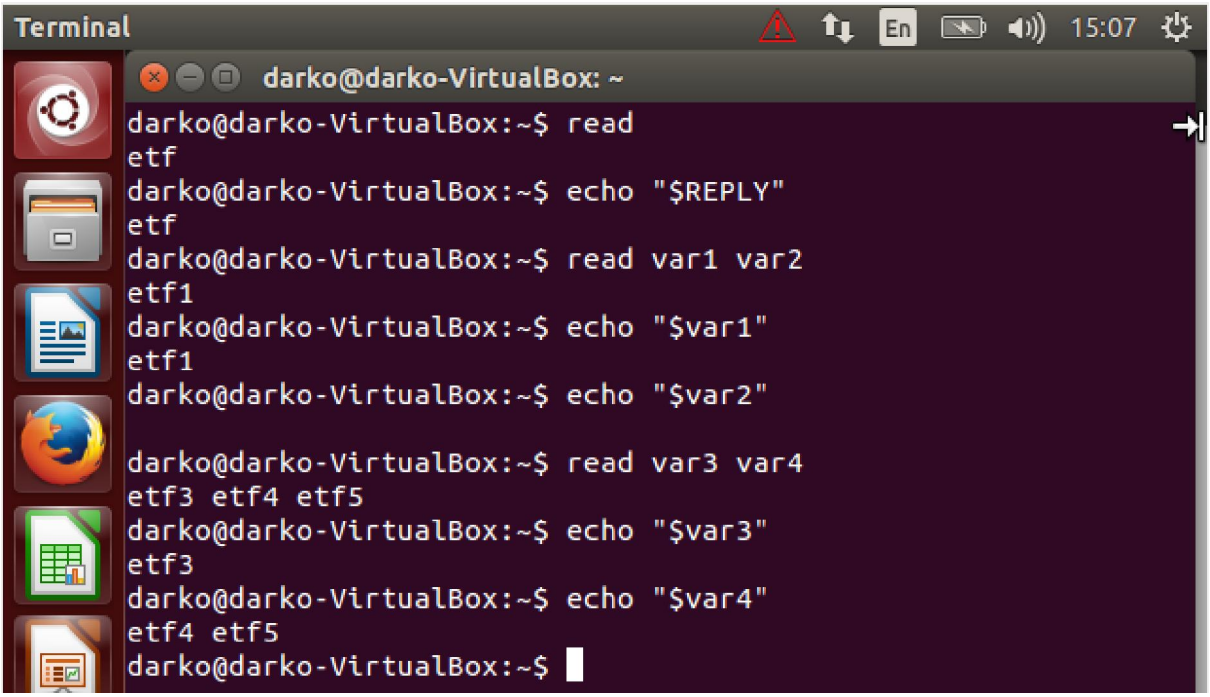
Slika 4.2.2. – Testiranje skripta za dodjelu vrijednosti promjenljivima



Slika 4.2.3. – Ispitivanje sadržaja promjenljivih okruženja

4.3. read

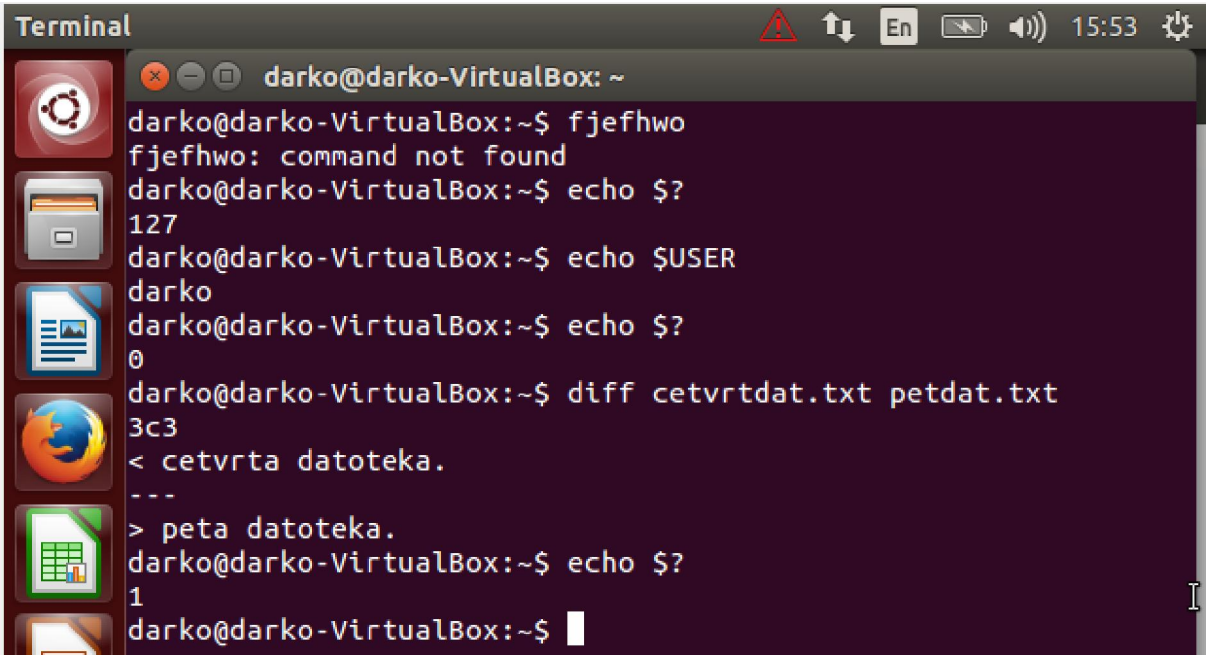
Služi za čitanje standardnog ulaza (obično tastature) i smještanje informacija u promjenljive ljuske. U skriptovima se najčešće koristi da prima odgovore na pitanja koja postavlja skript. Ukoliko nije navedena promjenljiva u koju se smješta unos, taj unos se smješta u promjenljivu *REPLY*. Ako je broj unijetih parametara manji od broja promjenljivih, promjenljive koje su neiskorišćene neće sadržati ništa. Ako je broj unijetih parametara veći od broja promjenljivih parametri koji su "višak" smještaju se u poslednju promjenljivu.



Slika 4.3.1. – Upotreba naredbe read

4.4. Izlazni status

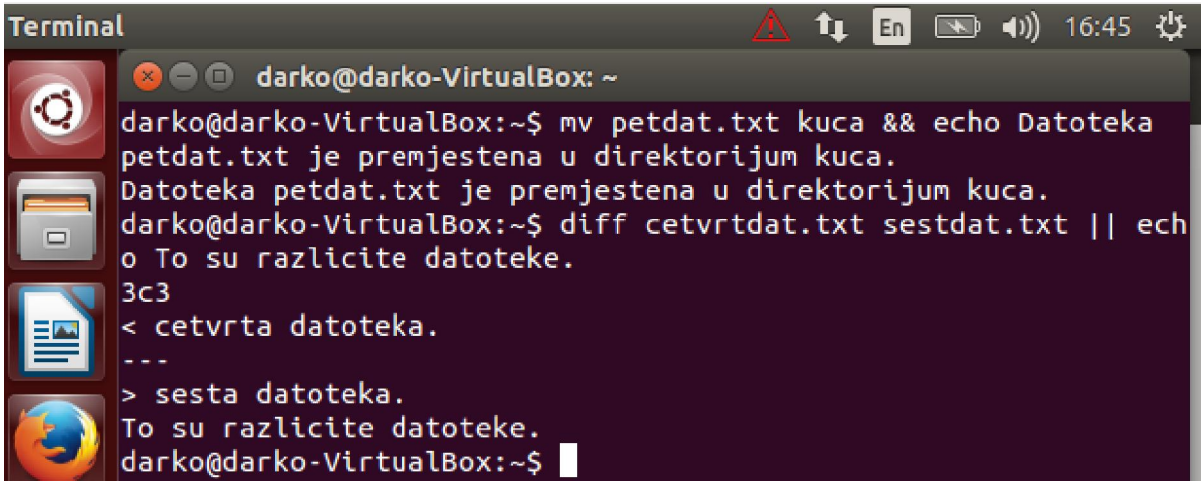
Nakon izvršenja bilo koje komande, nevidljivo se šalje poruka o njenoj uspješnosti u vidu izlaznog statusa. Izlazni status 0 upućuje na to da je komanda uspješno izvršena. Izlazni status različit od nule najčešće je znak da se desila neka greška. Međutim, to nije uvijek slučaj. Komanda diff daje izlazni status 0 ako su datoteke identične. Ako nisu identične, izlazni status je različit od nule iako nije došlo do greške. Izlazni status se ispituje unosom echo $?.



Slika 4.4.1. – Ispitivanje izlaznog statusa

4.5. Logički konstrukti && i ||

Moguće je postići da se neka naredba u skriptu izvrši samo ako je ispunjen određen uslov. Takvi uslovi su najčešće izlazni statusi drugih naredbi. Konstrukcija komanda1 && komanda2 omogućava da se komanda2 izvrši ako je izlazni status komande 1 jednak 0. Konstrukcija komanda1 || komanda2 omogućava da se komanda2 izvrši ako se komanda1 izvrši sa statusom koji je različit od nula.



Slika 4.5.1. – Primjer upotrebe logičkih konstrukata && i ||

4.6. if naredba

Ova naredba predstavlja mnogo moćniju konstrukciju u odnosu na gore navedene naredbe. Ima sledeći oblik:

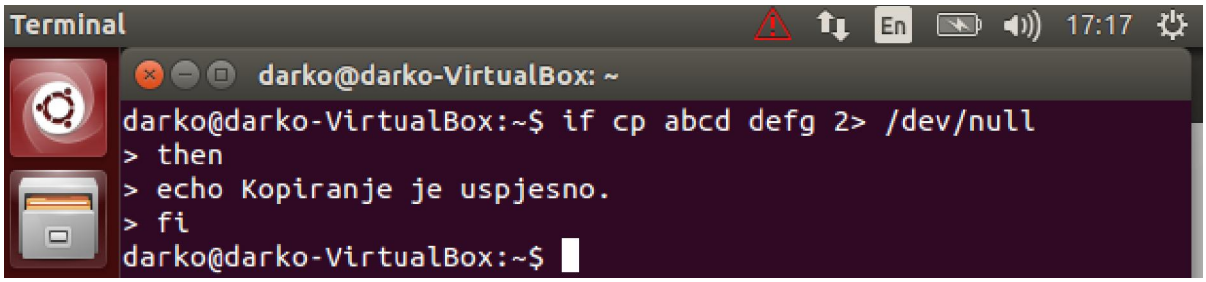
if komanda1

then

blok naredbi

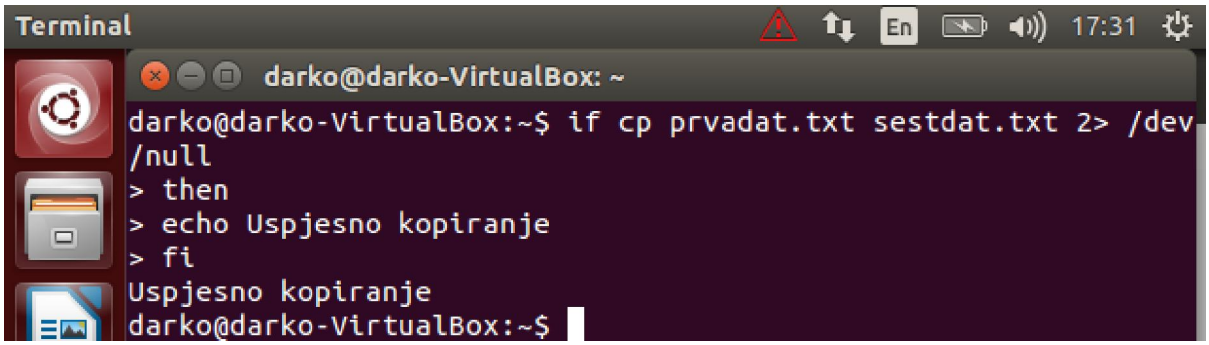
fi# Ukoliko izlazni status nije nula prelazi se u ovaj red i terminira se if.

Na slici 4.6.1. ilustrovana je primjena if naredbe. Zadato je da se izvrši kopiranje datoteka, i da se poruka ako je kopiranje uspješno ispiše na ekranu. Kako kopiranje nije uspješno, jer su zadate nepostojeće datoteke, greška koja se desila preusmjerena je u /dev/null pa na ekranu nije prikazano ništa.



Slika 4.6.1. – Primjer neuspješnog kopiranja uz upotrebu if

Za kopiranje postojećih datoteka isti blok naredbi izgleda ovako:



Slika 4.6.2. – Primjer uspješnog kopiranja uz upotrebu if

4.7. else

Koristi se sa if naredbom i omogućava da se ukoliko komanda u uslovu ne vrati nulti izlaz zada novi blok naredbi. Ima oblik:

if komanda1 # ukoliko je izlazni status komanda1 0 prelazi se u prvi blok naredbi

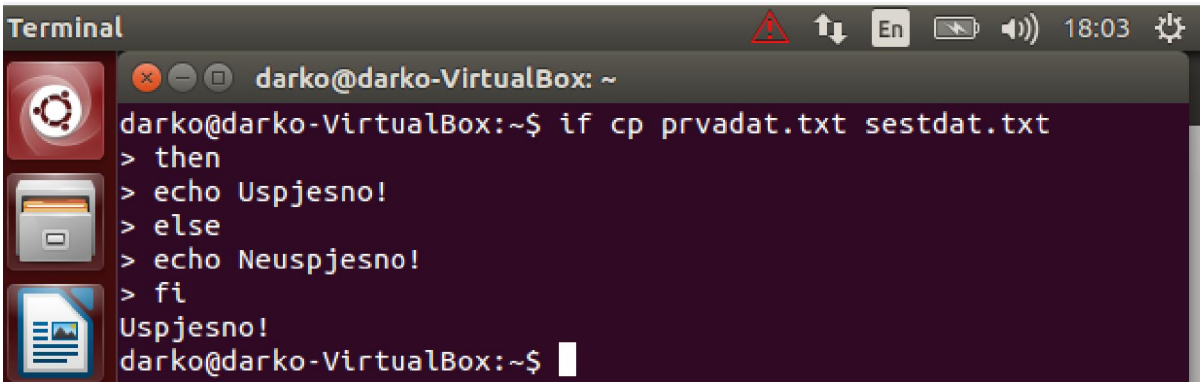
then

prvi blok naredbi

else

drugi blok naredbi # ukoliko je izlaz komanda1 nenulti prelazi se u drugi blok naredbi

fi



Slika 4.7.1. – Upotreba else naredbe

4.8. elif

Često je potrebno zadati više od dva bloka naredbi. To omogućava elif. Ima sledeći oblik:

if komanda1

then

prvi blok naredbi

elif komanda2

then

drugi blok naredbi

else

treci blok naredbi

fi

Ukoliko *komanda1* u if-u daje nenulti izlazni status prelazi se na elif. Ako *komanda2* u elif-u daje nenulti izlazni status prelazi se na else. Naredba elif se može ponavaljati beskonačno puta, dok je naredbu else moguće iskoristiti samo jednom.

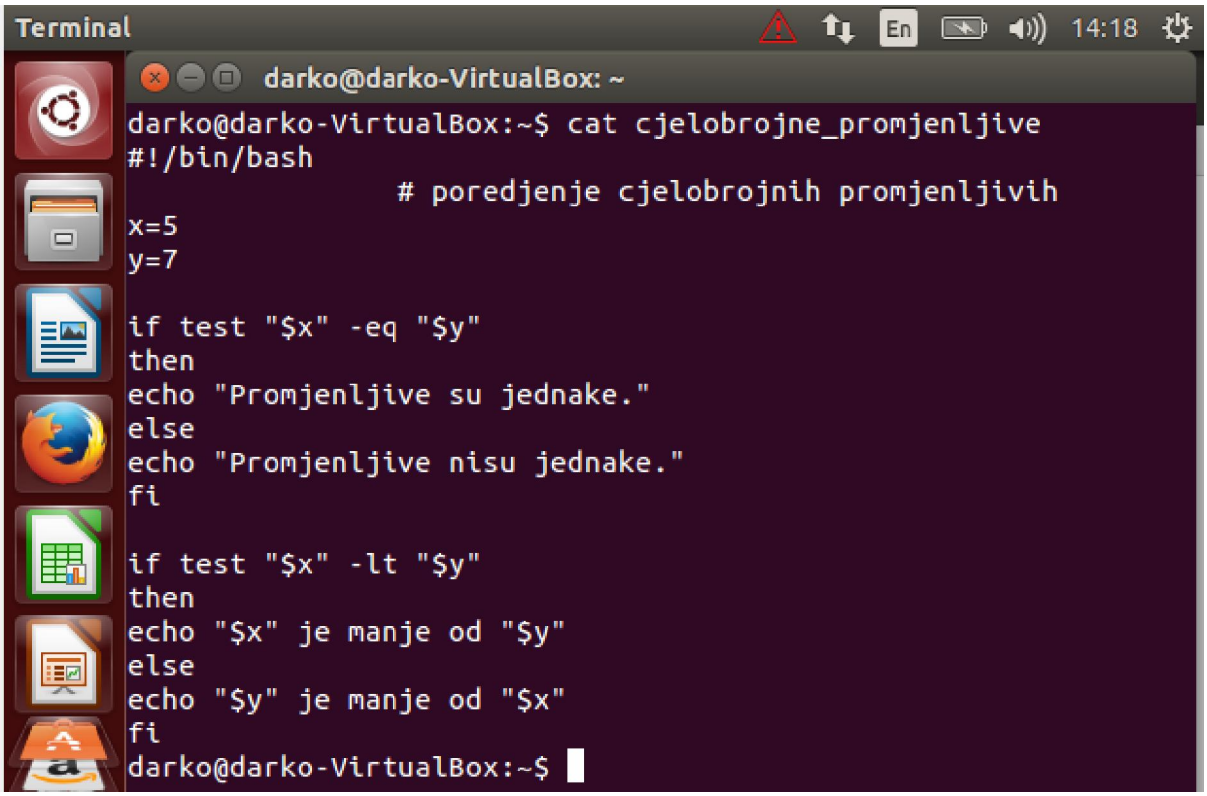
4.9. test

Koristii se sa if naredbom. Ne daje nikakav rezultat na ekranu već if koristi izlaz ove naredbe kao svoj uslov. Naredba test najčešće poredi dvije vrijednosti i generiše nulu za uspješno poređenje i nenulti izlaz za neuspješno poređenje. Iako ljuska ne poznaje tipove podataka već sve tretira kao niz znakova moguće je izvršiti određene operacije poređenja nad promjenljivima. Od ključne važnosti je da li se promjenljiva sastoji samo od cifara ili ne.

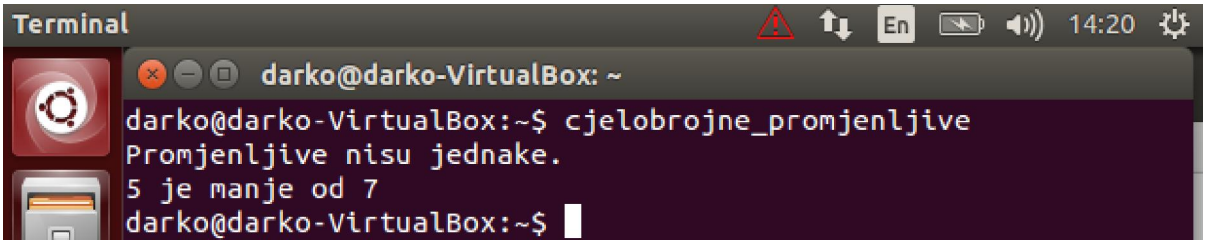
4.9.1. Ispitivanje cjelobrojnih promjenljivih

Operacije poređenja koje su dozvoljene:

- -eq jednako
- -ne nije jednako
- -le manje ili jednako
- -lt manje
- -ge veće ili jednako
- -gt veće



Slika 4.9.1.1. – Ilustracija poređenja cjelobrojnih promjenljivih



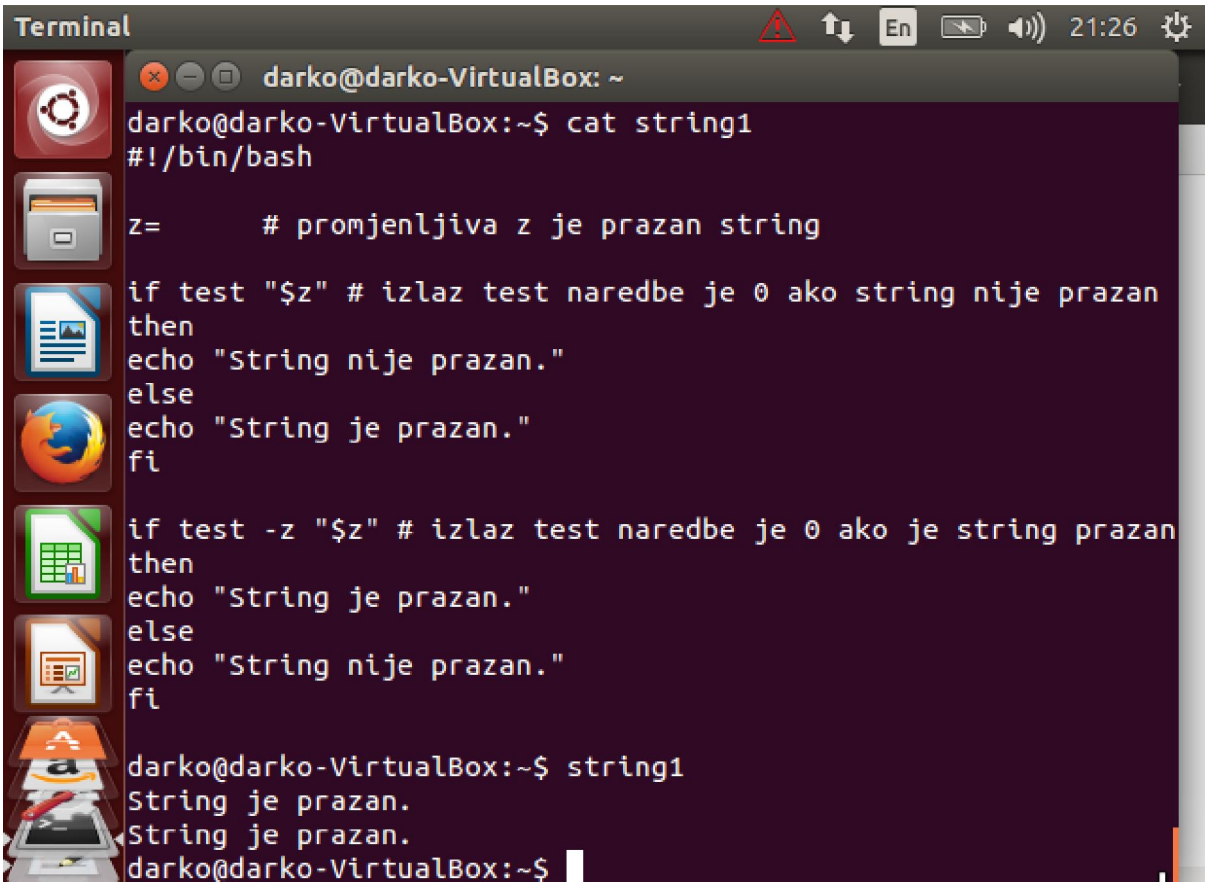
Slika 4.9.1.2. – Rezultat poređenja cjelobrojnih promjenljivih

4.9.2. Rad sa stringovima

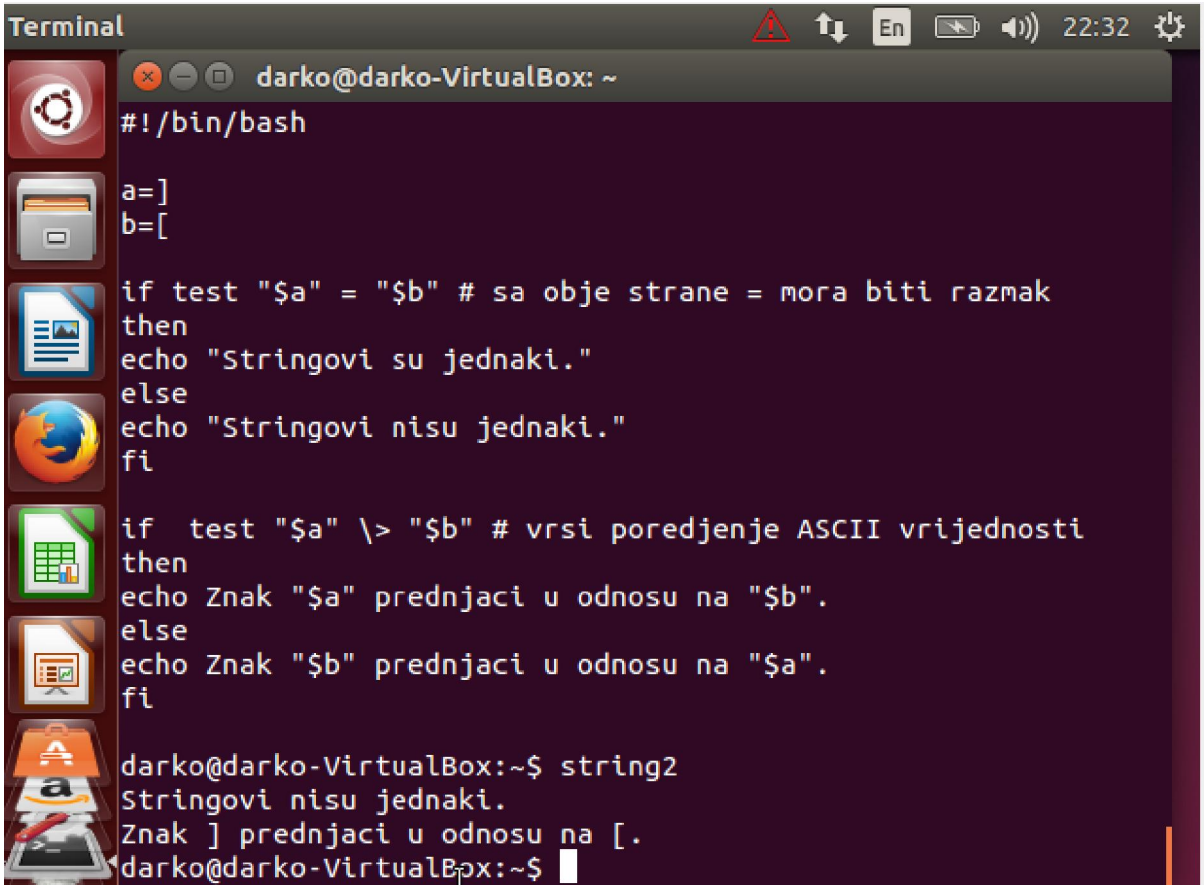
Stringovi predstavljaju nizove bilo kojih znakova i ljuska dozvoljava određene operacije njihovog poređenja. Neke od njih su:

- -n string ispituje da li je dužina stringa veća od nule
- -z string ispituje da li je dužina stringa jednaka nuli
- = ispituje da li su stringovi jednaki

- != ispituje da li su stringovi različiti
- > , < ispituje koji od dva stringa prednjači u ASCII tabeli



Slika 4.9.2.1. – Kod i rezultat skripta koji ilustruje operacije nad praznim stringom



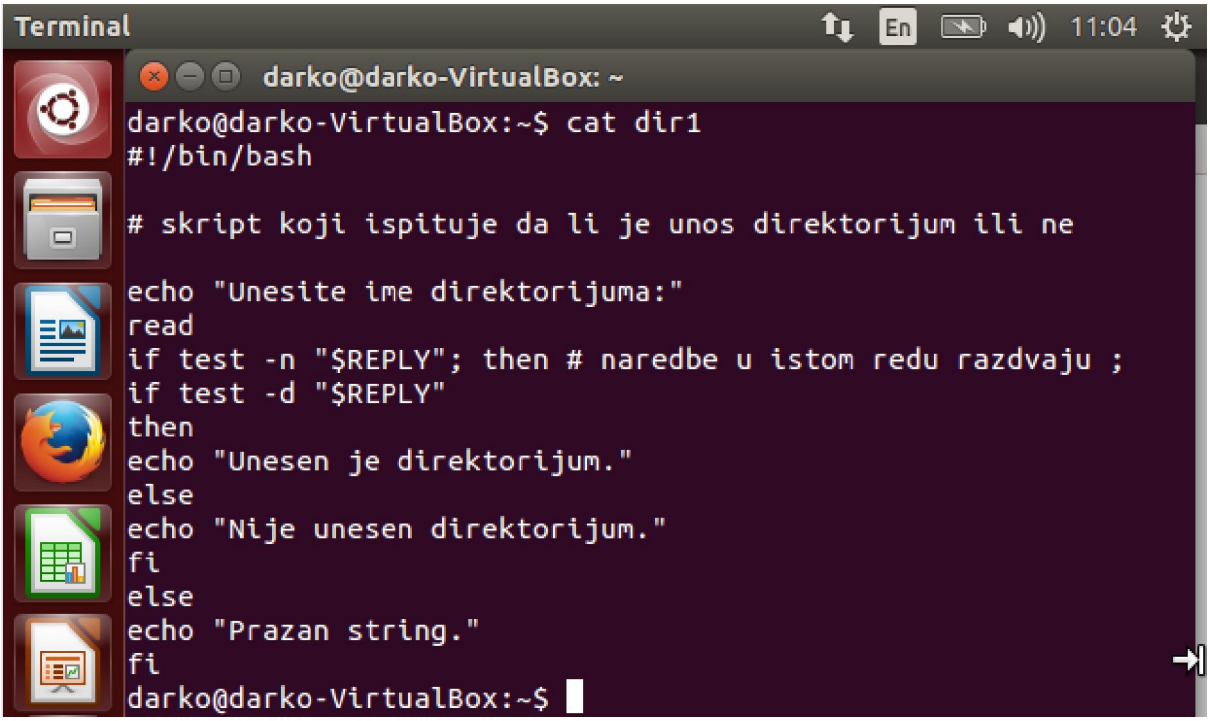
Slika 4.9.2.2. – Kod i rezultat skripta koji ilustruje operacije nad stringovima] i [. Ispituje se da li su jednaki i koji od stringova] ili [prednjači u ASCII tabeli. Uz < i > se koristiti \ jer uklanja redirekciju kod ovih znakova i omogućava da se oni koriste za poređenje.

4.9.3. Ispitivanje datoteka

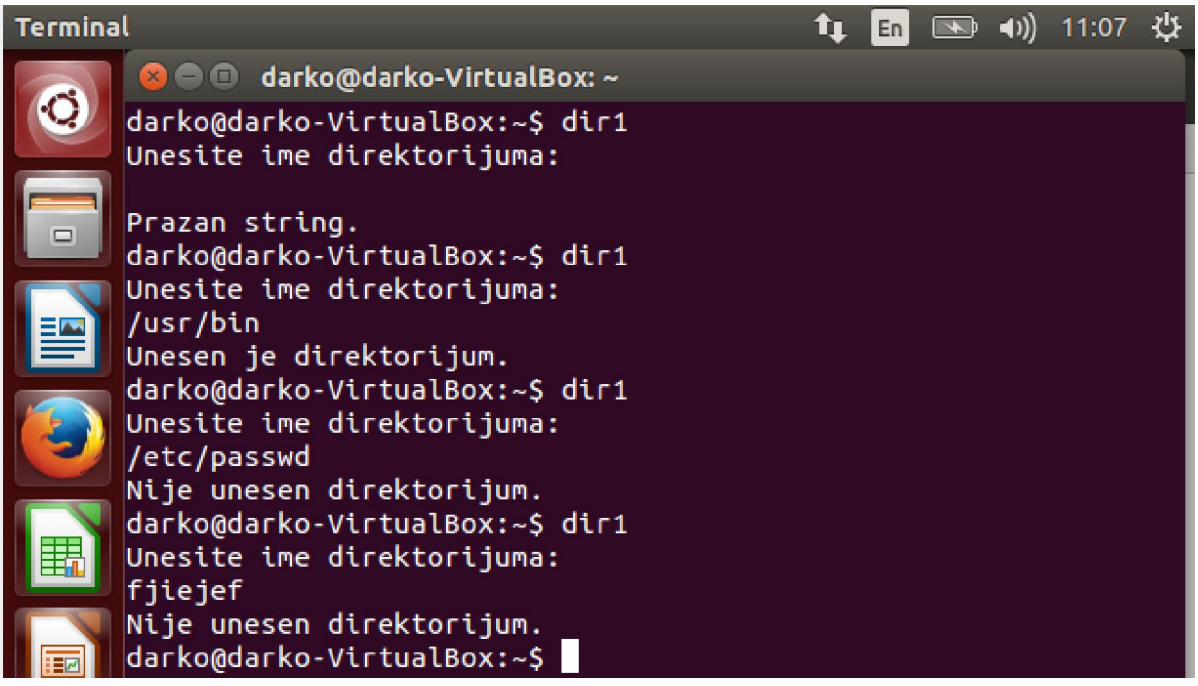
Neke od naredbi koje se koriste za ispitivanje datoteka su:

- -e datoteka ispituje da li datoteka postoji
- -f datoteka ispituje da li datoteka postoji i predstavlja običnu datoteku
- -s datoteka ispituje da li datoteka postoji i veća je od nule
- -x datoteka ispituje da li datoteka postoji i može se izvršavati/pretraživati
- -r datoteka ispituje da li datoteka postoji i može se čitati

- -d datoteka ispituje da li datoteka postoji i predstavlja direktorijum



Slika 4.9.3.1. – Skript koji ispituje da li je dati unos direktorijum ili ne

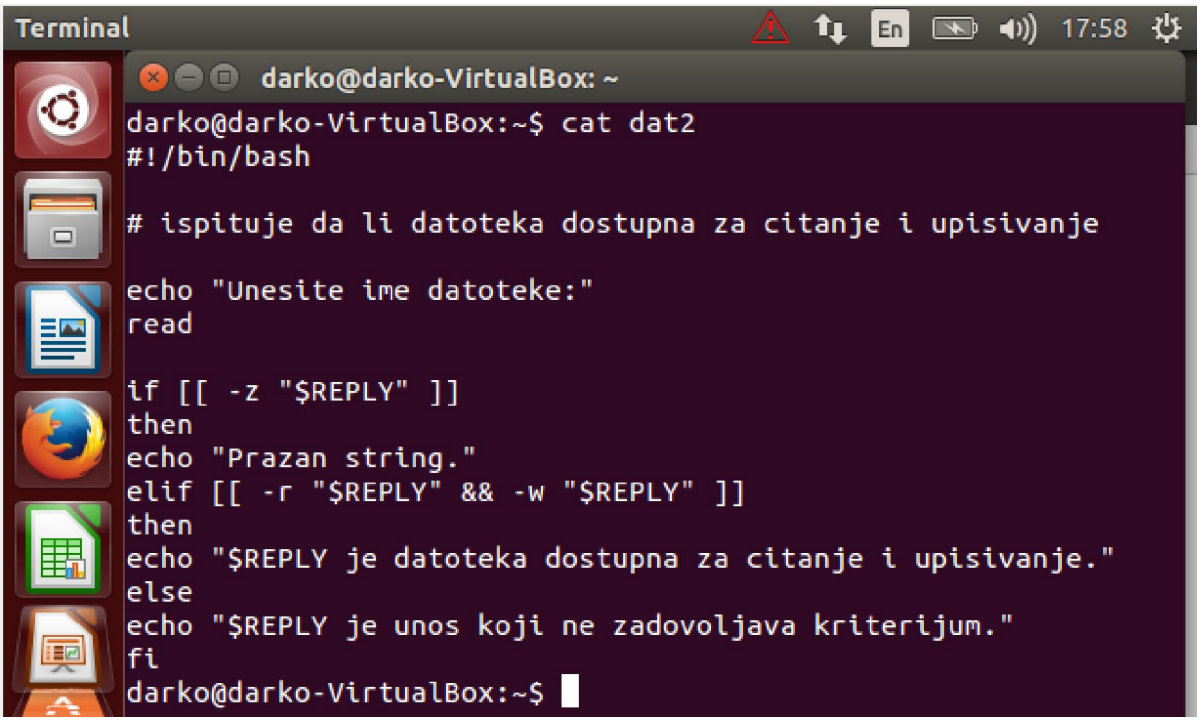


Slika 4.9.3.2. – Testiranje skripta koji provjerava da li je unos direktorijum ili ne

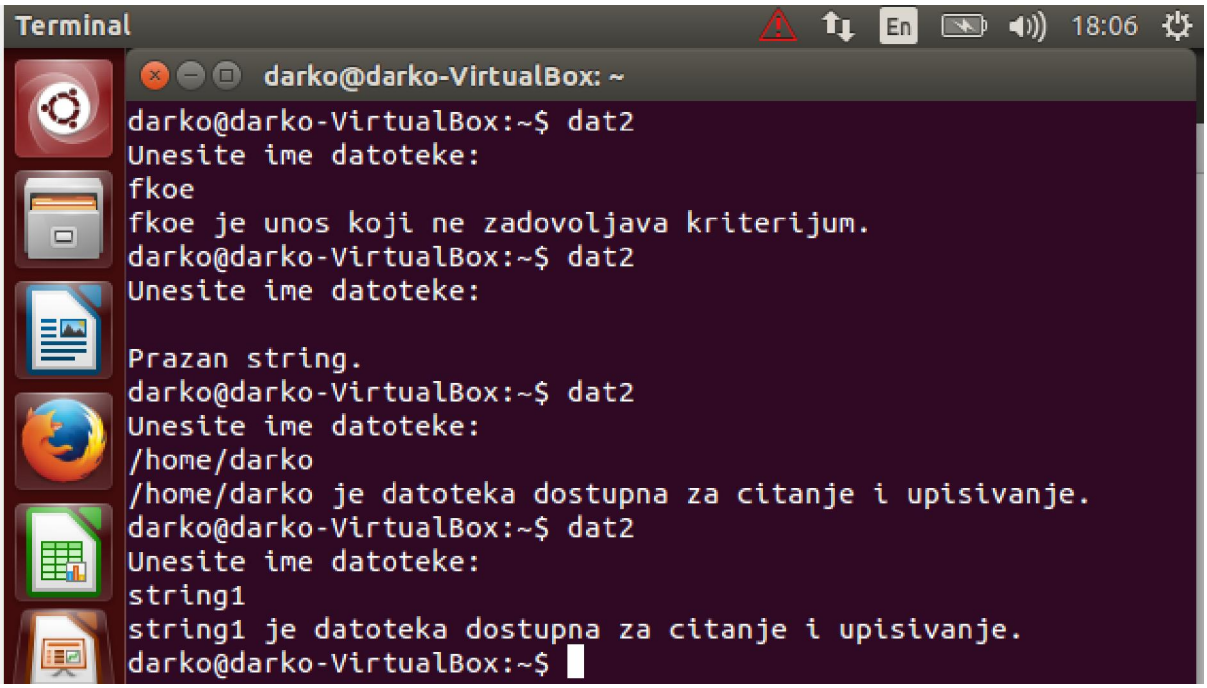
Ekvivalent zapisu test je naredba [ ]. Umjesto test *izraz* može se koristiti [*izraz*]. Oba zapisa daju identični rezultat a razlika je jedino u sintaksi.

4.10. Konstrukcija [[ ]]

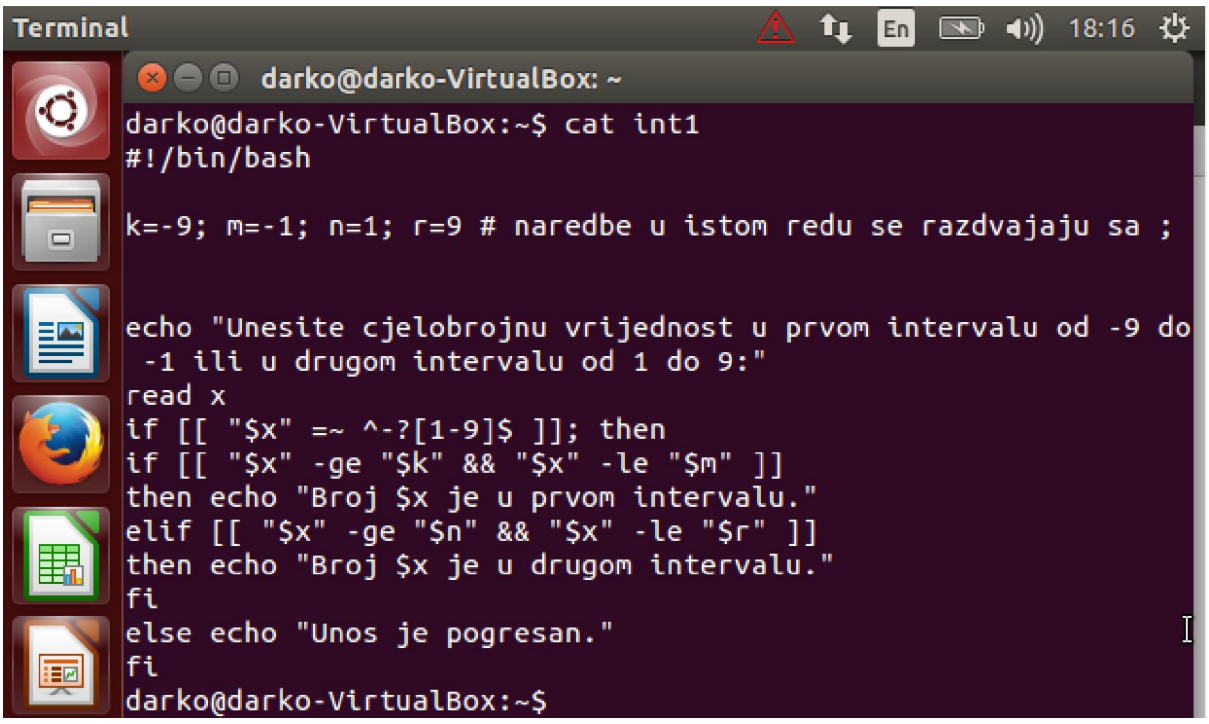
Konstrukcija [[*izraz*]] predstavlja poboljšanu verziju naredbe test. Dozvoljena je na primjer upotreba operatora && i || u sklopu [[*izraz*]] [3]. Navedeni operatori u sklopu naredbe test uzrokuju sintaksnu grešku [3]. Bitno poboljšanje predstavlja i mogućnost upotrebe proširenih regularnih izraza. Moguće je porediti zadati znakovni niz sa proširenim regularnim izrazom uz pomoć =~. U zavisnosti od uspješnosti poređenja vraća se 0 ili neka druga vrijednost.



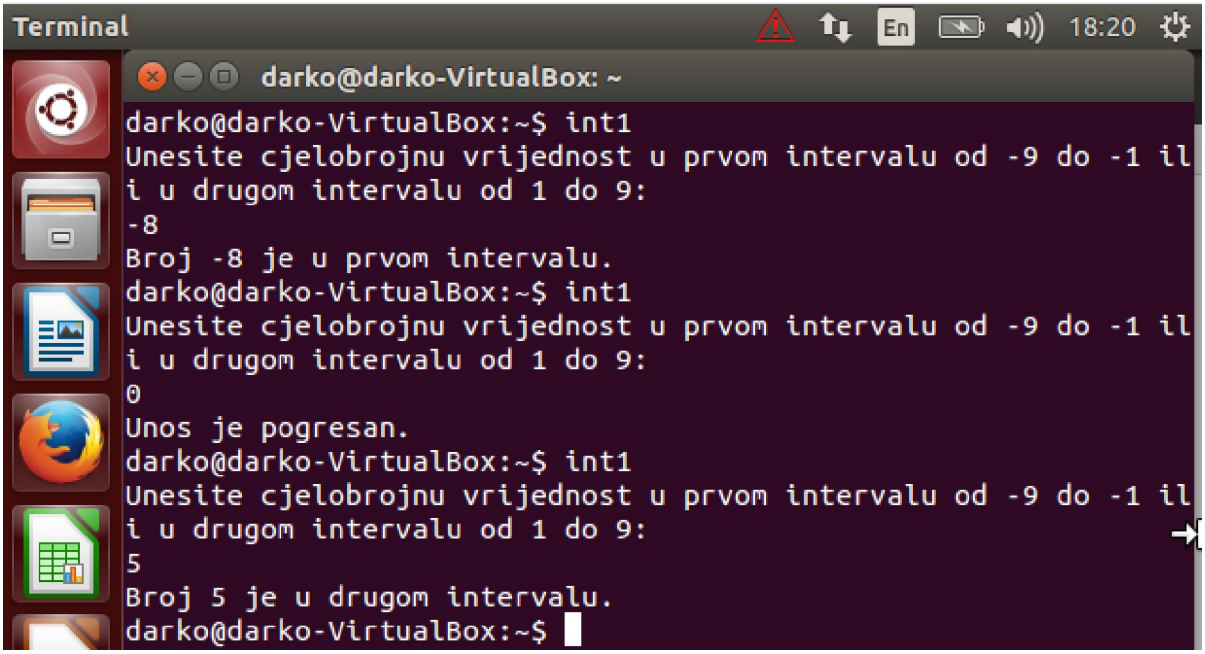
Slika 4.10.1. – Kod skripta koji provjerava da li je neka datoteka istovremeno dostupna korisniku za čitanje i upisivanje



Slika 4.10.2. – Testiranje skripta koji provjera da li je neka datoteka dostupna korisniku za čitanje i upisivanje. Važno je napomenuti da se i direktorijumi u unixolikim sistemima smatraju za datoteke koje sadrže druge datoteke i direktorijume.



Slika 4.10.3. – Kod skripta koji traži da se unese cijeli broj u ponuđenim intervalima. Važno je napomenuti da je ^-?[1-9]$ šablon koji predstavlja brojeve koji se nalaze u ponuđenim intervalima. Znak ^ predstavlja početak šablona, $ je kraj. Dio -? opciono pojavljivanje znaka -, a [1-9] predstavlja brojeve od 1 do 9.



Slika 4.10.4. – Testiranje skripta, ukoliko unos zadovoljava šablon brojeva u ponuđenim intervalima vrši se utvrđivanje kom intervalu pripada. Ako ne, ispisuje se poruka o pogrešnom unosu.

4.11. case naredba

Konstrukcija koja služi, da u slučaju kada postoji mnogo elif naredbi, pojednostavi pisanje. Ova naredba ima sledeći oblik:

case "$var" in

prvi_slucaj)
blok1
;;

drugi_slucaj)
blok2
;;

treci_slucaj)
blok3
;;

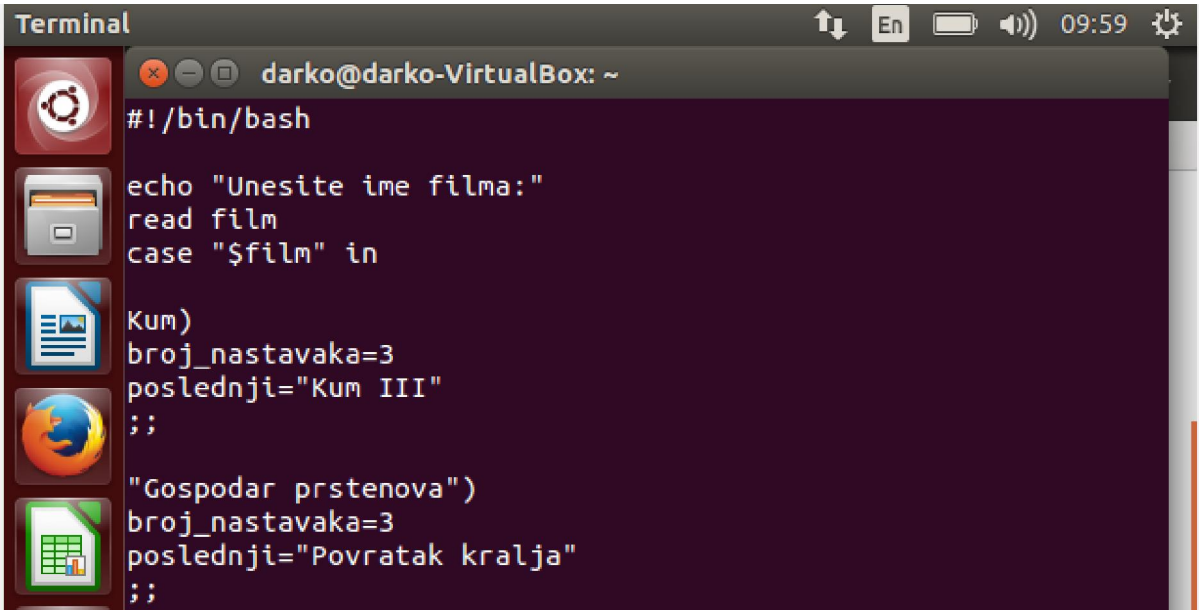
cetvrti_slucaj)
blok4

;;

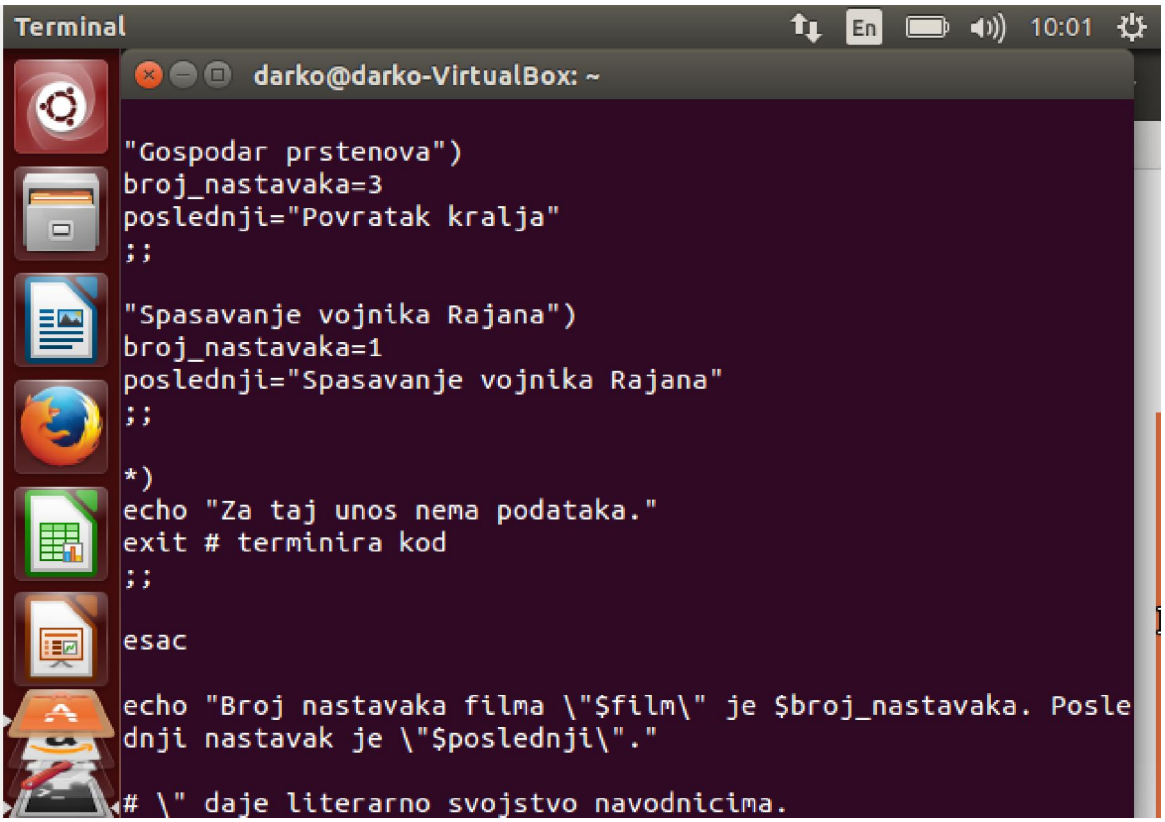
.
.
.
nti_slucaj)
blokn

;;
esac

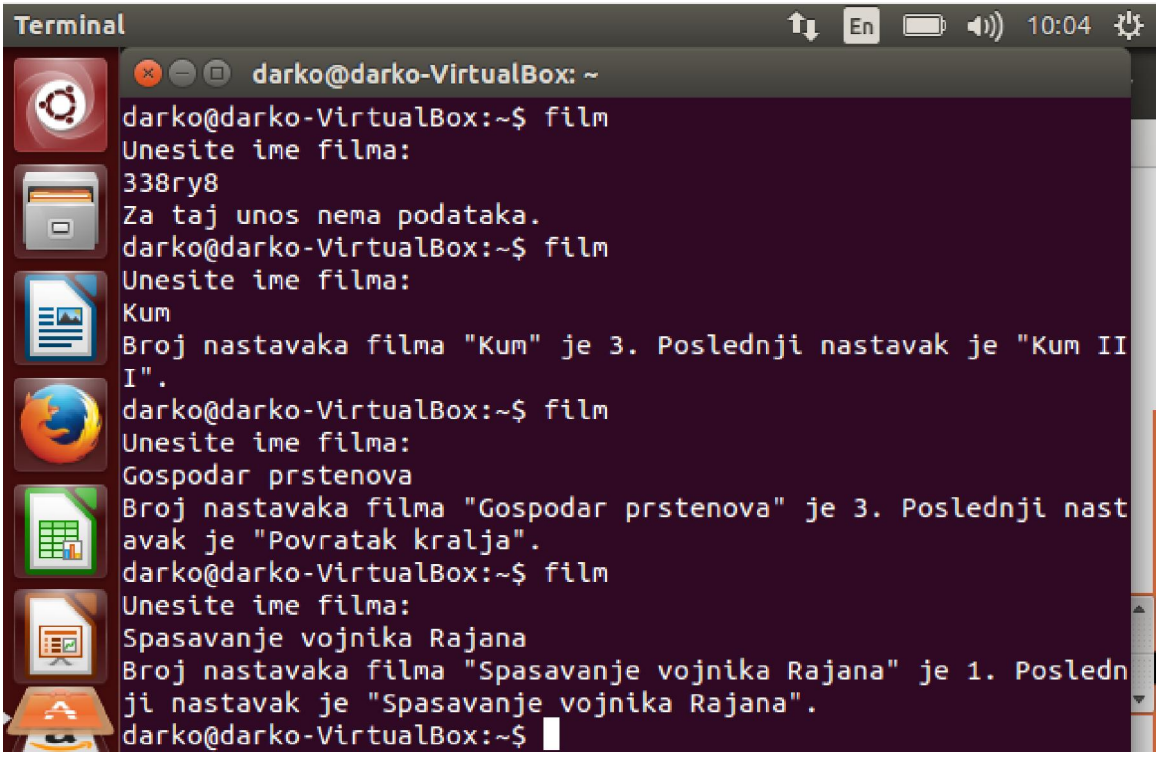
Sadržaj promjenljive *var* se poredi sa slučajevima od 1 do n i na osnovu uspješnosti se izvršava odgovarajući blok naredbi.



Slika 4.11.1. – Prvi dio skripta film koji od korisnika traži da unese ime filma. Taj unos se smješta u promjenljivu *film*. Sadržaj promjenljive film se poredi sa filmovima u bazi i na osnovu rezultata poređenja na ekranu se daje adekvatan ispis.



Slika 4.11.2. – Nastavak koda skripta film



Slika 4.11.3. - Testiranje skripta film

4.12.while

Ova petlja izvršava blok naredbi sve dok je njen uslov zadovoljen. Ima sledeći oblik:

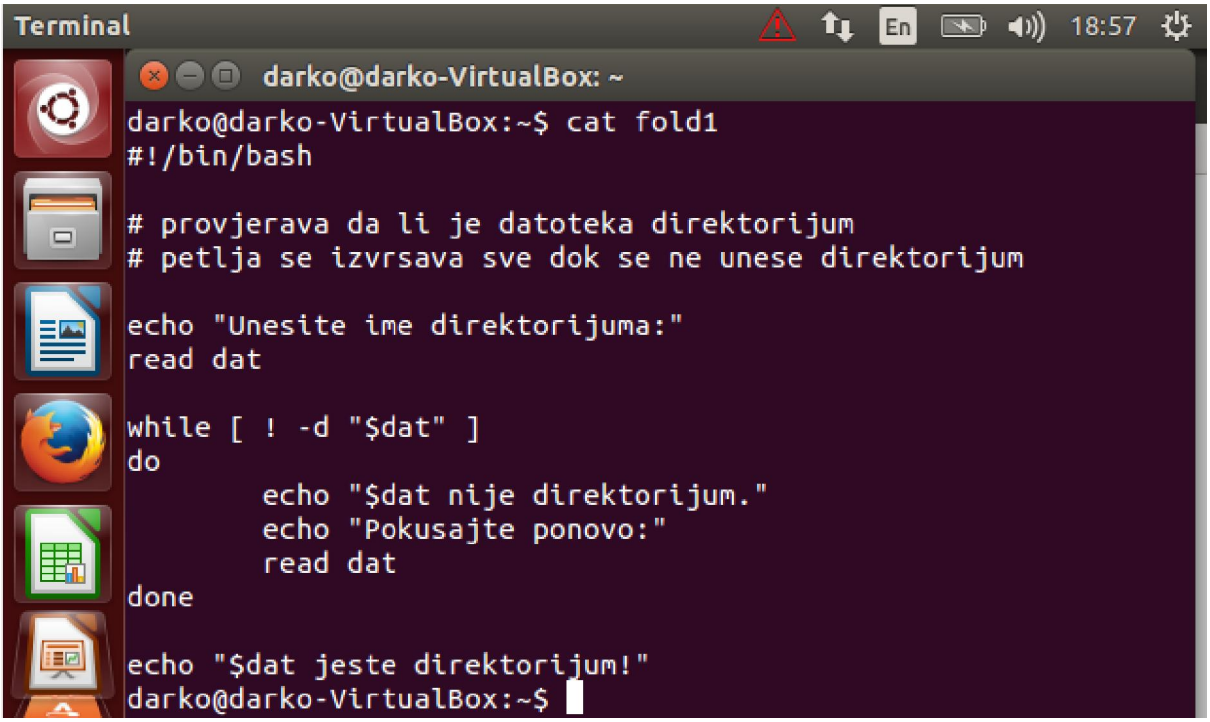
```
while komanda       # sve dok je izlazni status komanda nula ponavljaće se petlja
do
      blok naredbi
done
```

Kao i if naredba najčešće se koristi sa test.

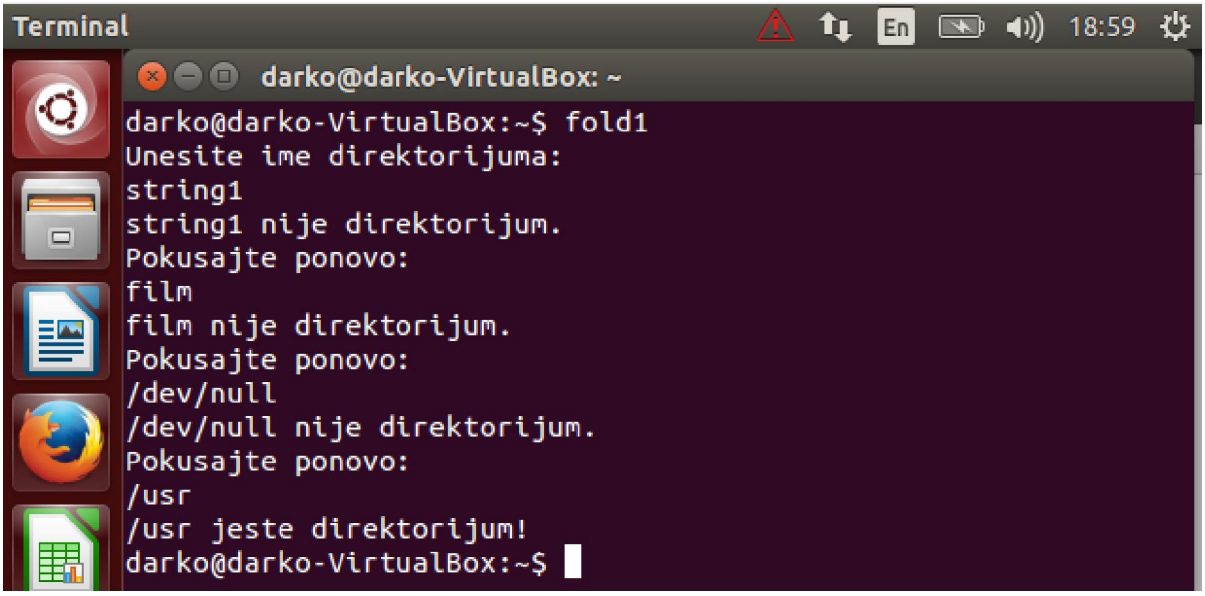
```
while test izraz       # sve dok je izlazni status test izraz nula izvršava se
do
      blok naredbi
done
```

Može da se koristi i sa svestranom [[ ]] konstrukcijom [3]. Tada ima oblik:

```
while [[ izraz ]]
do
      blok naredbi
done
```



Slika 4.12.1. – Kod skripta fold1 koji od korisnika traži da unese direktorijum. Petlja se izvršava sve dok korisnik ne unese direktorijum.



Slika 4.12.2. – Testiranje skripta fold1koji od korisnika traži da unese direktorijum

4.13. for

Ova petlja ima sledeći oblik:

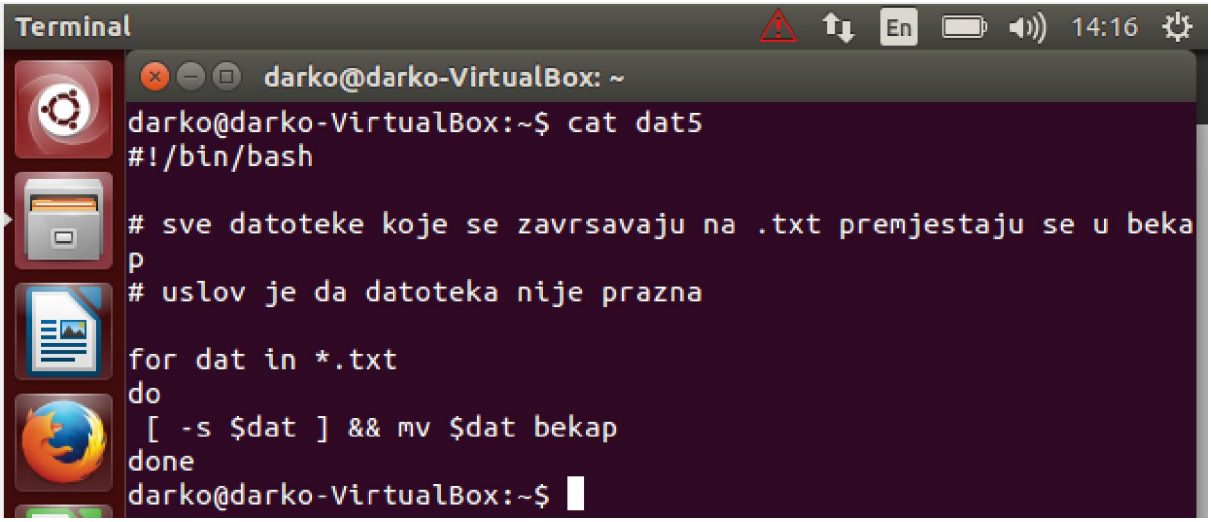
for var in list # promjenljiva var preuzima sve vrijednosti iz list

do

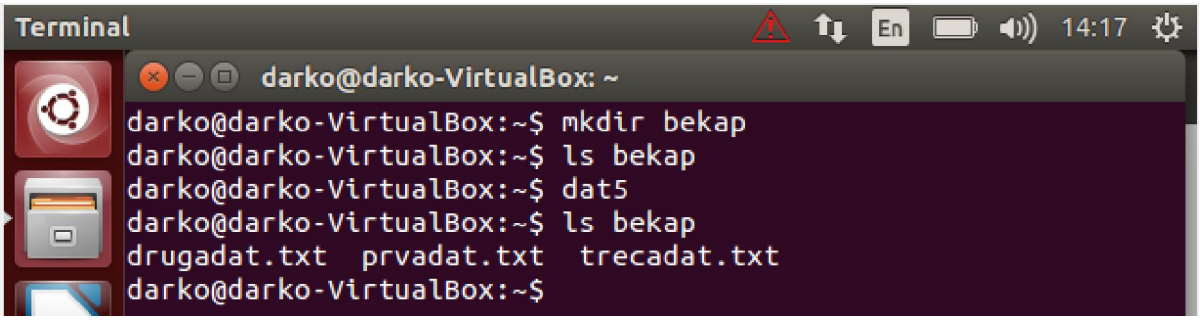
blok naredbi

done

Najčešće se koristi u radu sa datotekama.



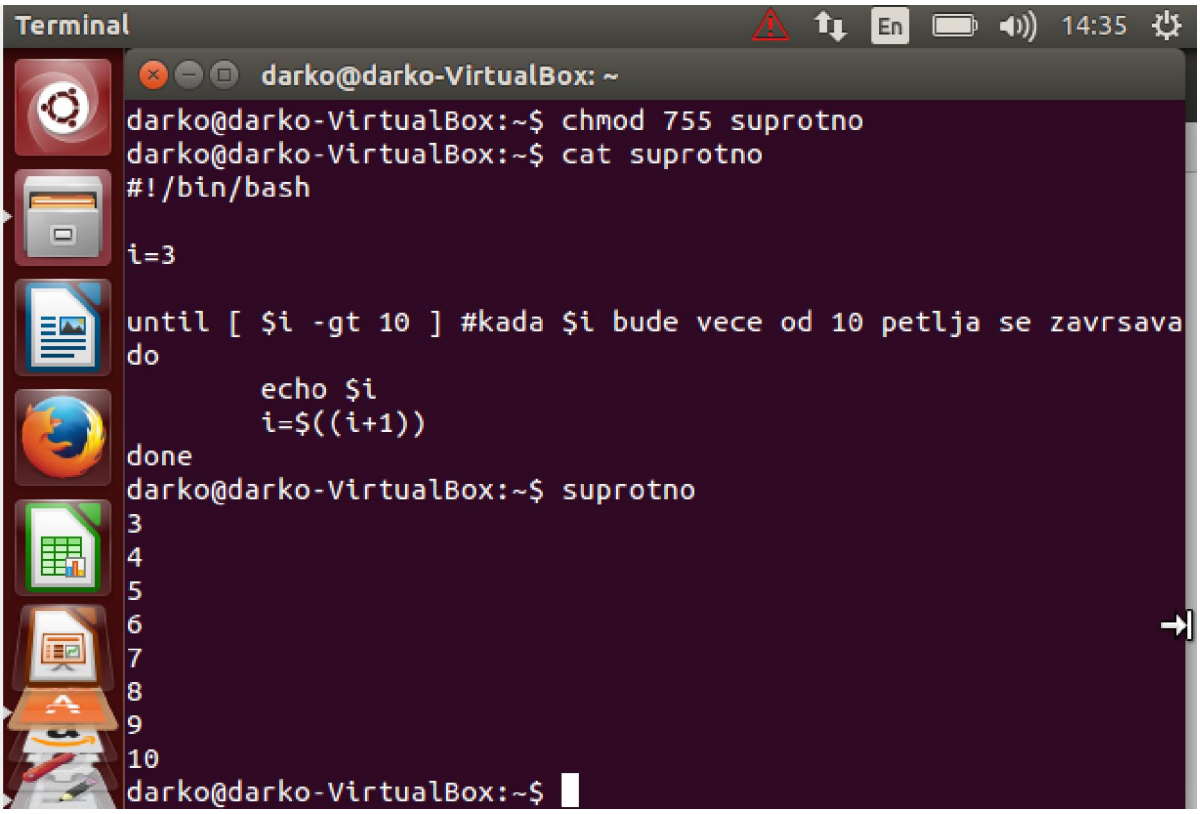
Slika 4.13.1. – Kod skripta dat5 koji premješta datoteke koje nisu prazne i koje se završavaju na .txt u direktorijum bekap



Slika 4.13.2. – Testiranje skripta dat5

4.14. until

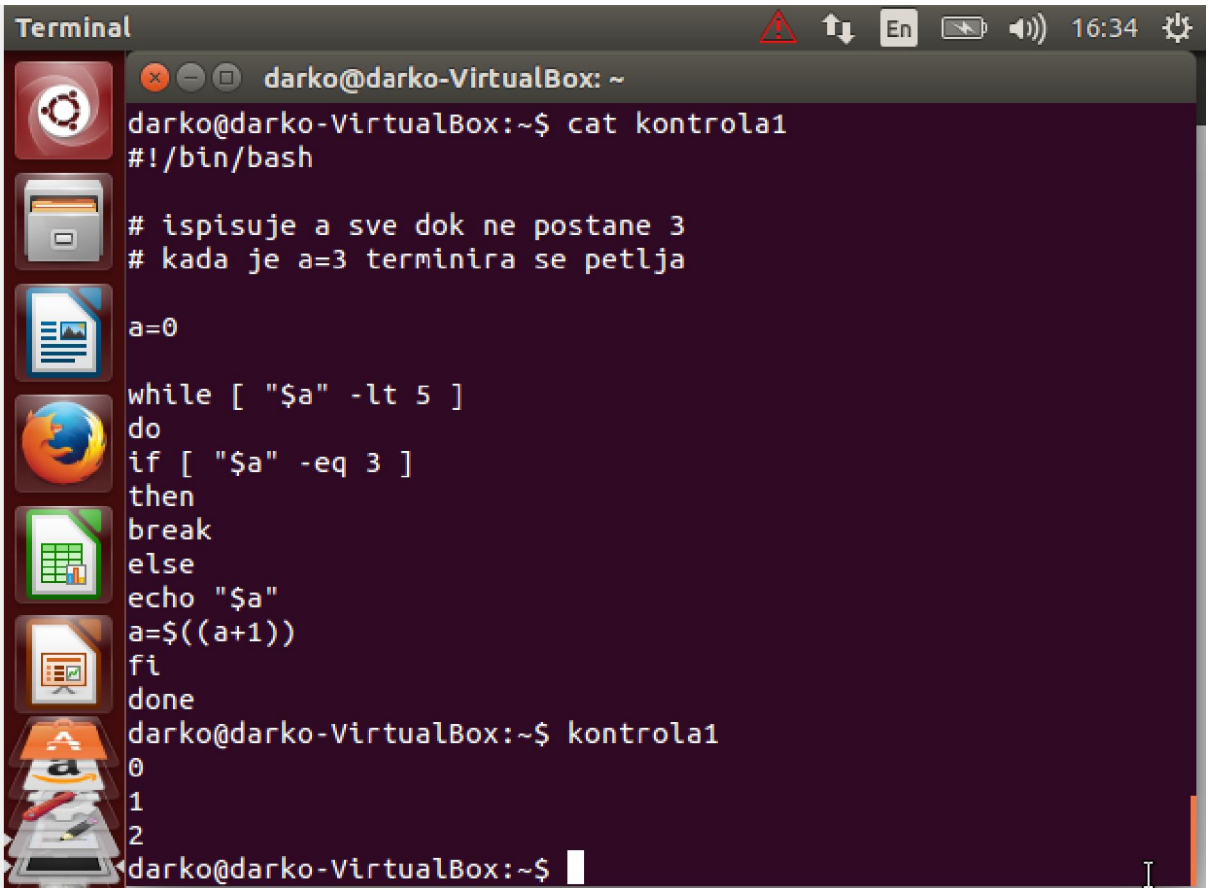
Ova petlja ima isti oblik kao i while. Suštinska razlika je u tome sto se until izvršava sve dok je izlazni status komande u uslovu različit od nula.



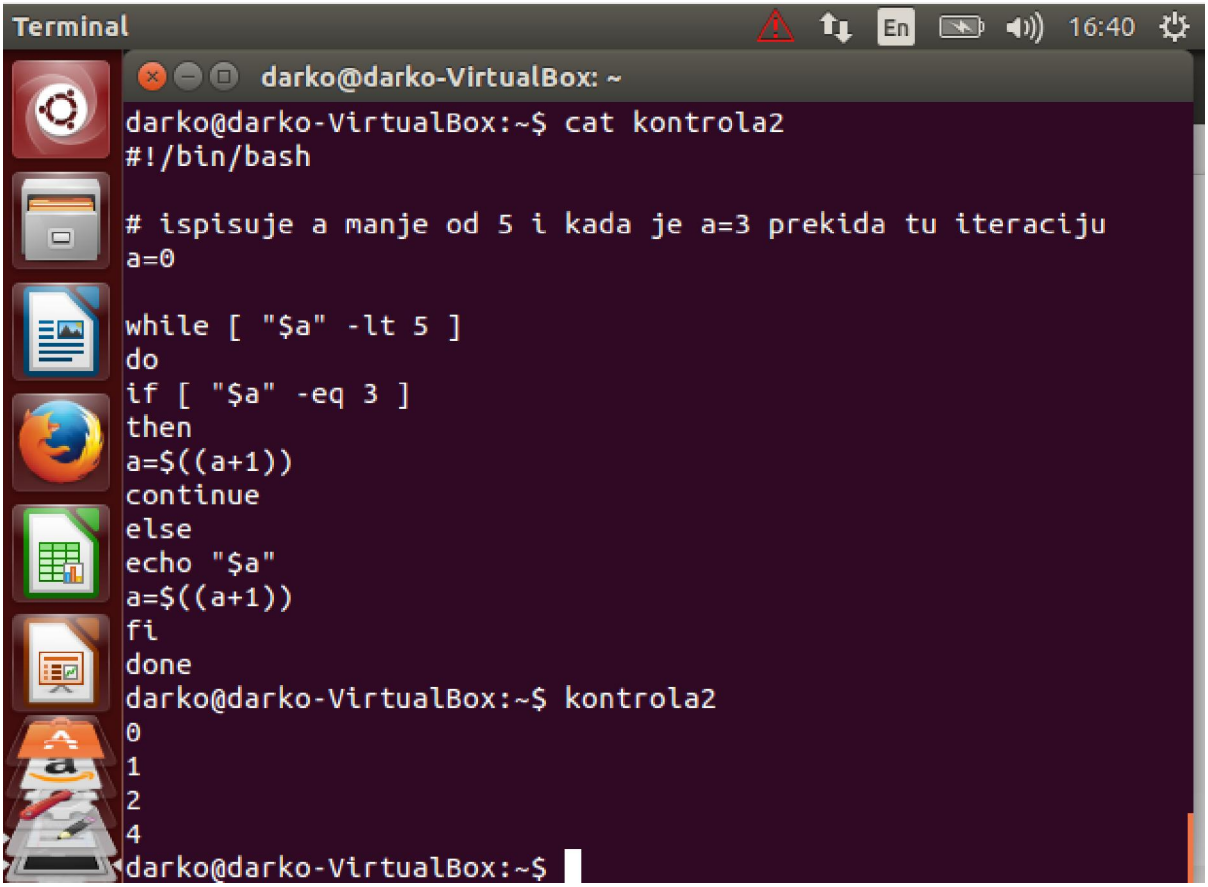
Slika 4.14.1. – Kod skripta koji ispisuje brojne vrijednosti koristeći until petlju

4.15. break i continue

Ove naredbe se koriste za kontrolu toka petlje. Naredba break trenutno terminira petlju. Naredba continue prekida tekuću iteraciju i prelazi na sledeću.



Slika 4.15.1. – Kod skripta koji ilustruje primjenu naredbe break



Slika 4.15.2. – Kod skripta koji ilustruje primjenu naredbe continue

4.16. Funkcije

4.16.1. Definisanje i pozivanje funkcija

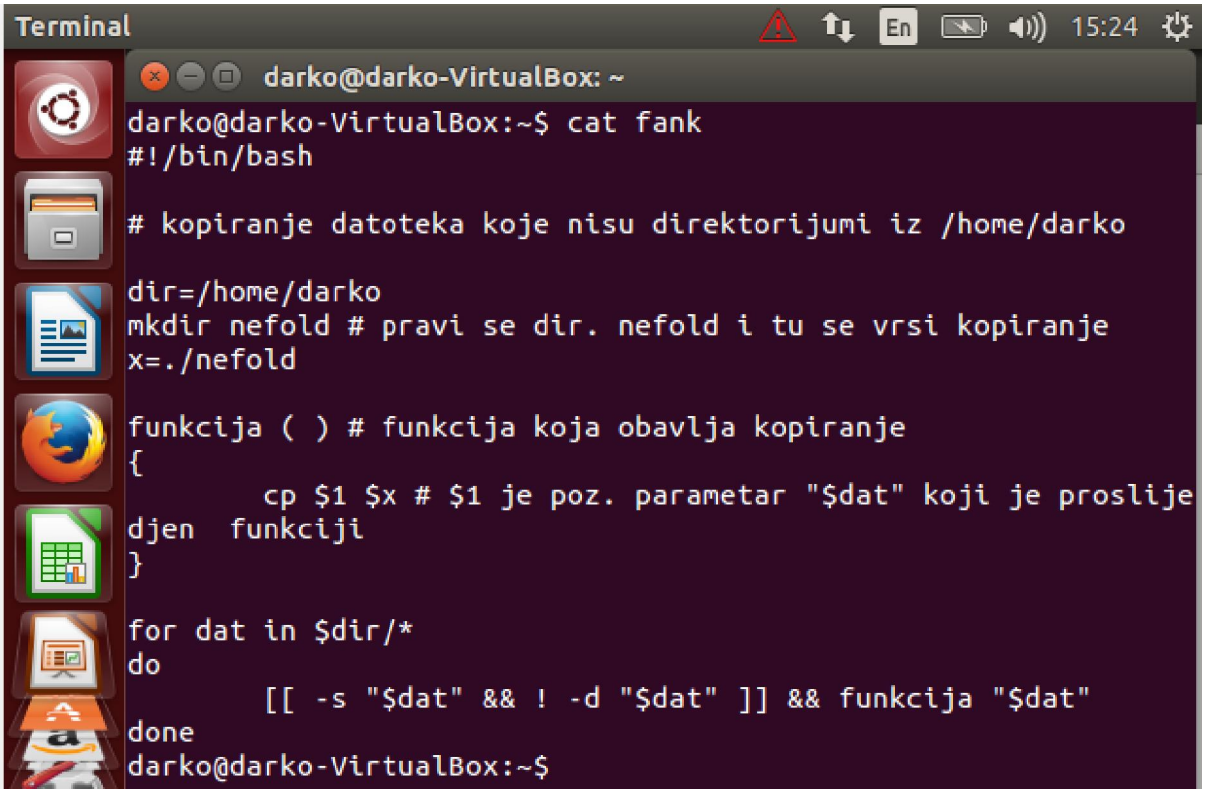
Funkcija predstavlja blok naredbi koji se može izvršiti iz bilo kog dijela programa jednostavno navodeći njeno ime. Ima sledeću strukturu:

ime_funkcije ( )
{
blok naredbi
}

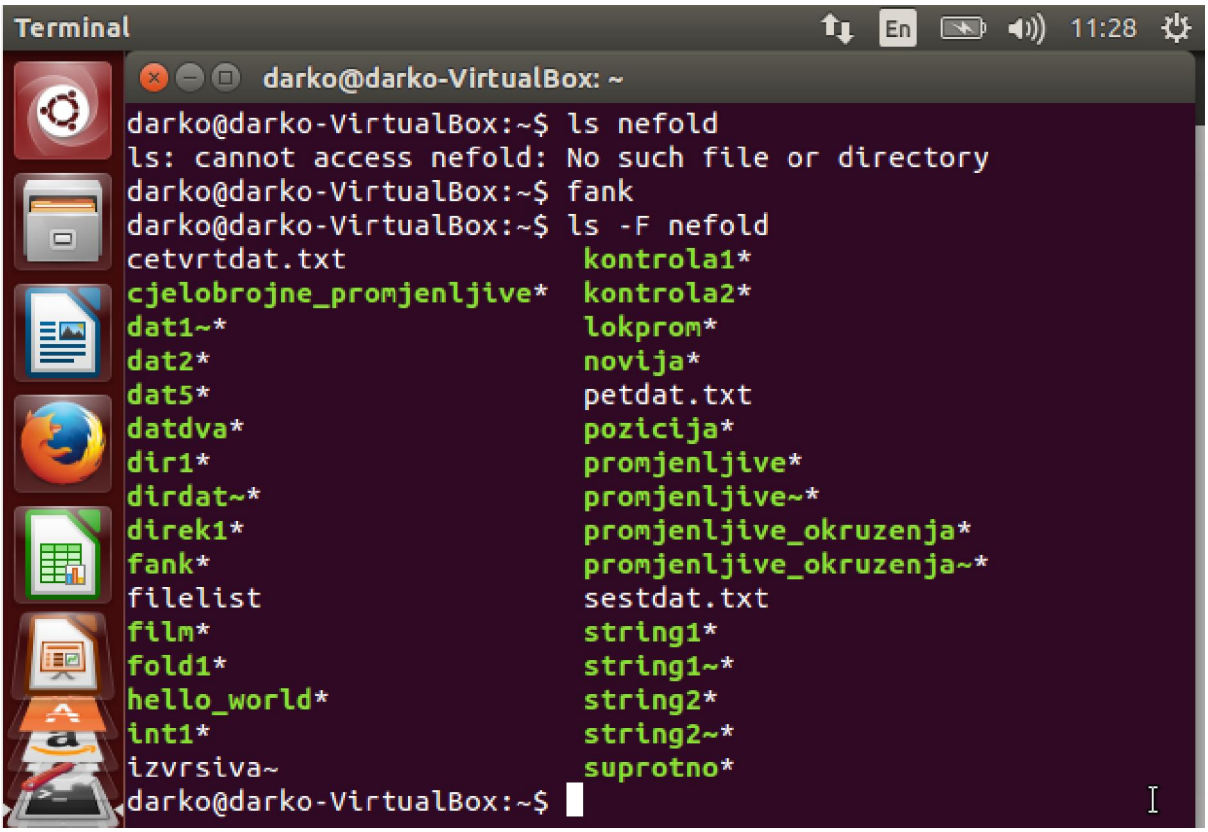
Ova funkcija se poziva unošenjem *ime_funkcije*. Pravila za davanje imena funkcijama su ista kao i pravila za davanje imena promjenljivima.

4.16.2. Pozicioni parametri

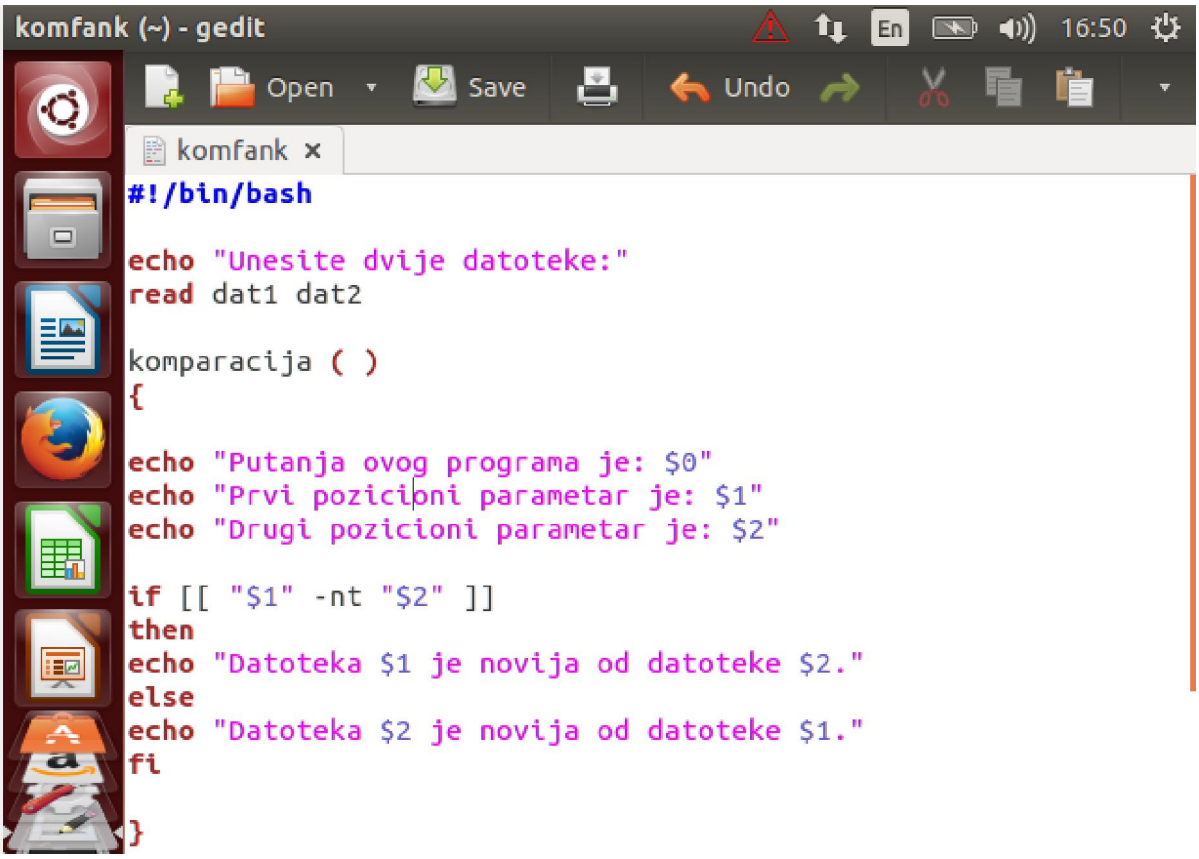
Ljuska ima niz promjenljivih zvanih pozicioni parametri, koje sadrže pojedinačne riječi na komandnoj liniji [1]. Označavaju se brojevima od 0 do 9. Za pozicione parametre iznad 9 koriste se vitičaste zagrade. Promjenljiva $# označava ukupan broj pozicionih parametara.



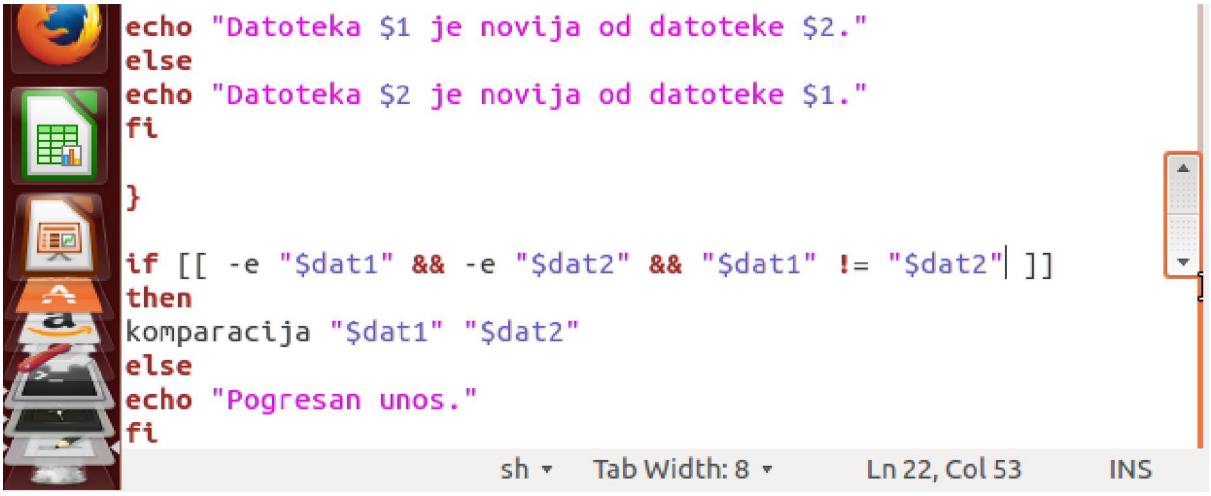
Slika 4.16.2.1. – Kod skripta fank. Definisana je funkcija koja vrši željeno kopiranje, nakon čega sledi for petlja u glavnom programu. Funkcija mora biti definisana prije glavnog programa.



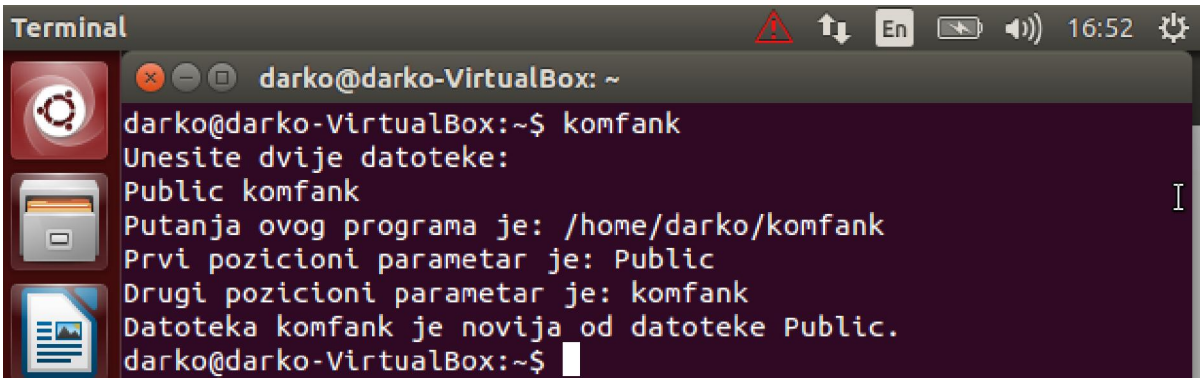
Slika 4.16.2.2. – Pozivanje skripta fank. Skript je kreirao direktorijum nefold i u njega kopirao sve datoteke koje nisu prazne i koje nisu direktorijumi.



Slika 4.16.2.3. – Prvi dio skripta komfank. Od korisnika se traži da unese imena dviju datoteke. Definisana je funkcija komparacija koja ispituje koja je od unesenih datoteka novija. Ova funkcija ispisuje $0 parametar koji označava putanju skripta komfank. Takođe, ispisuje i dva parametra $1 i $2. Ovi parametri su imena dviju datoteka koje su proslijeđene funkciji komparacija na obradu.



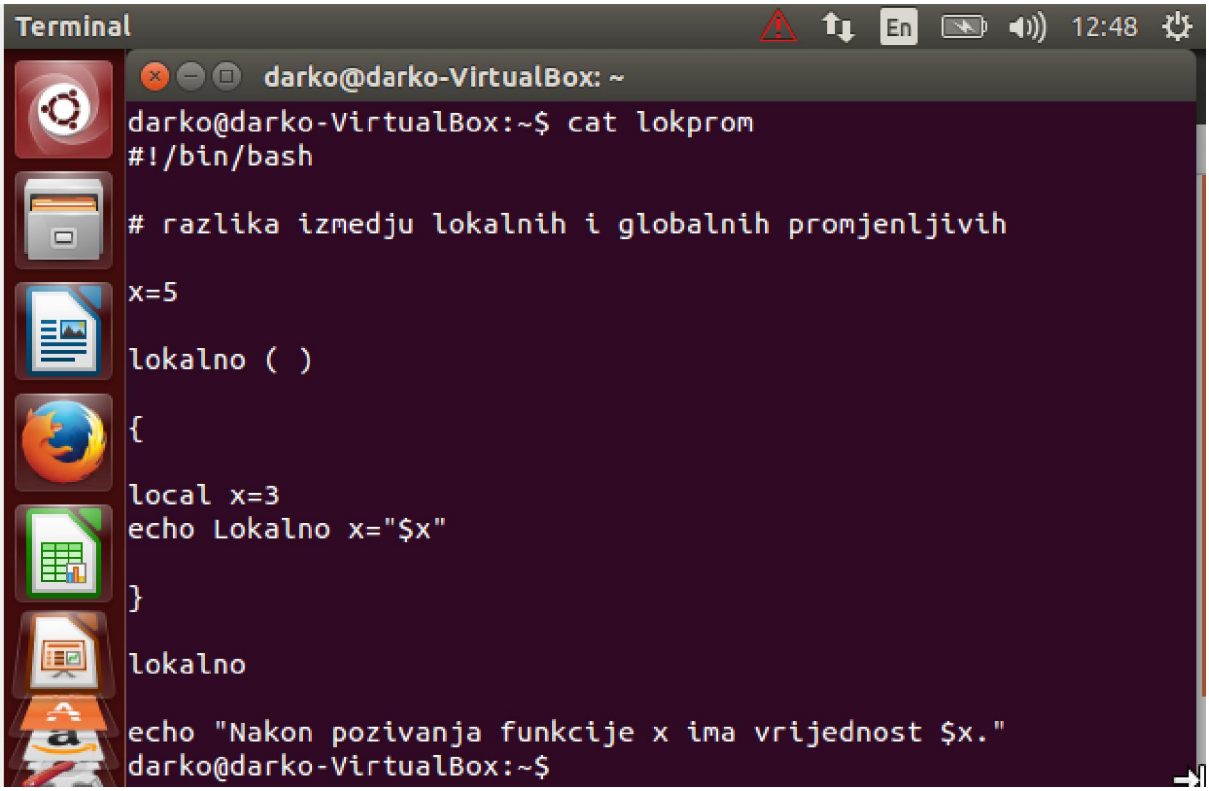
Slika 4.16.2.4. – Drugi dio skripta komfank. U glavnom programu se ispituje da li dati unosi predstavljaju datoteke i pritom moraju da budu različiti da bi se komparacija uspješno izvršila. Ako je testiranje uspješno, poziva se funkcija sa argumentima koji su unijeti sa tastature. Ako nije, ispisuje se poruka o pogrešnom unosu.



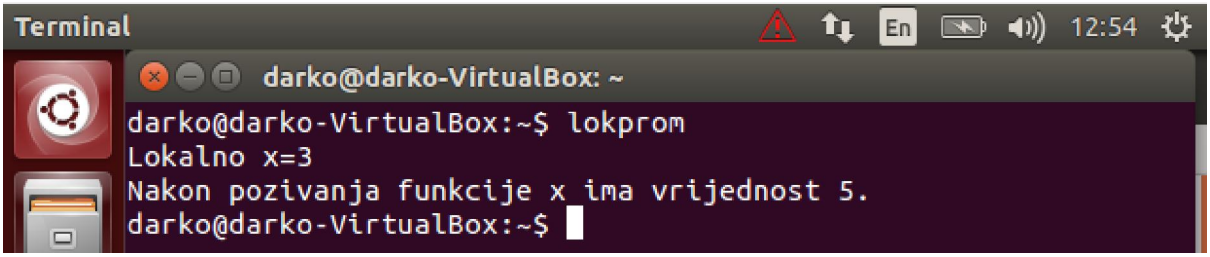
Slika 4.16.2.5. - Testiranje skripta komfank

4.16.3. Lokalne promjenljive

To su promjenljive koje važe samo u funkcijama. Ako se u funkciji želi napraviti promjenljiva *x* to se postiže unošenjem local x. Ova promjenljiva *x* će važiti samo u datoj funkciji. Ako u glavnom programu postoji globalna promjenljiva *x* njenu vrijednost neće biti moguće promijeniti korišćenjem lokalne promjenljive *x*.



Slika 4.16.3.1. – Kod skripta lokprom. Ilustruje razliku između lokalnih i globalnih promjenljivih.



Slika 4.16.3.2. – Izvršavanje skripta lokprom

5. ZAKLJUČAK

Najveća prednost pisanja komandnih skriptova leži u tome je to što je sintaksa ista kao da se komande unose sa komandne linije [4]. Često je mnogo brže napisati skript, nego kod sa identičnim učinkom, u nekom drugom programskom jeziku. Dobra strana je što komandni skriptovi omogućavaja automatizovanje operacija sa datotekama, koje bi se inače ručno obavljale.

Mnoge komande kao npr. cd i cp se razlikuju u samo jednom slovu pa pogrešan unos može prouzrokovati probleme. Zloupotreba znaka > može obrisati neku važnu datoteku u sistemu. Unošenjem >*datoteka* sadržaj datoteke *datoteka* će biti u potpunosti obrisan. Komandni skriptovi nisu namijenjeni za složene aritmetičke proračune. Ovi skriptovi su predviđeni da budu kratki. Pisanje dugačkih skriptova nije nešto što je poželjno.

LITERATURA

[1] Linux s komandne linije, William E. Shotts, Mikro knjiga

[2] Kako radi Linux, Brian Ward, Mikro knjiga

[3] Advanced Bash-Scripting Guide, Mendel Cooper

[4] https://en.wikipedia.org/wiki/Shell_script#Other_scripting_languages