**UNIVERZITET SINGIDUNUM**

**ADMINISTRACIJA I POSTAVKA LINUX SERVERA**

**-diplomski rad –**

**Mentor: Kandidat:**

Prof.dr Aleksandar Jevremović Aleksandar Milutinović

**Beograd, 2022.**

# Sadržaj

[Sadržaj 1](#_Toc119955620)

[Uvod 4](#_Toc119955621)

[1. *UNIX* kao preteča *Linux* 5](#_Toc119955622)

[2. *Linux* distribucije 8](#_Toc119955623)

[3. Struktura *Linux* sistema i dozvole 10](#_Toc119955624)

[3.1 Direktorijumi 10](#_Toc119955625)

[3.2 Dozvole u *Linux* sistemima 13](#_Toc119955626)

[3.2.1 Grupe dozvola 13](#_Toc119955627)

[3.2.2 Promena dozvola i komande 13](#_Toc119955628)

[3.2.3 Funkcija *setuid* 14](#_Toc119955629)

[3.2.4 Funkcija *setgid* 14](#_Toc119955630)

[3.2.5 Funkcija promene grupe i vlasnika 15](#_Toc119955631)

[3.2.6 Lepljivi bit 15](#_Toc119955632)

[3.3 Objašnjenje *sudo* komande 16](#_Toc119955633)

[4. Instalacija *CentOS* operativnog sistema 18](#_Toc119955634)

[4.1 Opšta konfiguracija 19](#_Toc119955635)

[5. Podešavanje udaljene konekcije SSH klijentom 25](#_Toc119955636)

[5.1 Opšta konfiguracija 25](#_Toc119955637)

[6.Automatizacija pravljenja virtualnog okruženja 30](#_Toc119955638)

[6.1 Uvod 30](#_Toc119955639)

[6.2 Konfiguracija virtualnog okruženja 31](#_Toc119955640)

[7. Daljinsko administriranje pomoću SSH 33](#_Toc119955641)

[7.1 Generisanje i upotreba *ssh-keygen* 34](#_Toc119955642)

[8. *Networking/DNS* 36](#_Toc119955643)

[8.1 A record 39](#_Toc119955644)

[8.2 AAAA rekord 40](#_Toc119955645)

[8.3 CNAME rekord 41](#_Toc119955646)

[8.4 MX rekord 41](#_Toc119955647)

[8.5 PTR rekord 42](#_Toc119955648)

[8.6 NS rekord 43](#_Toc119955649)

[8.7 SOA rekord 43](#_Toc119955650)

[8.8 SRV rekordi 44](#_Toc119955651)

[8.9 TXT rekordi 45](#_Toc119955652)

[9. Apache veb server i baze podataka 46](#_Toc119955653)

[9.1 Instalacija httpd (Apache) na CentOS 46](#_Toc119955654)

[9.1.1 Osnove Apache konfiguracije 47](#_Toc119955655)

[9.1.2 Funkcionalnost Apache konfiguracije 49](#_Toc119955656)

[9.2 MySQL, Maria DB baze podataka 49](#_Toc119955657)

[9.2.1 Podešavanje sigurne instalacije 50](#_Toc119955658)

[9.2.2 Listanje, kreiranje i odabir podataka i tabela 51](#_Toc119955659)

[10. Pravljenje osnovne konfiguracije domene upotrebom WordPress CMS 53](#_Toc119955660)

[10.1 Uvod 53](#_Toc119955661)

[10.2 Instalacija Wordpress aplikacije 53](#_Toc119955662)

[11. Shell skripte 58](#_Toc119955663)

[11.1 Uvod 58](#_Toc119955664)

[11.2 Početak pisanja skripti 58](#_Toc119955665)

[11.2.1 Varijable 59](#_Toc119955666)

[11.2.2 Uzimanje unosa iz terminala 60](#_Toc119955667)

[11.2.3 Kondicionali 62](#_Toc119955668)

[11.2.4 Petlje 64](#_Toc119955669)

[11.3 Regularni izrazi 65](#_Toc119955670)

[11.4 Sed 66](#_Toc119955671)

[11.5 AWK 69](#_Toc119955672)

[12. Firewall i nadgledanje sistema 70](#_Toc119955673)

[12.1 Uvod 70](#_Toc119955674)

[12.2 Filtriranje paketa 70](#_Toc119955675)

[12.3 IP lanci u firewall konfiguracijama 70](#_Toc119955676)

[12.3.1 Upravljanje pravilima IP lanca 71](#_Toc119955677)

[12.4 IP tabele u firewall konfiguracijama 72](#_Toc119955678)

[12.4.1 Upravljanje pravilima IP tabela 73](#_Toc119955679)

[12.5 Nadgledanje sistema 74](#_Toc119955680)

[Zaključak 78](#_Toc119955681)

[Literatura 79](#_Toc119955682)

# Uvod

Glavni problem koji se razmatra u ovom radu jeste upotreba *Linux* servera, kao i njegova administracija, koristeći različite metode za rešavanje problema koji se javljaju u ovom polju.

Glavni cilj istraživanja jeste da se predstavi, a i olakša, upotreba i administracija *Linux* sistema budućim *Linux* administratorima koji bi čitali ovaj rad.

Takođe, još jedan cilj jeste da se približi razumevanje ovih operativnih sistema iz prostog razloga što je izuzetno moćan i omogućava korisniku potpunu slobodu, što nemamo u *Windows* operativnim sistemima.

Tehnologije koje će se koristiti su *Virtual Box* okruženje za virtualizaciju, *Vagrant* koji omogućava jednostavnu automatizaciju kreacije servera, *shell* skripte pisane za specifične zadatke, operativni sistem *CentOS 7*.

U ovom radu, koristiće se funkcionalni pristup kroz niz primera i upotreba različitih alata na *Linux* operativnom sistemu i prikazati različiti problemi sa kojima se neki *Linux* sistemski administrator može susresti na serveru.

# 1. *UNIX* kao preteča *Linux*

Kao preteča za *Linux* operativne sisteme bio je *UNIX*, koji je još uvek aktuelan operativni sistem. Možemo ga naći u ekosistemu *Apple* kompanije, kao neki od značajnih konkurenata *Windows* kompanije, gde je *macOS* (*Apple*) operativni sistem koji koristi *UNIX*.

*UNIX* je nastao 1969. godine i njegov nastanak se pridaje Denisu Ričiju (Dennis Ritchie) i Kenu Tomsonu (Ken Thompson). Operativni sistem je nastao u Belovim laboratorijama (Bell Labs) i kao osnova korišćen je *Multicsa*, prethodno neuspeli projekat višekorisničkog operativnog sistema.

U početku, ovaj operativni sistem nije imao podršku većih kompanija, a i sam operativni sistem nije bio na zavidnom nivou kao što je danas. *UNIX* je inicijalno bio sistem koji je mogao da radi samo na jednom zadatku, suprotno *Multicsa* koji je mogao da vrši više zadataka, za šta je i bio inicijalno razvijan. Porastom kompleksnosti upotrebe ovog operativnog sistema, pošto je zamenio tada korišćeni operativni sistem *DEC*, moralo je da se nađe rešenje kako bi se omogućila skalabilnost ovog operativnog sistema. ( Operativni sistemi, Popović, Branović, Šarac, Beograd 2011, 128)

Godine 1973. operativni sistem je napisan u jeziku *C*, što je omogućilo prenosivost koda na buduće verzije. Prva verzija, koja je bila u upotrebi van Belovih (Bell) laboratorija, bila je pod nazivom *V6* što je bilo šesto izdanje već postojećeg operativnog sistema.

Nakon inicijalnih primena, druge značajnije kompanije kao što su *AT&T* su stvorile svoje varijacije na postojeći dizajn. *AT&T* verzija koja je razvijena bila je *System V Release 4 (SVR4)* 1982. godine.

Sam operativni sistem razvijao se zavidnom brzinom iz zbog dostupnosti izvornog koda i jednostavnosti dizajna. Poznatije varijante nastale su na Univerzitetu u Berkliju pod nazivom *Berkley Software Distributions (BSD)*.

U ovim verzijama *UNIX* operativnih sistema javlja se virtualna memorija, straničenje na zahtev i *TCP-IP* protokoli. Neke od varijanti ovog operativnog sistema su *Darwin*, *Dragonfly BSD, FreeBSD, NetBSD i OpenBSD* sistemi.

Neke kompanije su razvijale i svoje verzije *UNIX* operativnog sistema sa višestrukim radnim stranicama i serverima kao što su *Digital*-ov *Tru64, IBM AIX, Sun*-ov *Solaris*….

Dobre osobine *UNIX* sistema pokazale su se u sistemskim pozivima (nekoliko stotina), preglednom dizajnu, sistemu datoteka.

Sve ove osobine omogućile su lakše upravljanje podacima i uređajima svodeći ih na na skup prostih sistemskih poziva: *open(), read(),write(),ioctl(), close()*. Uz sve to, pisan je u jeziku *C*, koji se smatra jednim od robusnijih programskih jezika i koristi se u operativnim sistemima, a i šire (varijacije već postojećeg recimo *C++*). Omogućuje prenosivost kroz upotrebu *C* programskog jezika što je jedna od značajnih prednosti kada se piše *OS*.

*UNIX* veoma brzo kreira procese i ima jedinstven sistemski poziv *fork()*. Konačno, *UNIX* obezbeđuje prostu i robusnu komunikaciju između procesa. Danas je *UNIX* napredan operativni sistem koji omogućava široki spektar funkcija. Primena *UNIX* se svodi od uređaja koji koriste na stotine procesora do malih ugrađenih (embedded) uređaja.

*Linux* je nastao na Univerzitetu u Helsinkiju 1991. godine, i njegov nastanak se pripisuje Linusu Torvaldsu (Linus Torvalds), koji je razvio ovaj sistem kao sredstvo za učenje koji koriste *Intel*-ove mikroprocesore i386 ili 80386.

Razlog zbog kojeg je ovaj operativni sistem nastao su problemi koje je Linus Torvalds imao sa licencama tada već dobro utemeljenog operativnog sistema *UNIX*.

Prvobitno je razvijen kao emulator (imitacija okruženja operativnog sistema) koji se povezivao sa *UNIX* sistemima na fakultetu, a vremenom se razvijao i unapređivao.

Prva zvanična verzija ovog operativnog sistema puštena je na internet 1991, a od tada *Linux* postaje kolaborativni projekat.

*Linux* radi na sledećim procesorima:

MD x86-64, ARM, Compaq Alpha, CRIS, DEC VAX, H8/300, Hitachi Su-perH,   
 HP PA-RISC, IBM S/390, Intel IA-64, MIPS, Motorola 68000, PowerPC, SPARC, UltraSPARC i v850.

Kako se razvijao *Linux* tako je na scenu stupilo mnoštvo kompanija koje su imale svoju verziju *Linuxa*, a i generalno su se specijalizovale za ovaj operativni sistem.

Neke kompanije su *Red Hat*, *IBM, Novell*… Oni nude svoja rešenja za desktop sisteme, servere, ugrađene (embedded) sisteme.

*Linux* je uzeo veliki deo koda od *UNIX* operativnih sistema, zato bi se moglo reći da je klon. Tokom razvijanja operativnog sistema korišćene su ideje na kojima se zasniva *UNIX* npr. portabilnost i jednostavnost, no, kada se sve uzme u obzir, *Linux* ne izgleda kao potpuna kopija zato što je krenuo drugom putanjom od *UNIX* operativnog sistema na kojem je zasnovan.

Jedna od vodećih ideja *Linux-*a jeste da je sve *open source* tj. izvorni kôd je dostupan svima da rade sa njim šta hoće, što je obuhvaćeno licencom otvorenog koda *GPL* (General Public License version 2.0). Jedina obaveza koju bi imao neko ko je preuzeo ovaj kôd jeste da daje ista prava koja sam koristi, uključujući i raspolaganje izvornim kodom.

Osnovni delovi operativnog sistema *Linux* su *jezgro* (kernel), *C biblioteka* (library) , *prevodilac* (code interpreter), kao i niz alata, osnovne sistemske funkcije (kao što su *proces prijavljivanja* i *prevodilac komandi*).

*Linux* raspolaže i mnoštvom desktop okruženja i upravljačima prozora (window manger) koji su slični desktop okruženjima, samo što se fokusiraju na minimalizmu i slobodi modifikovanja. Takođe, postoji i mnoštvo aplikacija za *Linux*, komercijalnih i slobodnih.

# 2. *Linux* distribucije

Jedna od najvećih prepreka za nekog ko počinje bilo šta sa *Linux* operativnim sistemima je koju distribuciju izabrati.

Pregledom sledeće stranice:

[***https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux\_Distribution\_Timeline.svg***](https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg)  
  
može se videti koliko je izbor distribucije velika prepreka prilikom odabira. Postoje stotine varijacija na već postojeći *Linux* operativni sistem.

Kada bi se distribucije razvrstale na jednostavniju podelu, kao početna grana možda bi se uzela *Debian* jer je to, verovatno, i najpopularnija grana *Linux*-a, ili *Red Hat* grana koja je takođe popularna.

Jedna od popularnijih firmi u *Debian* grani je *Ubuntu* na koju se naknadne popularnije verzije nadovezuju. Moglo bi se reći da sve distribucije *Ubuntu* grane nude samo drugačiji izgled.. *Debian* sistemi koriste *APT* upravljač paketima za instalaciju programa i softvera.

Postoji i *SUSE* kao jedna od značajnijih grana. *SUSE* je sam po sebi grana *Slackware Linux*, koji je, uzgred, jedna od najstarijih inicijalnih grana *Linux* operativnog sistema.

*SUSE* je nastao samo godinu dana nakon *Slackware* i veću popularnost je stekao negde oko 2010. g. izlaskom *OpenSuse* operativnog sistema. Upravljač paketima koji ova distribucija koristi je *YAST. SUSE* nudi i plaćenu verziju svog operativnog sistema, odatle imamo i granu *openSUSE* koja je dostupna svima.

Sledeća u grupi većih ili značajnijih operativnih sistema je grupa *Red Hat* Linux. Popularne distribucije koje spadaju u ovu grupu su *Fedora, CentOS*, kao i sam *Red Hat* operativni sistem koji je razvijan za komercijalna rešenja.

Upravljač paketima koji se koristi je *RPM* (red hat package manager), *yum* (koji se nalazi na *CentOS* i *Fedora*).

Uglavnom sve verzije od *Red Hat* su visokokvalitetne verzije. Jedne od većih pokretača *web* servera na internetu su *Red Hat* i *CentOS*, gde je *CentOS* besplatna verzija pristupačnija, pa je klijenti često koriste za svoj server u nekim kompanijama koje se bave ovim delatnostima.

Postoji i *DNF package manager* kod novijih verzija kao što su *AlmaLinux* koji bi trebalo da zameni *CentOS* u nekoj bližoj budućnosti, ali to još uvek nije izvesno.

Takođe manje pomenuta, ali istaknuta distribucija *Linux-*a je *Arch Linux*. Prednost ove distribucije u odnosu na sve ostale je da je specifično pravljen tako da korisnik ima potpunu kontrolu od samog početka počevši od instalacije.

*Arch Linux* operativni sistemi imaju pristup velikoj količini repozitorijuma koje su postavljene od samog osnivača pa do običnog korisnika.

*Open Source AUR* (Arch user repositories) su u suštini pokretane od grupe korisnika koji postavljaju ove aplikacije na repozitorijum, a zvanični upravljač paketima je *pacman*.

Tipovi operativnih sistema koji bi trebalo da se navedu su sledeći:

*Rolling release*: Ovo su *bleeding edge* ili najaktuelniji operativni sistemi sa svim aplikacijama na najnovijoj verziji. Za serverske potrebe uglavnom se izbegavaju *rolling release* verzije operativnih sistema iz prostog razloga što ove verzije mogu da budu nestabilne, a za pravilan rad servera podrazumeva se operativni sistem koji nema takve probleme.

*Stable Release*: Ovo su verzije na kojima su aplikacije testirane i puštene u rad u ekosistemu operativnog sistema. Stabilne verzije operativnih sistema zastupljene su kod korisnika, a i na serverima zato što oni imaju manju količinu problema koje dolaze sa nestabilnim softverom.

Najvažnije je izdvojiti da upotreba distribucije zavisi od korisnika i sadržaja koji distribucija nudi, što odgovara korisniku u datom trenutku.

# 3. Struktura *Linux* sistema i dozvole

*Linux*, kao i svi trenutno prisutni operativni sistemi, ima svoju strukturu fajlova i način kako se vrše interakcije nad njima. Kako bi se bolje razumelo, bolje je gledati njegovu strukturu kroz terminal da bismo dobili neki stepen vizualizacije o načinu kako se sve organizuje.

*Slika 1: Struktura fajlsistema*



*Linux* distribucije uglavnom pokazuju sličnu stukturu direktorijuma, ali ponekad postoje varijacije koje su uglavnom sitne i specifične za operativni sistem.

## 3.1 Direktorijumi

*/bin*

Ovo je direktorijum koji sadrži *binary* fajlove, tj. neke aplikacije i programe koji mogu da se pokrenu na *OS.* Ako prelistamo ovaj direktorijum možemo naći komande koje često koristimo u terminalu npr. *ls* komandu za listanje fajlova, kao i druge osnovne alatke za pravljenje i brisanje fajlova i direktorijuma.

*/boot*

*Boot* direktorijum sadrži sve informacije za pokretanje operativnog sistema i skladištenje.

Ovaj direktorijum može biti prilično komplikovan za popravku ukoliko dođe do njegove korupcije, tako da se, uglavnom, bilo kakva konfiguracija izbegava, sem ako korisnik zna apsolutno šta želi da uradi.

*/dev*

*Dev* direktorijum sadrzi fajlove uređaja. Mnogi od ovih fajlova se generišu prilikom pokretanja operativnog sistema ili tokom rada računara, recimo, ako ubacimo kameru ili USB, novi uređaj će se pojaviti u ovom direktorijumu.

*/etc*

Sadrži konfiguracione fajlove i početne skripte. Ovaj direktorijum dobija ime od najranijih *UNIX* konvencija koji je stajao za ,,et cetera”, zato što je ovo bilo mesto za stavljanje fajlova za koje sistemski administrator nije bio siguran gde da ih stavi.

*/home*

Ovaj direktorijum sadrži početni direktorijum svih korisnika, osim administratora, tj. sadrži podatke svakog specifičnog korisnika.

*/lib*

Fajlovi biblioteke uskladišteni su u ovaj direktorijum, koji može da sadrži isečke koda aplikacije koji iscrtavaju prozor na desktopu ili šajlu fajlove u hard disk.

*/media*

Eksterni mediji, kao što su USB, priključeni su na ovaj direktorijum, kao i drugi uređaji USB hard diskovi , SD kartice, eksterni SSD … dok je računar uključen.

*/mnt*

Ovaj direktorijum je mesto gde bi ručno postavili uređaje za skladištenje ili particije. Danas se retko koristi.

*/opt*

Uglavnom ima više objašenja za ovaj direktorijum, ali po pravilu, ovaj direktorijum sadrži opcione fajlove koji nisu neophodni za pokretanje operativnog sistema.

*/proc*

Ovaj direktorijum se često naziva virtuelni direktorijum i sadrži mnoštvo informacija počevši od specifikacije sistema i procesa koji se pokreću u datom trenutku. Neki pod direktorijumi ovog direktorijuma su */proc/meminfo*, */proc/mounts* i */proc/partitions*. Bitno je izdvojiti ako se bilo koji od ovih poddirektorijuma izlista sa *cat,* *more* ili *less* komandom, dobili bismo neke informacije na terminalu koje bi nam bile korisne za informisanje o aktivnim uređajima na operativnom sistemu.

*/root*

Ovaj direktorijum je sličan korisničkom */home* direktorijumu, jedina razlika jeste da je ovaj direktorijum rezervisan za *root* korisnika i drugi korisnici ne mogu da mu pristupe bez upotrebe *sudo* komande koja bi im dala povećane privilegije pristupa.

*/run*

Sistemski procesi koriste ovaj direktorijum da čuvaju privremene podatke.

*/sbin*

isto što i */bin* s tim što sadrži aplikacije koje samo *superuser* i *root* koriste ili su im potrebni u radu.

*/usr*

Sadrži sekundarne programe, biblioteke i dokumentacije o programima povezanim sa korisnikom.

*/var*

Direktorijum *log* fajlova, sadrži informacione fajlove koji pomažu prilikom otkrivanja nekih većih kvarova, barem na serverskim sistemima. Direktorijum *var* stoji za promenljivi tip podataka (varijable), odatle i skraćenica. Takođe, ovaj direktorijum sadrži i druge fajlove koji su promenljivog tipa.

/sys

Virtualni direktorijum kao i */proc* koji sadrži informacije o uređajima koji su povezani na operativni sistem.

## 3.2 Dozvole u *Linux* sistemima

U *Linux* sistemima postoje tri vrste dozvola:

* Dozvola za čitanje (r), ili 4 u brojčanom zapisu; korisnik može da čita ili da proveri fajl.
* Dozvola za pisanje (w), ili 2 u brojčanom zapisu; korisnik može da modifikuje fajl.
* Dozvola za izvršavanje (x), ili 1 u brojčanom zapisu; korisnik može da izvršava fajl.

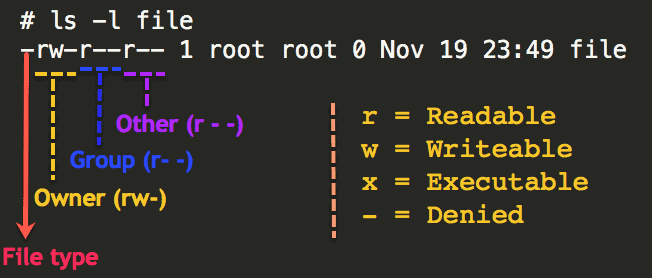
### 3.2.1 Grupe dozvola

*Owner* – dozvole vlasnika se odnose samo na vlasnika fajla ili vlasnika sadržaja, neće uticati na akcije korisnika.

*Group* – grupne dozvole se odnose na grupu koja je dodeljena fajlu ili direktorijumu, na nju neće uticati akcije drugih korisnika ( koji nisu članovi operativnog sistema).

*All-users* – dozvole svih korisnika odnose se na sve korisnike koji pokušavaju da pristupe fajlu. Ova dozvola najviše se kontroliše tokom rada radi sigurnosti fajlova.

*Slika 2 : Dozvole fajla*



*izvor: thegeekdiary.com/understanding-basic-file-permissions-and-ownership-in-linux*

### 3.2.2 Promena dozvola i komande

Tehnike promene dozvola zavise od preferencije korisnika, ali generalno, barem kod sistemskih administratora, koristi se numerički metod gde broj 4, 2 ili 1 predstavlja neku vrstu dozvole koja je ranije navedena.

**$ chmod 777 fajl.txt**

Ovo će dati sve dozvole jednom fajlu na sistemu ili, u simboličkom smislu, odgovara RWX RWX RWX , *read write and execute* za vlasnika, grupu i sve ostale korisnike.

### 3.2.3 Funkcija *setuid*

Funkcija *setuid* (set user identity) dozvoljava korisniku da pokreće fajlove samo sa dozvolama vlasnika (owner) grupe. Uglavnom se koristi kada je neophodno da se korisnicima obezbede veće privilegije, bez upotrebe *sudo* komande tj. da se korisnici dodaju u *wheel* grupu.

Podešavanje ovog načina izvršavanja fajla vrši se već ugrađenom funkcijom:

**$ chmod u+s fajl.txt**

ili

**$ chmod 4777 fajl.txt**

*Slika 3 : Dozvole fajla nakon setuid*



Dozvole nakon bilo koje od dve prethodne komande, biće kao sa slike (slika 3). Pokazatelj da je podešen ovaj način izvršavanja je *„s”* karakter u polju za vlasnika (owner), a i sam sistem ga osvetli drugačijom bojom da se skrene pažnja na fajl.

### 3.2.4 Funkcija *setgid*

*Setgid* (set group identity) funkcija je slična prethodnoj funkciji *setuid*, s tim da umesto da menja vlasničke (owner) dozvole, ova funkcija menja grupne (group) dozvole.

Podešavanje ove funkcionalnosti vrši se na sličan način s tim da se menja argument (flag) na 2 umesto 4 ili se simbolički dodeljuje grupi (g) *set argument* (s).

**$ chmod g+s fajl.txt**

ili

**$ chmod 2777 fajl.txt**

*Slika 4 : Dozvole fajla nakon setgid*



Dozvole nakon bilo koje od dve prethodne komande biće kao sa slike (slika 4). Može da se primeti da je podešeno po *„s”* karakteru u dozvolama grupe (group). Takođe, kao i sa vlasničkim (owner) dozvolama, fajl je osvetljen drugačijom bojom, tako da skrene pažnju da je funkcija aktivna.

### 3.2.5 Funkcija promene grupe i vlasnika

Komanda ili funkcija promene vlasnika, grupe, omogućava manipulaciju fajla u zavisnosti od potreba sistemskog admnistratora ili *sudo* korisnika operativnog sistema. Najčešća upotreba ove komande na višekorisničkom serveru, jeste da dodelimo fajl korisniku nakon čega korisnik može da izvršava željene promene.

Sintaksa komande:

**$chown user:group fajl**

*Slika 5 : Promena vlasnika i grupe*



*Slika 6: Rezultat komande chown*

### 3.2.6 Lepljivi bit

Dodatna komanda za manipulaciju specijalnih privilegija pored *setuid* i *setgid* bila bi upotreba lepljivog bita (sticky bit). Sistemski administrator koristio bi ovu komandu kako bi obezbedio specijalnu privilegiju koja omogućava samo vlasniku fajla da briše i prepravlja fajl, dok drugi korisnici mogu samo da čitaju sadržaj. Vrlo često možemo videti upotrebu lepljivog bita (sticky bit) u */tmp* direktorijumu gde na višekorisničkom serveru svaki korisnik kreira svoj proces koji nešto radi, a procesi generišu fajlove (pod imenom korisnika koji je pokrenuo aplikaciju) u /*tmp* direktorijumu koji su im neophodni za pravilan rad. Lepljivi bit će onemogućiti korisnicima da manipulišu privremenim fajlovima koje je kreirao drugi korisnik.

**$ chmod +t fajl.txt**

ili

**$chmod 1777 fajl.txt**

*Slika 7: Dozvole /tmp direktorijuma sa lepljivim bitom*

*Slika 8: Dozvole fajl.txt sa lepljivim bitom*

Sa slika (slika 7 i slika 8) može da se primeti upotreba lepljivog bita po karakteru *„t”* koji nam prikazuje da je fajl podešen sa specijalnom dozvolom.

## 3.3 Objašnjenje *sudo* komande

Česta komanda koja se vidi prilikom rada na bilo kom *Linux* serveru, ako korisnik nije inicijalno pristupio serveru kao *root* korisnik, je *sudo*.

*Sudo* (super users do) komanda se koristi kada želimo da izvršimo komandu koja zahteva povećane privilegije koje dolaze sa *root* korisnikom.

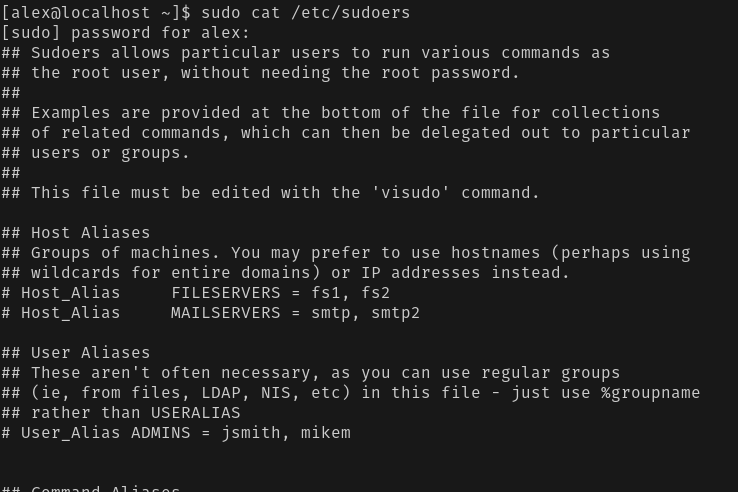
Ako se pokreće komanda za koju nemamo privilegije kao korisnici dobijamo sledeće:

*Slika 9 :Sudoers fajl*



Izvršavanjem ove iste komande sa *sudo* dobijamo:

*Slika 10 :Sudoers fajl sadržaj*



Može da se vidi sa slike (slika 10) da je korišćenjem povećanih privilegija dozvoljen pristup *sudoers* fajlu kojem ima pristup *root* korisnik.

Prolaskom kroz *sudoers* fajl, pri kraju ćemo naići na *%wheel* (iznad ovog odeljka piše objašnjenje čemu služi).

*Slika 11 :Sudoers fajl wheel grupa*



Sledeće pitanje koje se postavlja jeste da li je korisnik zapravo u grupi *wheel*.   
Ovo lako može da se proveri pregledom sledećeg fajla */etc/group:*

*Slika 12: Sudoers fajl - pripadnost korisnika*



Može se primetiti (slika 12) da je grupa *wheel* prvi segment. Drugi segment je *x* što znači da su upotrebljene skrivene lozinke, *10* je *ID* grupe i jedini član *wheel* grupe je *alex*. Na novijim sistemima ovo se često zove i *admin* pa postoje i varijacije u *sudoers* fajlu.

Mogućnost pokretanja superkorisnika pomoću komande *sudo*, nije pravo svakog korisnika na sistemu, ono zavisi od kompanije i prava korisnika koje taj učesnik ima u timu administratora. (Adam K. Dean, Administriranje Linux Sistema, 47)

# 4. Instalacija *CentOS* operativnog sistema

Pre same instalacije nephodno je izabrati odgovarajuću arhitekturu za operativni system. Ako se ne izabere pravilno, dobija se upozorenje, što nije strašno samo po sebi, ali može da uspori instalaciju. (Treba ponovo da se preuzima *image* sa pravilnom arhitekturom.)

*Slika 13: Greška arhitekture sistema*



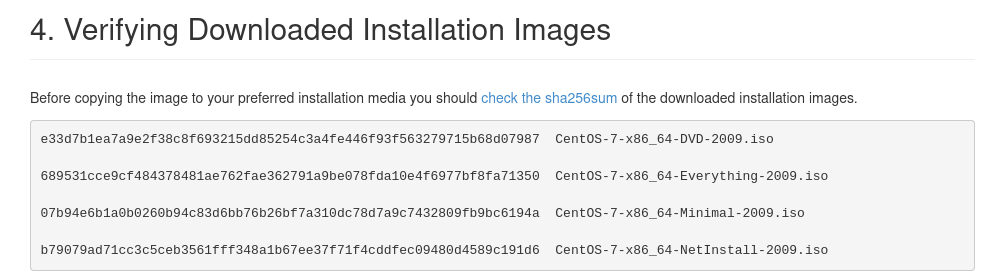
Takođe nije loše proveriti da li je *ISO* (fajl koji je preuzet sa interneta koji radi isto kao i CD, prilikom *boot up* sekvence) operativnog sistema ispravan proverom s*ha256sum.*

*Slika 14: Shasum CentOS iso fajla*



Kako bi se potvrdilo da li ovo odgovara zvaničnom s*ha256sum* potrebno je posetiti sledeću stranicu <https://wiki.centos.org/action/show/Manuals/ReleaseNotes/>. Ovde moramo da pronađemo verziju operativnog sistema, u ovom slučaju je to *CentOS*. Slična dokumentacija koja obuhvata *CentOS* postoji i za *Debian* i druge distribucije *Linux.*

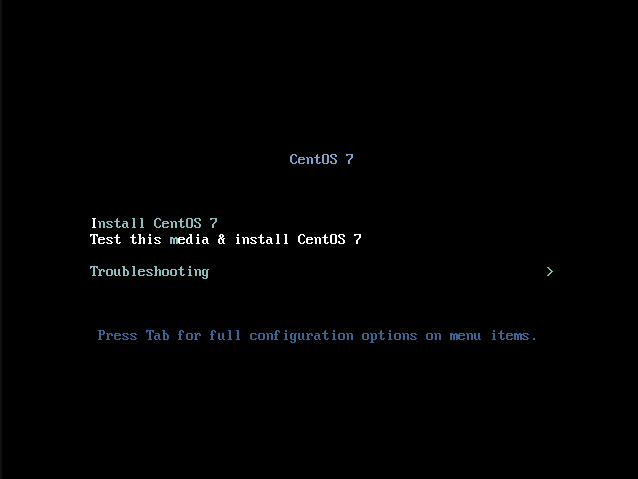
*Slika 15: Prikaz shasum256 operativnog sistema na dokumentaciji*



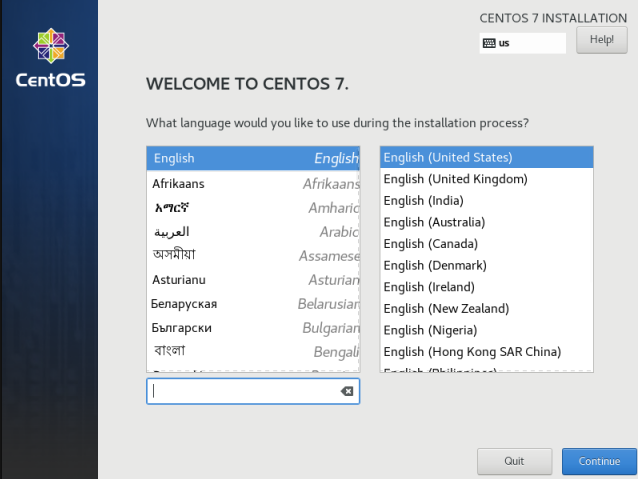
## 4.1 Opšta konfiguracija

Pre svega treba obezbediti *boot order* (princip pokretanja medijuma na kojem se nalazi instalacija operativnog sistema) kako bismo rekli sistemu da pokrene *ISO* fajl koji se nalazi na *USB* ili nekom drugom medijumu. Ovo podešavanje može da se odradi kroz *BIOS* koji nam nudi odabir medijuma koji će biti upotrebljen prilikom pokretanja računara.

Nakon odabranog medijuma, instalacija počinje nakon što se izabere opcija *install CentOS 7*. Instalacija je jednostavna, zato što se modernije distribucije trude da ovaj proces učine što lakšim za konfiguraciju.

*Slika 16: Početni prozor instalacije*  


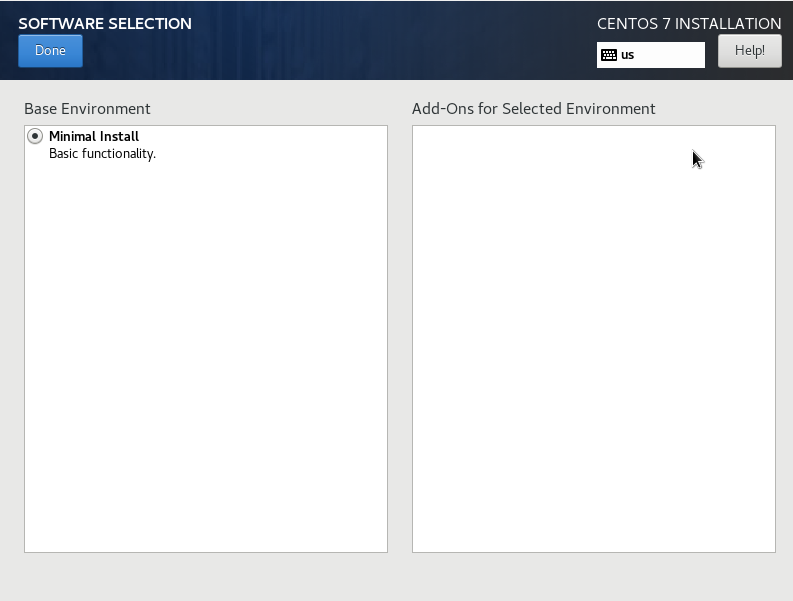
*Slika 17: Odabir jezika instalacije*



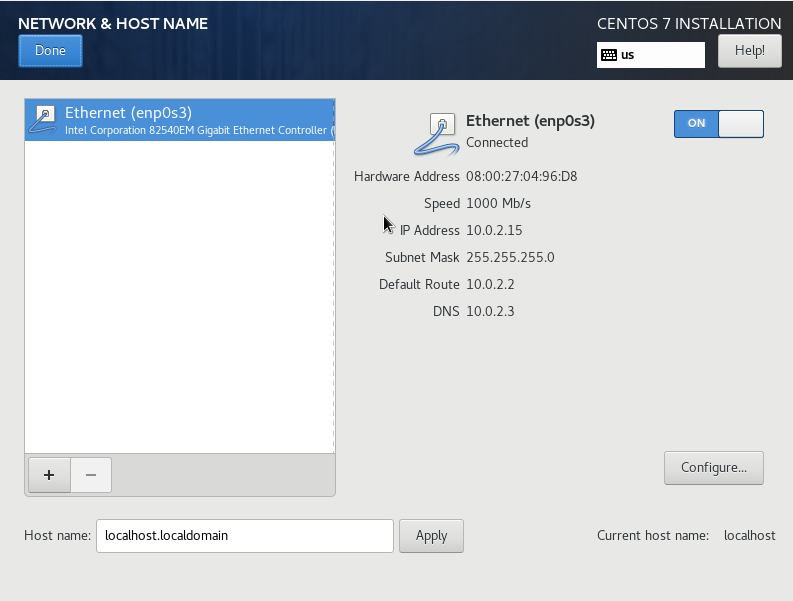
*Slika 18: Glavni instalacioni prozor*



Pošto je ideja da ovo bude server, odabirom opcije *Software Selection* pokušavamo da napravimo server što je moguće manjim, tj. da se sve radi kroz terminal. Ovo je bitno da se uradi jer aplikacije na serveru same po sebi zauzimaju dosta memorije i dodavanje *GUI* (Graphical user interface) zauzima još prostora, pa se se uglavnom serveri sa *GUI* izbegavaju.

*Slika 19: Odabir paketa instalacije*  


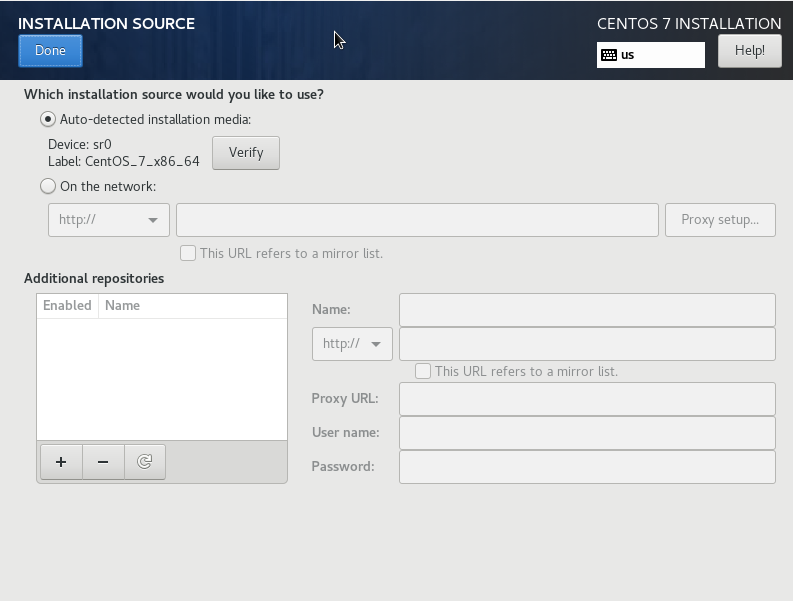
*Slika 20: Uključivanje network uređaja*



Sledeće što bi trebalo da se uradi jeste aktiviranje mrežnog uređaja odabirom opcije na dugmetu. Nakon što se klikne (slika 20) vidi se početno podešavanje za mrežu.

Pošto je ovo virtualno okruženje, *VirtualBox* kreira *NAT* mrežu sa podrazumevanim podešavanjem, što znači da se *VM* nalazi na istoj mreži kao i *host* računar (Virtualna mašina je sama na mreži, ali sa putanjom ka spoljašnjem svetu preko *host* mašine).

*Slika 21: Odabir instalaciong izvora*

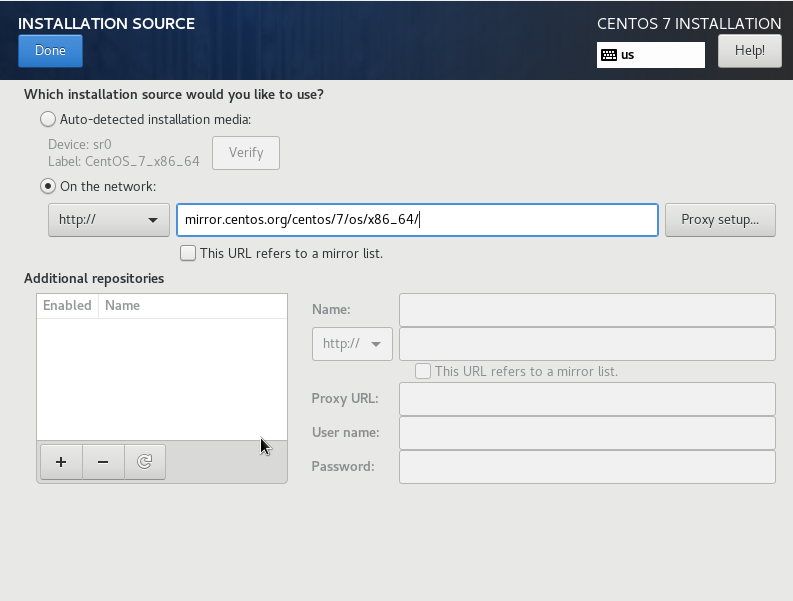


Takođe, veoma bitno tokom ove instalacije je podešavanje instalacionog izvora. Unutar ovog ekrana automatski je izabran medijum, *ISO* disk (*sr0* je oznaka za *drive* diska).

Promenićemo ovo dugme na *On the Network* (zbog prethodnog podešavanja za konekcije ka internetu).

Nakon podešavanja dobijamo:

*Slika 22: Unos instalacionog izvora*



Može se videti (slika 22) da unosimo samo *mirror* (alternativni server) koji će izvor (source) koristiti prilikom instalacije.

Ovom promenom, kada se vratimo na početni prozor, sistem će izbaciti upozorenja da je izvor promenjen i da se mora podesiti ponovo (slika 23).

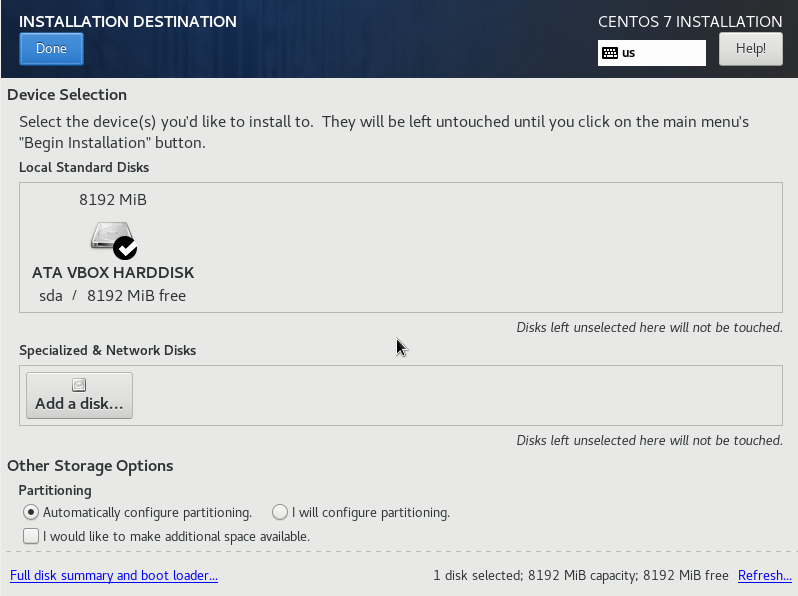
*Slika 23: Potvrda selekcije*



Postavka ostaje ista kao i prethodna, na *minimal installation,* i treba da se potvrdi u interfejsu.

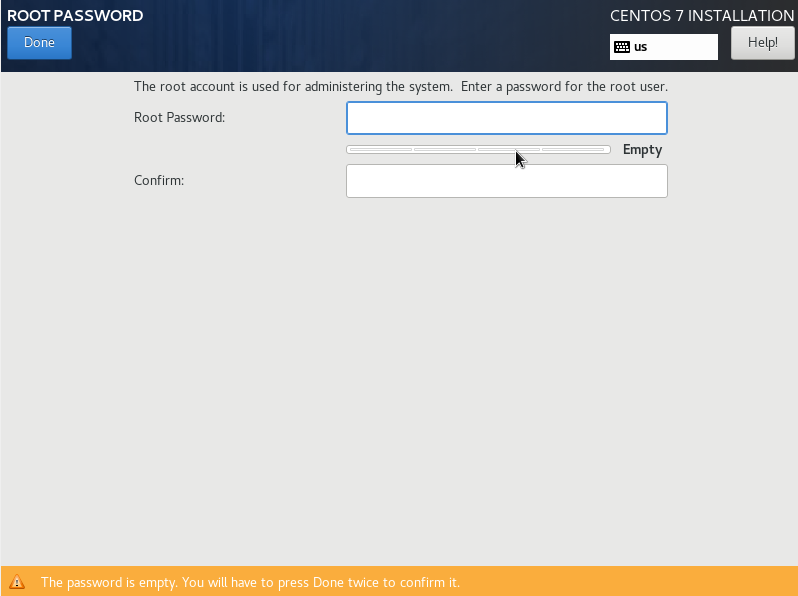
Poslednja stavka je odredište instalacije (installation destination), u ovom slučaju dozvolićemo instalaciji da kreira particije kako želi.

*Slika 24: Odabir diska instalacije*

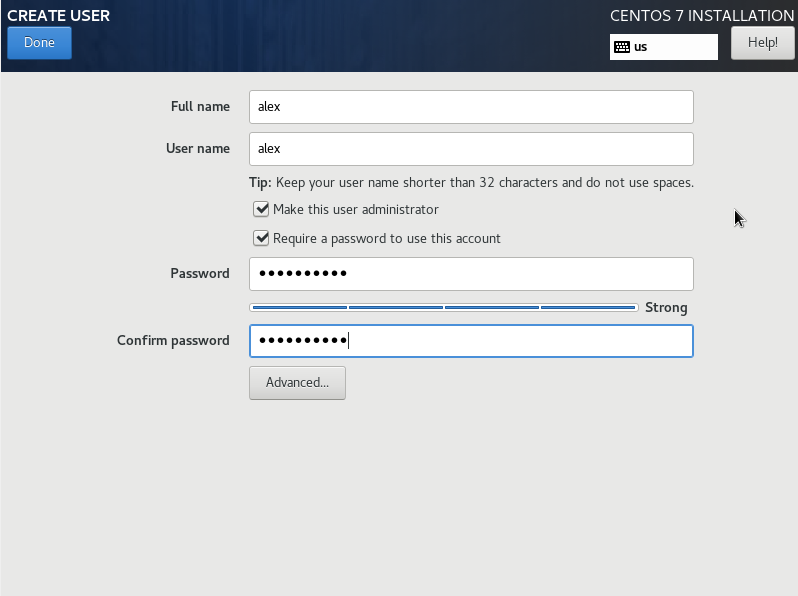


Tokom instalacije pojavljuje se prozor koji nudi podešavanje *root* korisnika. Kada počnemo instalaciju (uglavnom na *CentOS 8* i *9)* ovo se nalazi na inicijalnom prozoru:

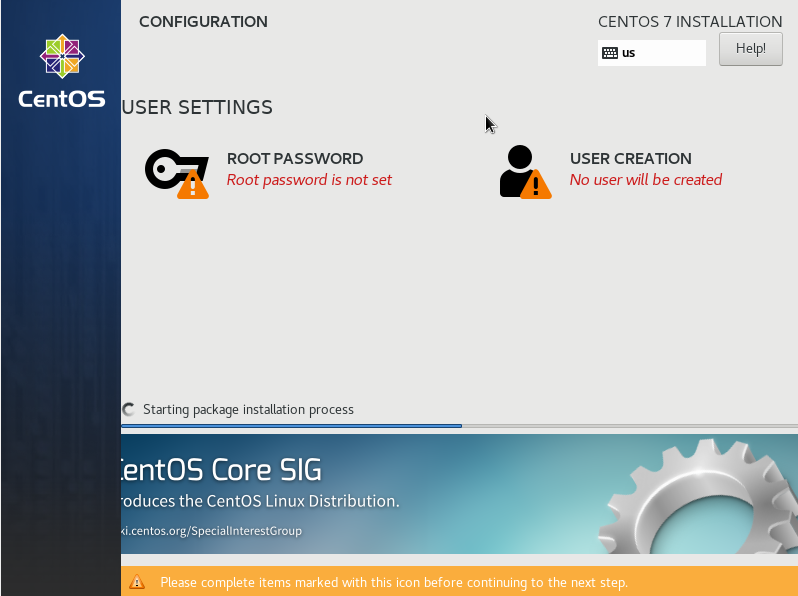
*Slika 25: Kreacija root korisnika*



*Slika 26: Kreacija korisnika*

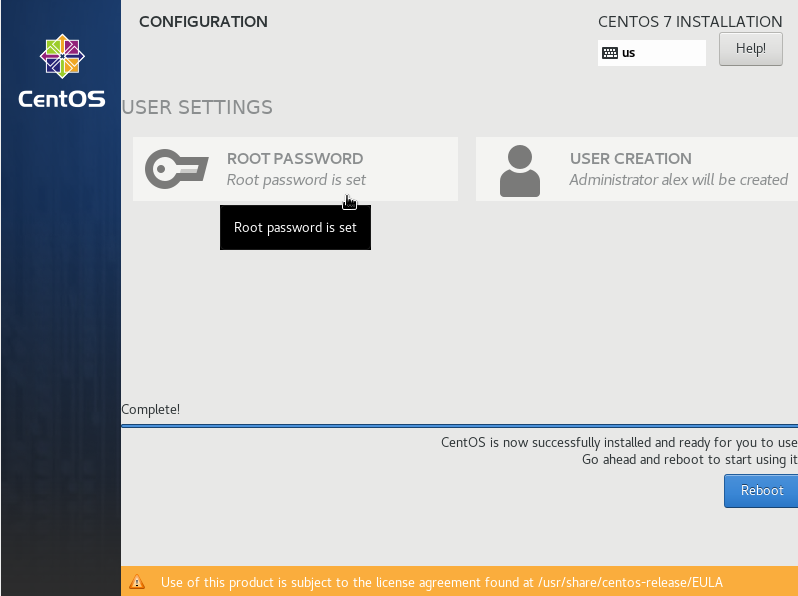


*Slika 27: Instalacija pre kreiranje korisnika*



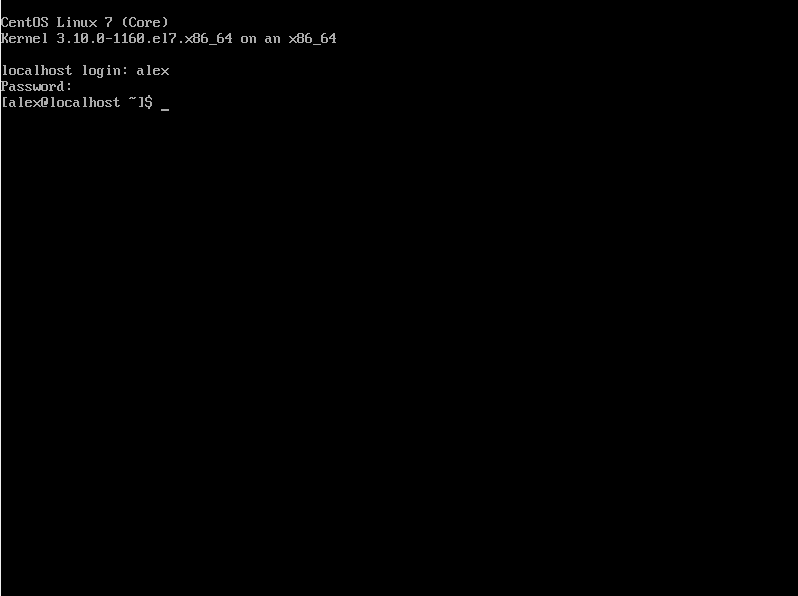
Izabrana je opcija da se korisnik načini administratorom. Ovo će ga dodeliti *sudo* korisnicima tj. korisnicima sa povećanim privilegijama na sistemu.

*Slika 28: Instalacije nakon kreiranja korisnika*



Prvi *login* na sistem izgleda ovako:

*Slika 29: Server login*



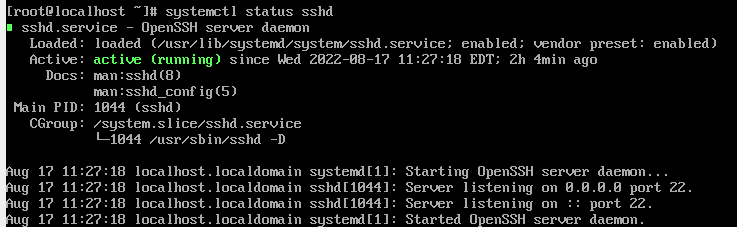
# 5. Podešavanje udaljene konekcije SSH klijentom

Generalno, kada se radi nešto na serverima, podešava se *SSH* klijent koji omogućava udaljenu konekciju do servera ili nekog računara u zavisnosti od njegove funkcije.

Samim tim što se primeri ovde svode na korištenje virtualne mašine, pogodno je podesiti ovaj tip konekcije zato što nam olakšava obavljanje zadataka.

## 5.1 Opšta konfiguracija

Prvo što ćemo da uradimo jeste da testiramo postojanje *SSH* klijenta, a i da li je *port* otvoren: *Slika 30: Status sshd servisa*



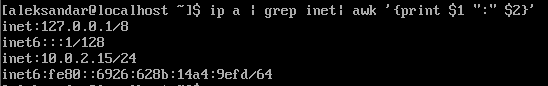
Sa slike (slika 30) servis je aktivan i sluša na *port*-u *22*, ali nešto nije u redu kada se konektujemo sa *host* mašine, konekcija se ne ostvaruje. Pretpostavka je da će eventualno vreme čekanja isteći i dobićemo ili *timeout error* ili *refused to connect*.

*Slika 31: Login na udaljeni server*



Da bismo dobili ovu početnu *IP* adresu možemo koristiti komandu *IP a -* ona će nam dati neophodne informacije o adapterima koji su priključeni na server.

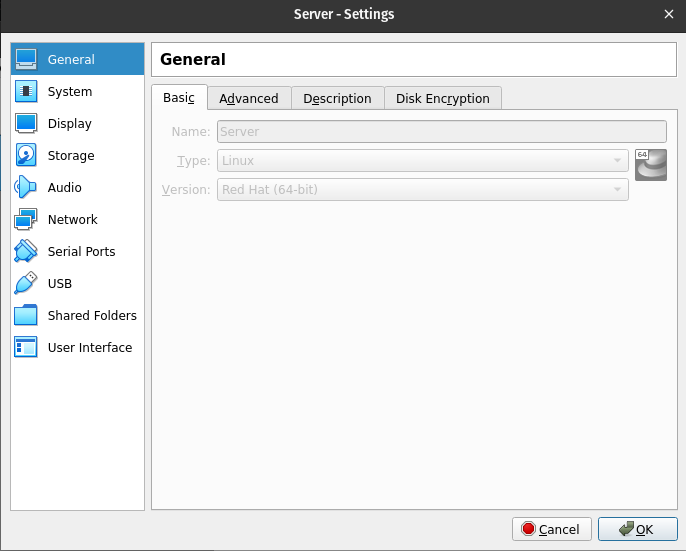
*Slika 32: Upotreba IP komande*



U spisku imamo *loop back* IP 127.0.0.1 i *IP* servera koji je 10.0.2.15. Ako bismo dobili *refused to connect* implicira se na to da je server blokirao *IP*.

Kako bismo konfigurisali udaljenu (remote) konekciju (sa *hosta*) moraćemo podesiti *port forwarding* u virtualnoj mašini (pošto simuliramo postojanje servera preko ove mašine).

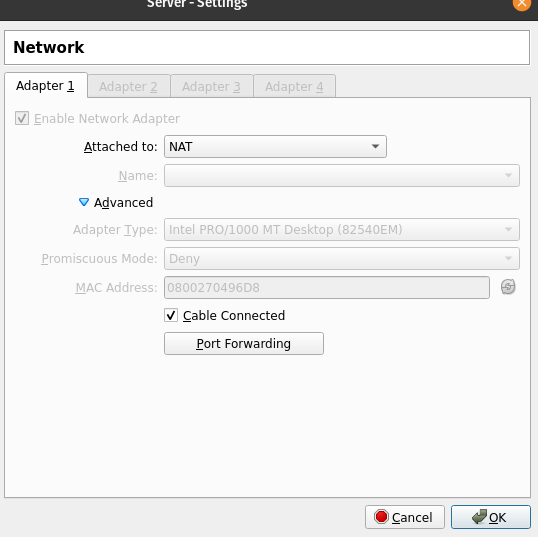
*Slika 33: Prozor podešavanja VM*



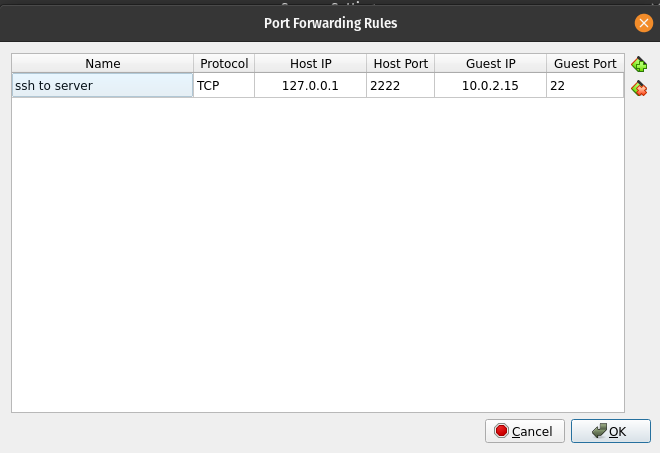
Da bismo konfigurisali prethodno navedeno, idemo na *network* u *virtual box* aplikaciju i pritiskom na *advanced* dugme dobijamo opciju odabira podešavanja *port forwarding.*

(*Port forwarding* je metod ručnog podešavanja, koji govori kako saobraćaj treba da prođe kroz *NAT* mrežu.) (Adam K. Dean, Administriranje Linux sistema, 36)

*Slika 34: Prozor podešavanja NAT konekcije VM*

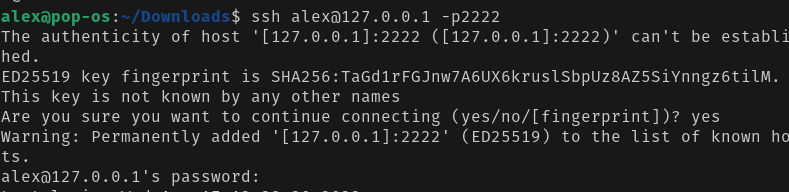


*Slika 35: Podešavanje pravila za port forwarding*



Podešavanje koje je ovde postavljeno pokazuje da kada želimo *SSH* konekciju upišemo sledeće: *ssh alex@127.0.0.1 -p2222*. Server će da prepozna ovaj tip konekcije ka njemu i prebaciće ga na *port 22*, koji koristi *SSH* klijent na virtualnoj mašini pri čemu je ovde ostvaren vid konekcije koji je uglavnom dostupan prilikom administriranja servera.

*Slika 36: Login na server koji je kreiran u virtualnom okruženju*

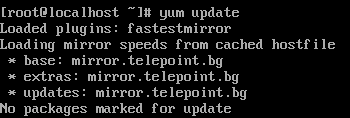


(*Guest IP* je *IP* servera koji možemo dobiti pozivom komande *IP a.*)

Sledeće što ćemo uraditi jeste *update* (metod unapređivanja već postojećeg softvera) koji će povući sve poslednje verzije softvera koji je pokrenut na serveru ukoliko za to postoji potreba. Kako bismo obavili ovaj zadatak koristićemo komandu:

**$ yum update**

***S****lika 37: Upotreba yum update komande*



Prilikom rada često je nephodno proveriti verziju jezgra (kernel), a to se može odraditi upotrebom sledeće komande:

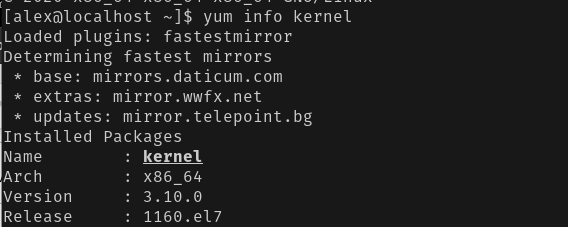
**$ uname -a**

*Slika 38: Upotreba komande uname*



Zatim pokretanjem sledeće komande:

*Slika 39: Prikaz poslednje verzije kernela u yum*



Može se videti da je *kernel* verzija ista kao i verzija pozvana *uname* komandom što implicira da je jezgro poslednje verzije.

*Slika 40: Verzija kernela*



# 6.Automatizacija pravljenja virtualnog okruženja

## 6.1 Uvod

Instalacija operativnog sistema na virtualnoj mašini može da bude prilično repetativna, jer, uglavnom, moramo praviti novo okruženje za određene probleme koje želimo da testiramo. Prema tome, ako svaki put ručno podešavamo mašinu, gubimo mnogo vremena za zadatke koji su možda izuzetno jednostavni. Zato je, na inicijativu programera i sistemskih administratora, kompanija *HashiCorp* razvila rešenje za automatizaciju kreiranja virtualnog okruženja.

Aplikacija *vagrant* je inovativno rešenje koje se služi pravilima otvorenog koda (open source) radi razmenjivanja već pripremljenih okruženja za testiranje. Ova radna okruženja mogu biti postavljena od samih članova organizacije pa do korisnika koji koriste ovu aplikaciju. Svako od ovih okruženja ili *box,* kako se zovu unutar ove zajednice, rade u vidu „recepta” koji, kada se unesu u konfiguracioni fajl, postavljaju okruženje po parametrima zadatim na ,,klaud“ (claud) serveru koji reguliše postojanje radnih okruženja.

Ovaj vid automatizacije izuzetno skraćuje vreme neophodno da se podesi okruženje za testiranje. Pokretanjem nekoliko jednostavnih komandi, podešava gotovo sve neophodno za ovaj tip virtualizacije.

*Vagrant* aplikacija radi po principu „*init”* fajlova koji se kreiraju prilikom pozivanja određene komande unutar lokalnog sistema. Ovi fajlovi sadrže početne informacije koje *vagrant* čita prilikom kreiranja okruženja i takođe omogućava konfigurisanje fajlova po želji korisnika. ( Dokumentacija *vagrant* aplikacije nudi šire objašenje šta može da se odradi unutar fajlova kako bi odgovaralo željama korisnika (HashiCorp, 2022))

Naravno, ovaj tip automatizacije nije novitet, pošto na realnim serverima postoje takođe fajlovi koje možemo nazvati fajlovi inicijalizacije (tehnički termin bi bio *kickstart* *fajl*). Ovi fajlovi rade sličan posao kao i *vagrant* za lokalnu mašinu gde podešavaju server u skladu sa napisanom skriptom unutar fajla. Ova podešavanja često možemo primetiti na *VPS* (Virtual private server) serverima koji su virtualne instance unutar jednog glavnog servera i njegovu konfiguraciju prenosi (nasleđuje) na potomke. Takođe, glavni server dodeljuje resurse svojim potomcima u zavisnosti od podešavanja. Na zasebnim (dedicated) serverima ovi fajlovi se distribuiraju u mrežu gde konfiguracioni fajlovi vrše podešavanje spram direktiva koje su zadate.

## 6.2 Konfiguracija virtualnog okruženja

1) **$ sudo apt install vagrant** (Ova aplikacija biće automatski instalirana i podešena jer postoji u bazi podataka). Na *Windows* mašinama bismo preuzeli aplikaciju i kroz instalacioni softver podesili podešavanje kako nam odgovara.

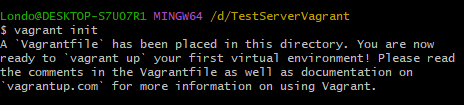
<https://www.vagrantup.com/downloads> (stranica koja nam nudi inicijalna upustva instalacije)

2) Kreiramo direktorijum na *Linux* mašini ili *Windows* mašini (Uglavnom je preporučljivo da ovaj direktorijum bude unutar zasebnog direktorijuma radi lakšeg upravljanja).

3) Inicijalizujemo direktorijum kao mesto za *vagrant* konfiguraciju koristeći sledeću komandu:

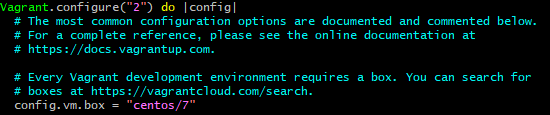
**$ vagrant init**

*Slika 41: Inicijalizacija vagrant fajla*



4) Sledeće što bi trebalo da se uradi jeste da se podesi „*init*” fajl tako da konfiguracija odgovara našim potrebama.

*Slika 42: Konfiguracija vagrant init fajla*

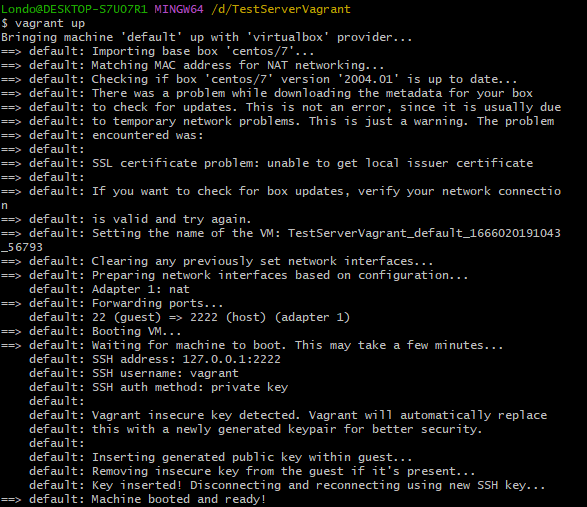


U konfiguracionom *vagrantu* fajlu podešavamo da koristimo *CentOS7* *box* za ovu virtualnu mašinu.

5) Pokrećemo virtualnu mašinu koristeći:

$ **vagrant up**

*Slika 43: Pokretanje vagrant servera*



6) Testiramo *login*:

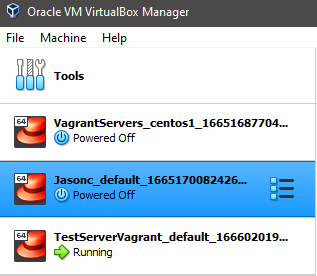
$ **vagrant ssh** (Ovo je kreirano već unapred, kada se izvrši komanda *vagrant ssh* kao sa slike 44)

*Slika 44: SSH konekcija ka novokreiranom serveru*



Postoji i $ **vagrant destroy** koji briše instancu potpuno iz *vm box* ukoliko za to postoji potreba, ali može isto to da se uradi iz *virtual box* aplikacije jer su sve instance tamo takođe predstavljene.

*Slika 45: Kreirane vagrant instance unutar virtual box aplikacije*



Bilo je nephodno prikazati oba procesa podešavanja, pošto je lakše razumeti *vagrant* automatizaciju kroz ručno podešavanje servera iz odeljka [Instalacija CentOS sistema](#_4._Instalacija_CentOS) .

# 7. Daljinsko administriranje pomoću SSH

Sistemski administrator obavlja poslove udaljeno, ali u isto vreme može da bude na licu mesta - kraj servera i da pravi neke promene.

Pretpostavka je da neće svaki put ići do *data* centra (mesta gde se nalaze serveri) ako se neke stvari mogu rešiti samo povezivanjem na udaljeni server.

Ovo je razlog zašto će sistemski administrator koristiti *SSH* (secure shell) konekcije koje omogućavaju administratoru *Linux* sistema da pristupi udaljenom serveru i napravi promene koje su neophodne.

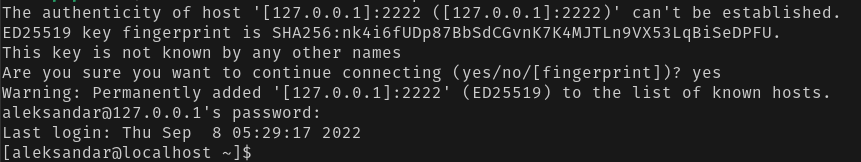
*Secure Shell* protocol jedan je od korišćenih primera, a s obzirom da je jedan od poznatijih, a i bezbednijih, predstavljen je u ovom radu. Deo je fabričkog podešavanja na svim *Linux* sistemima.

Komanda koja se koristi:

**$ ssh root@192.168.56.101 -p522 komanda ssh**

Prvi deo komande označava korisnika koji pristupa serveru, drugi deo komande nakon *@* simbola odnosi se na server *IP*. *IP* može da se zameni sa *hostname* (ime servera) i treći deo *-p* odnosi se na *port* koji će biti korišćen, u ovom slučaju to je *522*. Nakon ukucane komande dobićemo ključ, koji bi trebalo da potvrdimo i nakon toga unosimo lozinku tog korisnika.

*Slika 46: Potvrda prilikom udaljene SSH konekcije ka serveru*



Ovo je uobičajena metoda konekcije (Slika 46), ali konekcija ka serveru može da se uspostavi i na alternativni način - generisanjem ključa.

## 7.1 Generisanje i upotreba *ssh-keygen*

Većina ljudi koristi jednostavne lozinke koje se mogu lako pogoditi koristeći različite algoritme, klasična metoda je *brute force* (metoda sile) gde se pokušavaju sve moguće lozinke ili *dictionary attack* (metoda rečnika) koja koristi zadate reči da pronađe lozinku.

Jedna metoda koja je razvijena za sprečavanje ovih napada, a i za identifikaciju samo- autorizovanih korisnika je korišćenje kriptografskih ključeva koji mogu biti privatni i javni.

Ta metoda funkcioniše na sledeći način - kada je javni deo ključa na serveru, može samo da se izvrši *SSH* konekcija sa mašine koja poseduje privatni ključ.

Ovaj tip konekcije može donekle da olakša povezivanje, gde se izbegavaju neki dodatni koraci kao što je upisivanje lozinke, tj. lozinka se upisuje samo prvi put, dok svaki naredni, ta fraza (passphrase) ostaje zapamćena na mašini i koristi se iznova ukoliko želimo da se povežemo na isti server.

Kako bi to izgledalo između dva servera?

Prvo što bismo uradili je povezivanje na server (virtualna mašina).

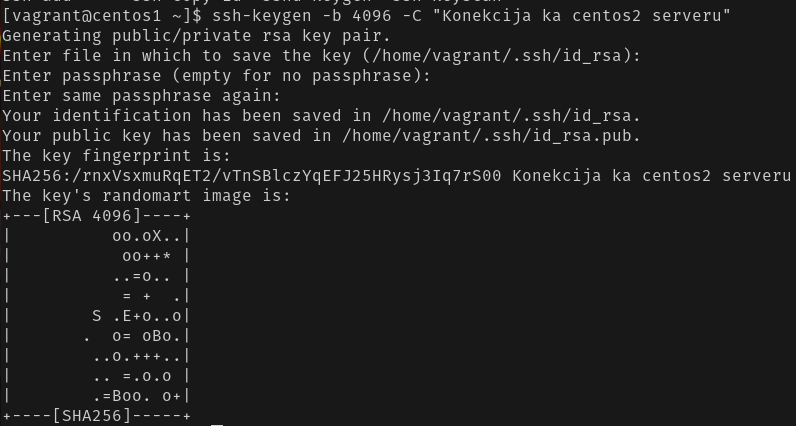
*Slika 47: Login na virtualnu mašinu*



Iz prethodnog odeljka ([Automatizacija pravljenja virtualnog okruzenja](#_6.Automatizacija_pravljenja_virtual)) može da se vidi šta je *vagrant*. Pošto je ovo generisano okruženje, daje nam olakšicu konekcije ka novoj mašini koja se zove *centos1*.

Drugo što bismo uradili na mašini jeste da generišemo par ključeva, koji će nam biti koristan prilikom konekcije ka drugom serveru.

*Slika 48: Generisanje ključa*

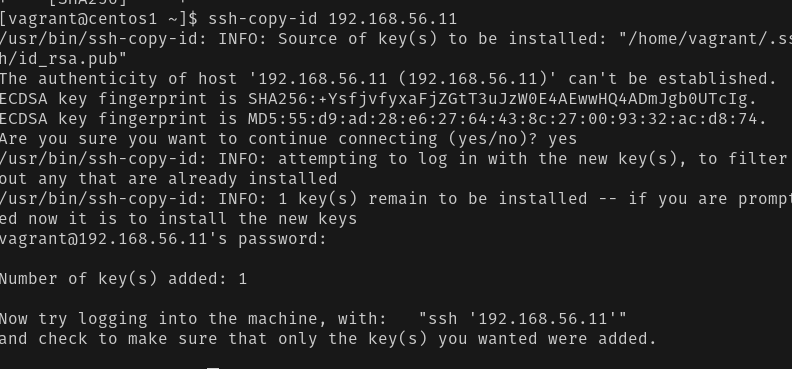


Ključ je generisan, čak i komanda pita ako želimo negde drugde da sačuvamo ključeve. Zatim unosimo frazu (passphrase) koju želimo i ključevi se čuvaju na serveru kao *id\_rsa i id\_rsa.pub*.

Naravno, ovo samo po sebi ništa ne znači, da bismo mogli ovo da koristitmo neophodno je da kopiramo javni ključ (public key) na odredišni server.

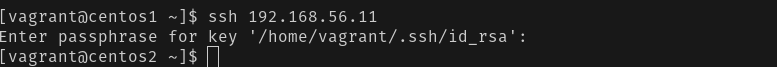
Ovo možemo uraditi ručno ili koristeći komandu *ssh-copy*kadestinacionom serveru koji prihvata ključ kada se unese lozinka koju će zatražiti prilikom razmene.

*Slika 49: Kopiranje ključa na udaljeni server*



Može da se primeti da nam komanda traži frazu (passphrase) koju smo zadali prilikom generisanja ovog ključa (Slika 50). Nakon unosa omogućena je konekcija.

*Slika 50: Login na udaljeni server preko ključa*



Naravno, ovaj tip ključa koji je korišćen nije jedini, postoje još i *ED25519, dsa, ecdsa, rsa.*

Pošto je *SSH* prilično opširan, ovde su izdvojene neke osnove. Korisne stranice su zvanična dokumentacija i *manual* (man) stranice iz terminala vezane za komandu *ssh*.

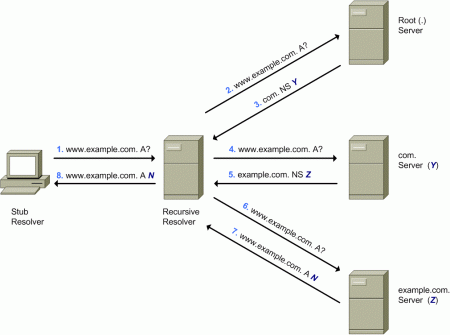
*https://www.openssh.com/manual.html*

# 8. *Networking/DNS*

Kada pretražujemo internet, uglavnom ne razmišljamo šta se dešava u pozadini. Ako bismo ukucali *google.com*, pomislili bismo da će pretraživač odmah pronaći ono što tražimo.

Naravno, ovo nije ceo proces koji se odvija u samo nekoliko sekundi, naša pretraga prolazi kroz niz provera kako bismo dobili rezultate o zadatom terminu.

*Slike 51: Predstava funkcionalnosti DNS*



*Izvor: https://www.internetsociety.org/wp-content/uploads/2019/02/dnsprivacy2-450x335.gif*

*DNS* pretvara *URL* u brojeve ili *IP* adresu koju računar razume, zato što se u mašinskom jeziku ne koriste imena.

Da bi se bolje razumeo *DNS* može se ukratko objasniti kako ovo funkcioniše u celosti.  
Na primer, tražimo neku stranicu *example.com* (Slika 51). Prvo što će se desiti jeste da će naš računar pretražiti svoj *cache* (privremenu memoriju) kako bi proverio da nekim slučajem nije upisana ova stranica unutar *hosts* fajla (fajl koji reguliše razrešavanje na lokalnom računaru). Ukoliko ovaj upit ne bude pronađen, prelazi se na naredni korak, a to je da se upit šalje na *resolver* (server koji je dizajniran da uzima *DNS* upite i da ih razrešava, česta asocijacija sa ovim je internet provajder ili *ISP-Internet Service Provider*). *Resolver* će proveriti svoju *cache* memoriju i ukoliko ne pronađe naš upit šalje ga na *root* server. *Root* server uzima naš upit, ali neće dati konačan rezultat već će usmeriti *resolver* na *TLD* (*Top level domain* ovde se razrešava *.com .org* ...), slična stvar se dešava i na *TLD*, on će reći *resolveru* gde dalje da šalje upit.  
Poslednje odredište gde *resolver* pita za domen je server zadužen za naš upit, pošto *TLD* daje odgovor gde može naš upit da se nađe na osnovu autoritativnih *name* servera, koje daje kao odgovor. Autoritativni *name* serveri zaduženi su da znaju sve o našem upitu (domenu koji tražimo) i prema tome daju traženi odgovor *resolveru* koji ovo pamti u svojoj privremenoj (cache) memoriji i šalje informaciju nazad na naš računar odakle je ovaj upit i počeo.

*IP* adrese mogu da budu iz dve grupe *IPv4* ili *IPv6*, gde je *IPv6* novija verzija i trebala bi da zameni *IPv4* u bližoj budućnost zbog nedostatka *IP* adresa.

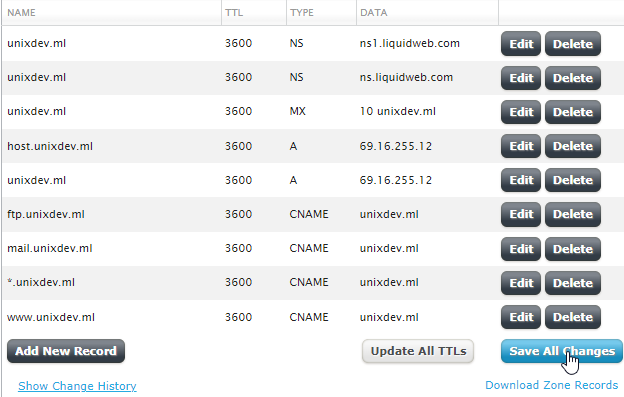
*DNS* je glavni gradivni element modernog interneta i može da se koristi analogija da je kao „telefonski imenik” - uzima ime i pretvara ga u telefonski broj. U našem slučaju, pretvara ime domena u *IP* adresu. Kada pretražujemo neki broj na našem telefonu vrlo često ćemo prelistati naš imenik, a zatim izabrati neki broj - ovo se zove lokalno skladište. Isti slučaj je i na računaru, gde on ima svoj lični lokalni fajl koji se zove *hosts* fajl i koji razrešava konekciju u zavisnosti kako je podešen.

Vrlo čest primer upotrebe *hosts* fajla je da organizacije podese ovaj fajl da se samo razrešava na imena koja se nalaze unutar organizacije i tako se kreira mala *DNS* mreža razrešavanja.

Uprkos ovome, na internetu stvari funkcionišu malo drugačije, imamo distribuirane *DNS* sisteme koji razrešavaju imena kao što su *google.com, facebook* itd…

Na distributivnim *DNS* sistemima, *DNS* rekordi (*records*) se razrešavaju u zavisnosti gde je *DNS* zona koja je zadužena za domen ili registar u zavisnosti od konfiguracije.

*Slika 52: DNS zona*



*izvor: liquidweb.com*

Iz primera podešavanja *records* (Slika 52), može da se primeti kako treba da se neki *records* razrešavaju na serveru.

*Slika 53: DNS zona podešavanje*



*izvor: liquidweb.com*

Svaki hosting ima svoj specifičan *UI*, ali uglavnom bi trebali da budu isti, s tim da postoje neke razlike prilikom postavljanja *CNAME*, npr. korišćenje @ simbola da ukazuje na domen.

Treba prvo početi od mogućih rekorda (*records)* koji mogu da budu na serveru, a to su:

* A
* AAAA
* CNAME
* MX
* PTR
* NS
* SOA
* SRV
* TXT

## 8.1 A record

*A record* ili adresni rekord, dodeljuje *IP* adresu domenu ili subdomenu.

Kada je *DNS* bio razvijen, prvobitno je bilo preporučljivo da dva A rekorda referiraju na istu *IP* adresu.

Na primer, imamo *nekidomen.tld* i želimo da dodelimo *10.10.0.1 IP* adresu našem *web* serveru, onda bi trebali da kreiramo *A rekord* sa *www.nekidomen.tld* kao *Fully Qualified Domain Name* i u polju vrednosti bismo postavili *IP adresu* servera *10.10.0.1*.

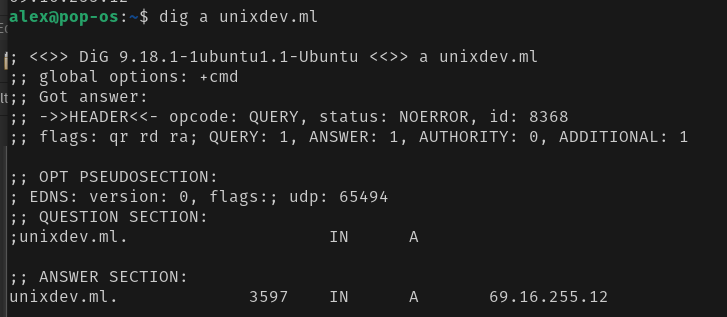
Pretpostavka je da će svi zahtevi za *www.nekidomen.tld* biti poslani na server sa *IP* adresom koju smo specificirali u polju vrednosti *DNS* rekorda.

Korisno je upotrebiti A rekord kada imamo subdomene koje se nalaze na različitim sistemima.

Ako bismo ovo podešavali preko A rekorda bilo bi dobro da koristimo *wild card* *\*.nekadomena.tld* A rekord koji nam omogućava da bilo šta stoji ispred, kao subdomen *nešto.nekidomen.tld* i zatim se razrešava u zavisnosti od *IP* koji mu je dodeljen u polju vrednosti (value field). Uglavnom se izbegava razrešavanje A rekordom pa je preporučljivo koristiti *CNAME* rekorde umesto *wildcard* A rekord.

Korisna komanda koja pomaže sistemskom administratoru da proveri rekorde neke domene je *dig* i to može da se uradi ovako:

*Slika 54: Komanda prikaza A rekorda domene*



Komanda daje način na koji pita domen šta joj je adresni *record*. Nakon što odredimo šta tražimo i za koji domen, imamo argument *a,* koji kaže komandi da traži samo A rekorde, naravano ovo može biti bilo šta drugo recimo *CNAME, MX, AAAA* …

Svako polje znači nešto, ako pogledamo *answer* sekciju videćemo *TTL* (time to live) koji je 3597 sekundi, onda imamo *IN* koje znači internet, poslednja dva polja odnose se na rekord i *IP* adresu ili *value* polje koje je popunjeno unutar *DNS* zone.

## 8.2 AAAA rekord

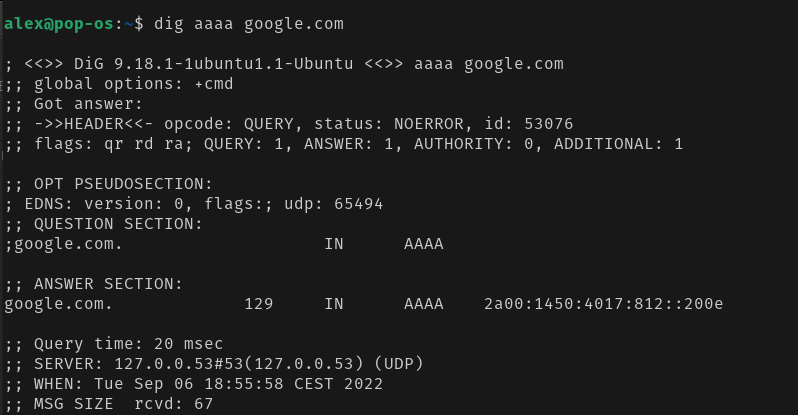
*AAAA* rekord ili *IPv6* adresni rekord mapira *hostname* na 128-bitnu *IPv6* adresu.

Regularni *DNS* adresni rekord ili *A* je definisan na 32-bita kao *IPv4* adresa. Zbog nedostatka ove verzije, razvijen je ovaj rekord koji će se poklapati sa *IPv6* adresama. Ovo nam omogućava da domene asociramo i ovako ako to želimo.

Četiri A slova su mnemonika (veština pamćenja) da nas asocira da je *IPv6* četiri puta veći od *IPv4*.

*AAAA* rekord je struktuiran na sličan način kao i A record, oba su binarni i master fajl format, samo što je *IPv6* veći…

*Slika 55: Komanda prikaza AAAA rekorda domene*



*AAAA* rekordi su tu da pomognu prilikom tranzicije i koegzistencije između *IPv4* i *IPv6* mreža. (IPv4 *nameserver* može da vrati IPv6 adresu)

## 8.3 CNAME rekord

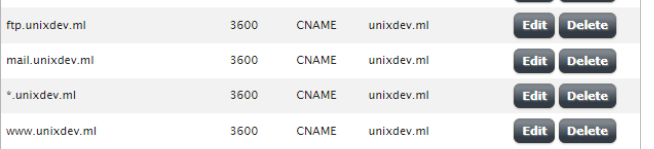
*CNAME* rekord ili *canonical name* rekord omogućava da domensko ime može da se predstavi kao *alias* druge (*alias* domen dobija sve subdomene i *DNS* rekorde originalnog).

*CNAME* bi trebao da se koristi, kada želimo da asociramo novi subdomen na već postojeći A rekord. Možemo uraditi recimo [*www.nekadomena.tld*](http://www.nekadomena.tld) na *nekadomena.tld*, koji bi trebalo da ima već postojeću dodeljenu *IP* adresu kao A rekord.

Ovaj metod podešavanja nam omogućava da imamo koliko želimo subdomena, bez da specificiramo *IP* na svaki rekord. Generalno pravilo je da se koristi *CNAME*, ako više servisa pokazuju na istu *IP* adresu. Na ovaj način, ako prepravimo A record, sve ostale domene servisi koji pokazuju na glavni A rekord prepraviće svoju *IP* adresu.

Pogledom na primer (Slika 56) može da se vidi par servisa koji koriste *CNAME* da bi pokazivali na isti *IP* kao A rekord glavne domene.

*Slika 56: Prikaz CNAME na hostingu*

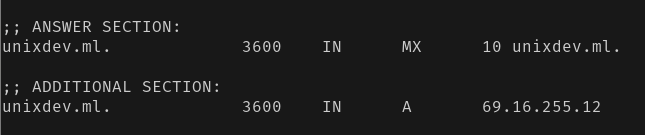
 *izvor: liquidweb.com*

Recimo za *ftp.unixdev.com* vidimo da je subdomen *FTP* servis koji je namenjen za istoimeni protokol koji će server koristiti. Nakon toga imamo *TTL* 3600 (u sekundama) i na kraju imamo *value* koji je *unixdev.ml* koji se odnosi na ime domena ili glavni A rekord.

## 8.4 MX rekord

*MX* rekord, ili *mail exchange* rekord mapira ime domena na listu *mail exchange* servera za taj domen, tj. ovaj rekord govori *mail* serveru gde treba da šalje imejlove.

*Slika 57: Struktura MX rekorda*



Kao što se može videti (Slika 57) *MX* rekord pokazuje da sve *mail* poruke idu na @ ili *unixdev.ml* i treba da budu rutirani na istoimeni domen, tj njegov lokalni *inbox*.

Takođe, *DNS* rekord pokazuje da je domen *unixdev.ml* lociran na IP 69.16.255.12 i da se rutira na taj *mail* server. Ovde se završava posao *MX* rekorda i posao preuzima *mail* server na toj *IP* adresi, on skuplja imejlove i distribuira ih na korisnike servera.

Prilikom pisanja ovih rekorda, bitna stvar je da ima tačku na kraju ( .) nakon imena domene u MX. U slučaju da nema ovu tačku na kraju, domen bi izgledao otprilike ovako *unixdev.ml.unixdev.ml.* Pregledom sekcija komande *dig* za *MX* rekord možemo primetiti broj 10, koji indicira preferencioni broj ili broj prioriteta, gde je niži broj veći prioritet. Odatle stoji da je *mail* uvek rutiran na server koji ima najveći prioritet ili najmanji preferencioni broj. (Ako imamo samo jedan *mail* server, sigurno je koristiti 0 kao najveći prioritet.)

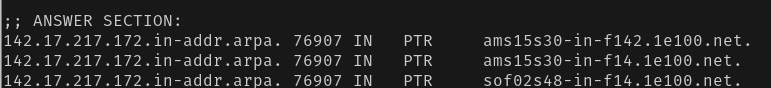
## 8.5 PTR rekord

*PTR* rekord ili *pointer* rekord mapira *IPv4* adresu na *canonical name* za hosta. Ovaj rekord se uglavnom koristi za bezbednost i *anti-spam* mere, gde većina *web* servera i *email* servera rade *reverse DNS lookup* da provere da li host zapravo dolazi od iste lokacije odakle prikazuje da dolazi. Preporučljivo je da imamo postavljen *PTR*, takođe je izuzetno bitno da imamo ovo postavljeno ukoliko pokrećemo *SMTP-mail* server. (Secure Mail Transfer Protocol – SMTP)

*Slika 58: PTR rekord na osnovu A rekorda sa prethodne slike*



*Slika 59: A rekord prikaz*



*IP* adresa je ispisana sa desna na levo i dodat je *in-addr.arpa* kao što se vidi (Slika 59). Ovo je sada na levoj strani gde je domen, a na desnoj strani, kao vrednost, su serveri namenjeni za *google* ili *google.com*.

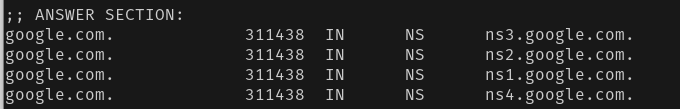
## 8.6 NS rekord

*NS* rekordi ili *name server* rekord mapira ime domena na listu *DNS* servera koji su autoritativni za taj domen, takođe i delegacija (gde će pretraga da vodi) zavisi od *NS* rekorda.

*NS* rekord ili autoritativni *name* server pokazuje gde su autoritativni *name* serveri za specifični domen. *NS* rekord autoritativnih *name* servera za bilo koji domen biće predstavljen na serveru „roditelj” (parent server). Ovi rekordi se zovu delegacioni rekordi i kao rekordi na serveru ,,roditelj” pokazuju delegaciju domena na autorativne rekorde.

*NS* rekordi koji mogu da se pronađu na serveru ,,roditelj” (parent) trebali bi da odgovaraju *NS* rekordima na autoritativnom serveru, ali postoje *NS* rekordi koji se vide na autoritativnom serveru i nisu isti kao na serveru ,,roditelj” (parent). Ova postavka je obično korišćena da se konfiguriše skriveni *name* server.

*Slika 60: Primer NS rekorda*



## 8.7 SOA rekord

*SOA* ili *State of Authority* rekord specificira *DNS* server koji daje autorativne informacije o domenu na internetu, *email* domenskog administratora, domenski unikatni serijski broj, nekoliko tajmera koji odgovaraju osvežavanju (refresh) zone.

*SOA* rekord je način da domenski administrator prikaže osnovne informacije o domenu, na primer, koliko puta se obnovi (update), kada je poslednja obnova rađena, kada da se proveri za više informacija, koji je *email* domenskog administratora itd…

*Zone* fajl može da sadrži samo jedan *SOA* rekord, tako da samo jedan može da poseduje domen, iz tog razloga često možemo videti da je ime zauzeto, recimo, kada kreiramo naš domen.

Dobro podešeni *SOA* rekord može da smanji upotrebu mreže (bandwidth) između *name* servera, poveća brzinu kojom može da se pristupu stranici, omogući stranici da bude aktivna čak i kada je glavni *DNS* server van funkcije.

*Slika 61: SOA rekord*



Prvo polje je ime ili *google.com* koje je pretraženo, drugo polje je *time to live* koje je 43 sekunde, *IN* stoji za internet što je implikacija da je domen na internetu.

*SOA* je tip rekorda, nakon toga imamo *name* server gde se domen nalazi i nakon toga imamo onoga kome odgovara *email*.

Imamo serijski broj koji je 472681414 koji pokazuje kada je poslednja revizija izvršena. Trebalo bi da je format *yyyymmddnn* (nn je broj revizije) nakon toga imamo *refresh rate* u sekundama koji je 900, *update retry* koji je 900, isticanje (expiry) koji je 1800 i 60 koji stoji kao minimum ili osnovno vreme koliko bi trebali potčinjeni serveri da čuvaju *zone* fajlove.

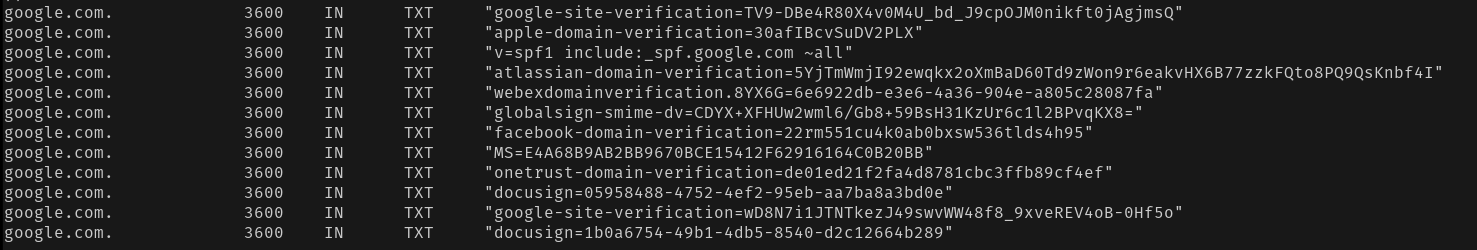
## 8.8 SRV rekordi

*SRV* su rekordi resursa i treba da nam daju povratne informacije o serverskim servisima. Teorija iza *SRV* rekorda jeste da ako imamo ime domena (domain name) *primer.com* i neki servis, recimo *http*, koji se pokreću na *tcp* u ovom slučaju, *DNS* upit će biti izvršen da se pronađe *hostname* koji daje ove informacije o domenu koje mogu, a ne moraju, da budu unutar domena.

## 8.9 TXT rekordi

*TXT* rekord dozvoljava administratoru da ubaci proizvoljan tekst u *DNS* rekord. Na primer, ovaj rekord se koristi da se implementiraju *SPF* i *DKIM* rekordi autentifikacije i njihove specifikacije.

*Slika 62: Primer sintakse TXT rekorda*



Prikazan je i prethodno pomenut *SPF* rekord ovde:

*v=spf1 include:\_spf.google.com -all* ovo se koristi za verifikaciju *email* servera.

# 9. Apache veb server i baze podataka

U poslu sistemskog administratora često se susrećemo sa veb serverima, možda jedan od najzastupljenijih veb servera na internetu je *Apache* koji se pokreće kao servis unutar serverskog računara. Pored samog *Apache,* kao jedan od izuzetno popularnih stoji i *Nginx*, ali pošto bi trebalo da se odlučimo za jedan, izabrao sam *Apache* pošto je meni bliži. Naredni odeljci će proći kroz osnovnu konfiguraciju jednog *Apache* servera koji bi se postavljali na našem serverskom računaru.

## 9.1 Instalacija httpd (Apache) na CentOS

Na *CentOS* sistemima *Apache* je *httpd* i instalacija softvera se vrši preko *yum package manager*.

Upotrebom upravljača paketima za *CentOS* možemo instalirati *Apache* bez nekih većih komplikacija, komanda koja bi bila upotrebljena za ovaj zadatak je sledeća:

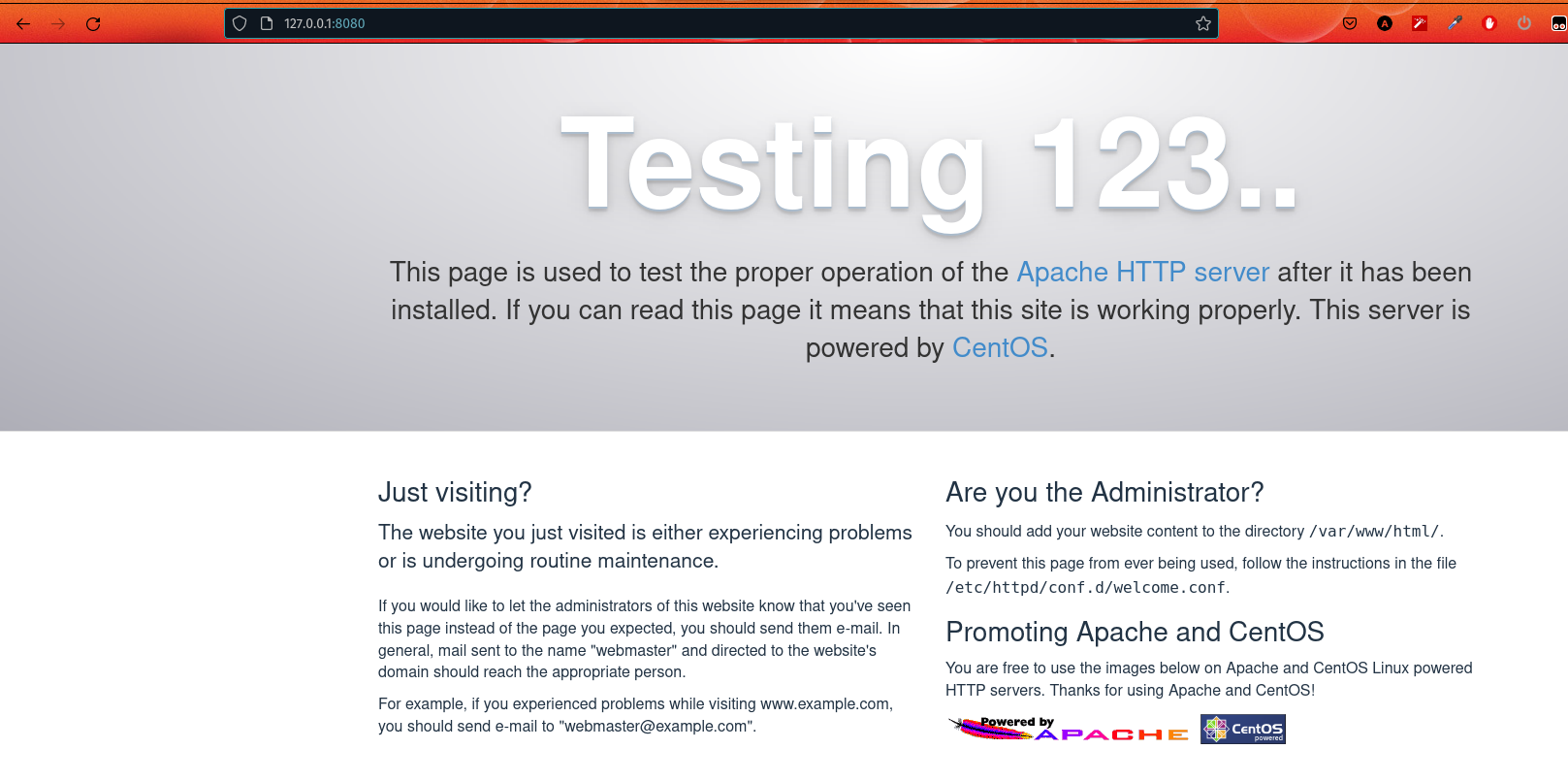
**$ yum install httpd -y**

Nakon toga treba da omogućimo servis, tj da podesimo da servis bude stalno aktivan prilikom pokretanja sistema, recimo ovo možemo porediti sa aplikacijama koje se uključuju kada pokrenemo *Windows*.

**$ systemctl enable –now httpd** (*now* stoji da odmah ponovo pokrene i aktivira servis)

Kada podesimo *servis/daemon* trebalo bi da se prikaže *index* stranica koja pokazuje da je uspešno podešen *httpd servis*. ( zahteva port forwarding ako želimo da vidimo prikaz sa host računara, a ne sa virtualne mašine )

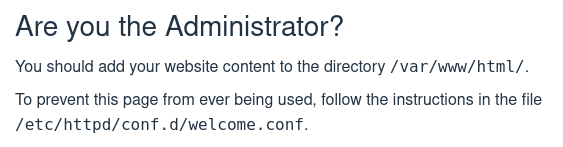
*Slika 63: Apache početna stranica*



### 9.1.1 Osnove Apache konfiguracije

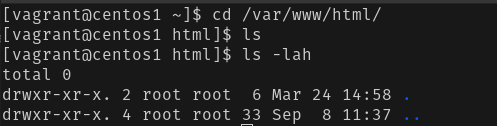
Pogledom na stranicu uočava se sledeća poruka:

*Slika 64: Poruka gde se nalazi konfiguracija od Apache stranice*



Primećujemo da *Apache* nudi gde treba sadržaj stranice da se nalazi. Kao što se vidi (Slika 65), sam direktorijum je prazan, pa je nepohodno obezbediti *index.html* koji će nam prikazati neke promene.

*Slika 65: Sadržaj www/html direktorijuma*



(Primer kako bi kreirali fajl preko terminala i uneli neki tekst unutar fajla istovremeno)

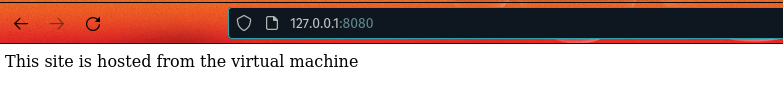
**$ cat << HERE | tee -a /var/www/html/index.html** (komanda za upisivanje)

**This site is hosted from the virtual machine**

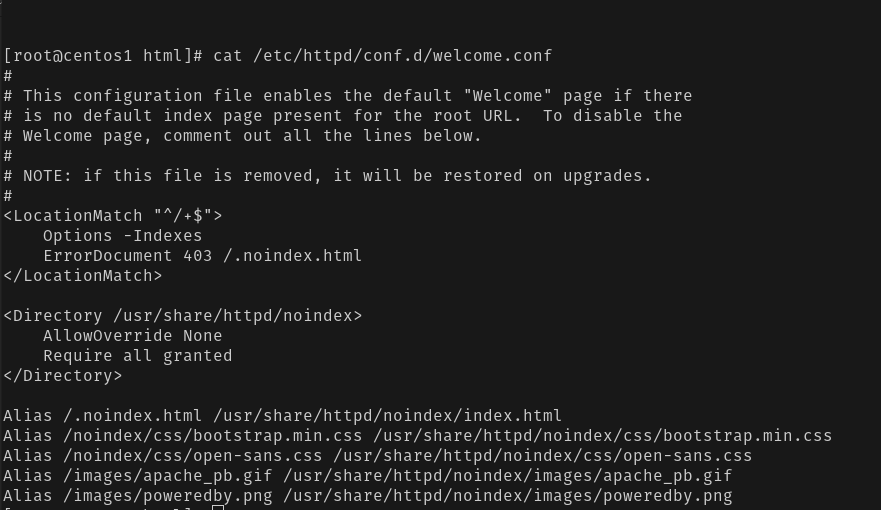
**HERE** (Ovo je indikacija terminalu da je kraj unosa teksta na ovoj tački)

Kreiramo fajl *index.html* koji će sadržati informaciju da li se ovaj sajt zapravo nalazi na virtualnoj mašini. Samom kreacijom ovog fajla *Apache* server će ovo očitati i promeniti sadržaj na stranici.

*Slika 66: Prikaz index.html stranice*

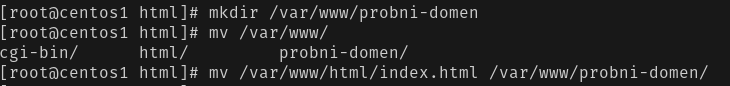


*Slika 67: Welcome konfiguracioni fajl*



Konfiguracioni fajl *welcome.conf* uključuje podrazumevanu stranicu *Welcome*, ako ne postoji *index.html* unutar zadatog direktorijuma odakle iščitava fajlove. Ukoliko postoji velika količina vebsajtova (virtualnih hostova) na jednom hostu, nije idealno da se samo zada *index.html*. Zato možemo da napravimo direktorijum za vebsajt odakle će se čitati informacije. Prvo što treba da se uradi jeste da kreiramo direktorijum za naš domen i pomerimo *index.html* fajl u njega. Zatim moramo da kreiramo konfiguracioni fajl za ovaj domen u direktorijumu kao */etc/httpd/conf.d/probni-domen.conf* koji će govoriti odakle će čitati ovaj domen.

*Slika 68: Kreiranje docroot domene*



*Slika 69: Kreiranje fajla probni-domen.conf i njegov sadržaj*



### 9.1.2 Funkcionalnost Apache konfiguracije

*Slika 70: Prikaz DocRoot u httpd.conf*



Razlog zbog kojeg fajl može da se prebaci u direktorijum */var/www/html* i možemo da ga prikažemo u pretraživaču je podešavanje *DocumentRoot* unutar *Apache* konfiguracije. *DocumentRoot* se isčitava iz */var/www/html*, a razlog zašto *Apache* iščitava *index.html* je sledeći:

*Slika 71: Prikaz indeksiranja fajlova koji se prvi čita*



Ova linija obrađuje koji fajl će biti učitan kada bude zatražen direktorijum. Bez obzira što imamo definisane varijable u */etc/httpd/conf/httpd.conf* možemo da dodamo konfiguraciju za veb sajtove pod direktorijumom, što se može videti u primeru sa *virtual* *hostom*.

## 9.2 MySQL, Maria DB baze podataka

Tradicionalno, baze podataka su odlično mesto za skladištenje uređenih podataka specifičnog tipa i veličine, implikacija je da možemo imati bazu podataka za bilo šta (Adam K Dean Administracija Linux sistema, 669). Pošto su uglavnom poznatije relacione baze koristićemo *MySQL* baze, no ne baš *MySQL* pošto je to originalno ime koje sada pripada *Oracle* firmi, već *MariaDB,* što je grana ove baze i dostupna je svima koji žele da je koriste zbog *Open-Source licence*. *MySQL* baze su generalno poznate i široj javnosti zato što jedna od većih aplikacija za servisiranje sajtova kao što je *WordPress* koristi ovu bazu. Dobra praksa sa softverom je da se instalira pre nego što se počne upotrebljavati, instalacija se vrši na sledeći način:

*Slika 72: Instalacija MariaDB servera preko yum*



Moramo aktivirati servis, slično kao i sa *Apache* servisom:

*Slika 73: Aktiviranje MariaDb servisa*



Ovo je teorijski kraj instalacije, baza je aktivna. U narednom odeljku ćemo prolaziti kroz osnovne bezbednosne konfiguracije.

### 9.2.1 Podešavanje sigurne instalacije

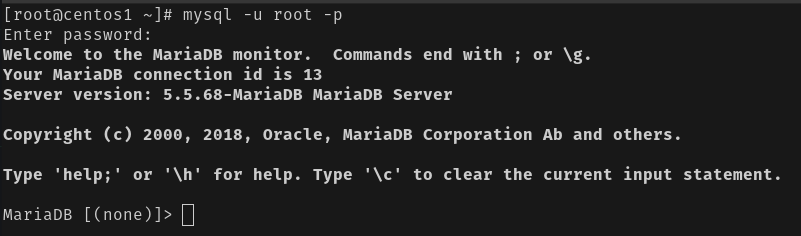
Pošto instalacija *MySQL* sama po sebi ne podešava nikakavu lozinku za *root* korisnika, uglavnom je preporučljivo da se pokrene skripta koja vrši osnovno bezbednosno podešavanje, a to je *mysql\_secure\_installation*.

Ova skripta će pitati niz pitanja koja su:

* *Enter current password for root:* (ovo možemo ostaviti prazno pošto nema *password*)
* *Set root password: Y*
* *New password: nekipass*
* *Remove Anonymous users: Y*
* *Dissallow root login remotely: Y*
* *Remove test database and access to it: Y*
* *Reload privilege tables now: Y*

Nakon osnovnog podešavanja trebalo bi da nam bude omogućeno da pristupimo bazi:

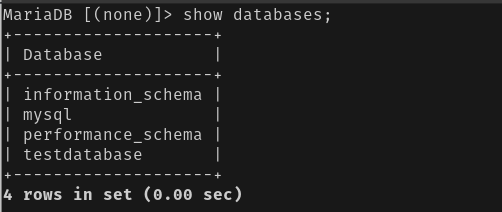
*Slika 74: Login na MySQL servis*



### 9.2.2 Listanje, kreiranje i odabir podataka i tabela

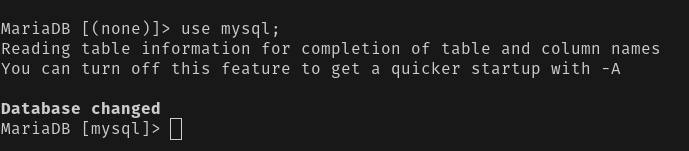
Bazom može da se upravlja iz terminala, koji se otvara pokretanjem iz serverskog *shella*. Osnovna komanda da proverimo koje baze su aktivne na serveru bila bi:

*Slika 75: MySQL command shell*

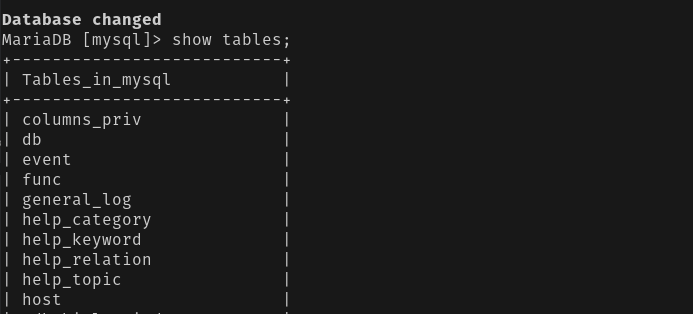


Ili, na primer, želimo da selektujemo neku od ponuđenih baza, možemo da koristimo sledeće:

*Slika 76: Prikaz use komande u mysql*



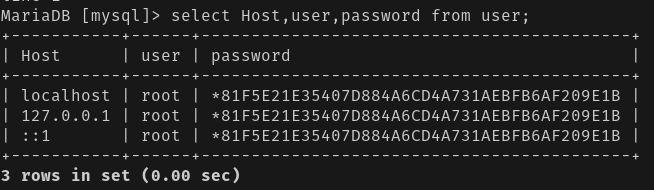
*Slika 77: Promena baze podataka i prikaz tabela*



Može da se primeti da sve ovo može da se radi iz terminala bez ikakvog pristupa nekom eksternom servisu kao što je recimo *phpMyAdmin* koji može da se podesi na server.

Neke česte komanda koje se koriste u *mysql* bazama su takođe *insert, select, create database drop.*

*Slika 78: Selekcija vrednosti iz tabele user*



Ovde smo selektovali *host, user, password* iz tabele *user* (Slika 78) da se prikažu sadržane informacije.

*Slika 79: Upotreba create database komande*



*Slika 80: Upotreba drop database komande*



Upotreba komandi koje bi sistemski administrator koristio prilikom kreacije baze (create) ili brisanja baze (drop).

Isto tako postoje i komande za kreiranje i unošenje u tabele sa kojima često radi sistemski administrator:

*Slika 81: Upotreba create komande za tabele*



*Slika 82: Upotreba insert komande za unos vrednosti u tabelu*



Takođe, postoje i dozvole za baze podataka, koje su važne koliko i regularne dozvole fajl sistema.   
Na primer, ne želimo da dve *wordpress* instalacije na istom hostu mogu da čitaju istu bazu podataka međusobno, pa se zato kreiraju posebni korisnički nalozi za svaku od njih i dodeljuju im se posebne baze podataka unutar *MariaDB*. Na zvaničnoj dokumentaciji postoje sve moguće implementacije ovih dozvola za korisnika:  
*https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html*

(Wordpress, 2022)

# 10. Pravljenje osnovne konfiguracije domene upotrebom WordPress CMS

## 10.1 Uvod

*WordPress*, možda i najpopularniji *CMS* (Content Management System) za domene, vrlo često je viđen i na zvaničnim hosting kompanijama kao osnovna instalacija za server.

Izabran je jer se cela instalacija može obaviti veoma brzo uz prethodno instalirane servise kao što je *MySQL* (*MariaDB*) i *httpd* (*Apache*) koji su prikazani u prethodnim poglavljima.

Pošto je *WordPress* baziran na *PHP* neophodno je obezbediti odgovarajuće biblioteke a i sam *PHP*, koji bi trebalo da bude oko verzije 7.4, koja je relativno nova, ali ima još par iznad ove. Na zvaničnom repozitorijumu *CentOS* u *yum*, *5.6 php* verzija je dostupna i zato moramo da dobavimo određeni repozitorijum koji sadrži novije verzije *PHP* instalacija.

## 10.2 Instalacija Wordpress aplikacije

Prvo moramo instalirati pakete za *remi* repozitorijume:

*Slika 83: Instalacije epel-release paketa*



*Epel release* repozitorijum omogućava operativnom sistemu da raspolaže sa više paketa koji se često koriste sa popularnim softverima*.* (www.redhat.com)

Dodajemo repozitorijum u listu kako bismo mogli da ga koristimo sa *yum* upravljačem paketima:

*Slika 84: Dodavanje repozitorijuma za instalaciju*



Instaliramo *yum* alatke koje će omogućiti da podešavamo verzije *PHP*:

*Slika 85: Instalacija dodatnih yum alatki*



Stavljamo *PHP 7.4* kao fabričku verziju koristeći novoinstalirane alatke iz paketa *yum-utils:*

*Slika 86: Uključivanje remi-php74 iz repozitorijuma*



U suštini, *yum* preuzima 7.4 verziju sa repozitorijuma koji smo prethodno specificirali. Tu se pokazala korisnom alatka *yum utils*.

Instaliramo dodatne biblioteke za *PHP* koje će kasnije biti korisne za *WordPress*:

*Slika 87: Instalacija PHP paketa i biblioteka*



Nakon instaliranih paketa i biblioteka server bi trebalo da pokreće *PHP* verziju koja mu je bila specificirana.

*Slika 88: Provera verzije PHP*



(Takođe nije loše ponovo pokrenuti *httpd* nakon ovih izmena *systemctl restart httpd*)

Pošto je *wordpress* instalacija dostupna svima sa zvaničnog sajta, počinjemo odatle.

Pre svega treba dobaviti paket, ali u *CLI* (command line interface) nemamo *browser*, sem ako to ne napravimo nekako, no postoji mnoštvo komandi koje će nam omogućiti isto.

Naravno moramo imati link odakle ćemo preuzeti neophodne sadržaje:

**https://wordpress.org/latest.zip**

Komanda *wget* jedna je od poznatijih, ali imamo i *curl* koji dolazi kao fabrička alatka i možemo njega da koristimo.

Preuzećemo paket *WordPress* u direktorijum */var/www/html/* jer je to mesto odakle *Apache* iščitava podatke.

*Slika 89: Preuzimanje wordpress fajla sa interneta upotrebom wget*



Trebalo bi da dobijemo ovakav podatak:

*Slika 90: Prikaz preuzetog fajla*



Upotrebom bilo komande *tar* ili *unzip* možemo izvući podatke iz ovog fajla.

Kada to uradimo, dobijamo direktorijum *wordpress*:

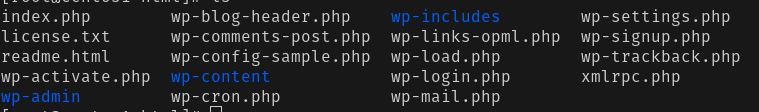
*Slika 91: Prikaz otpakovanog fajla*



Pošto *Apache* isčitava podatke u zavisnosti kako mu je podešen *docroot*, ili odakle će čitati konfiguracije, nephodno je fajlove iz *WordPress* kopirati na *root* nivo. (*Virtual host* odeljak pokazuje kako bi bila napravljena drugačija konfiguracija.)

Kada izvučemo sve fajlove trebalo bi da se dobije sledeće:

*Slika 92: Prikaz otpakovanog fajla*



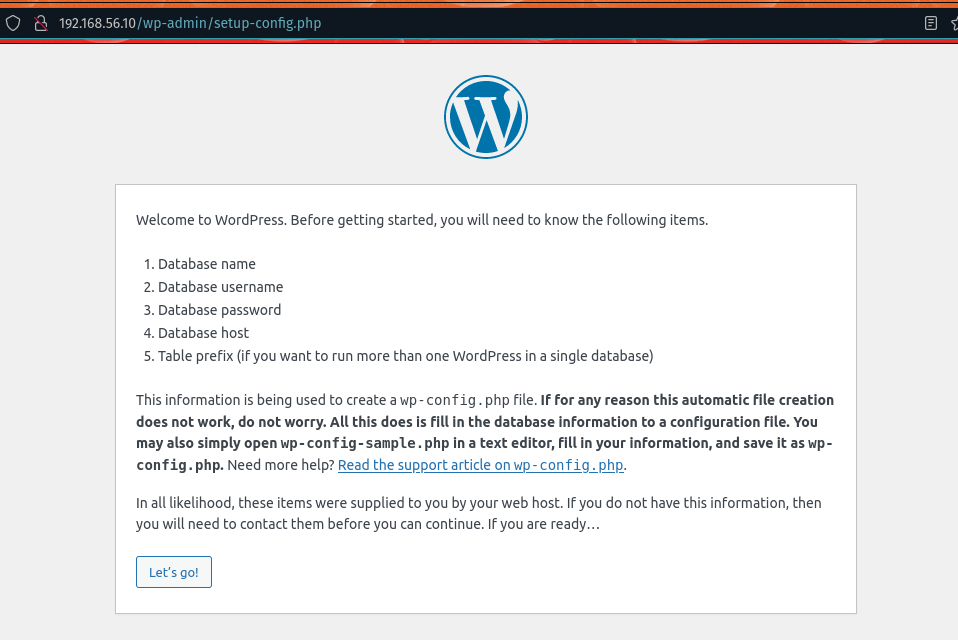
Naravno, ovo sada izgleda ovako, ali korišćene su *Linux* komande kako bi se pomerali fajlovi, recimo:

*Slika 93: Brisanje wordpress direktorijuma*



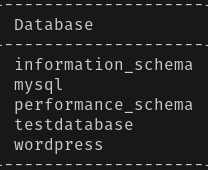
Nakon ovih podešavanja pozivom određenog *Wordpress* fajla trebali bismo dobiti sledeću stranicu:

*Slika 94: Prozor instalacije WordPress*

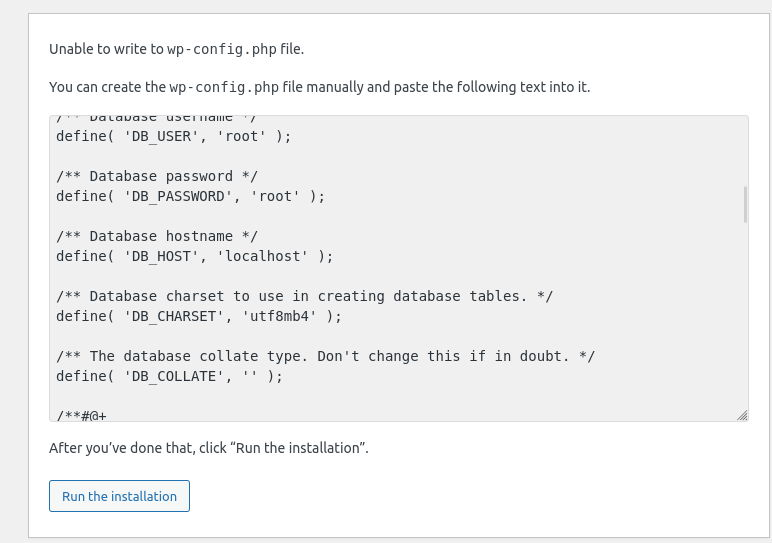


Instalacija je prilično jasna i jednostavna i nepotrebno je detaljisanje o svakom koraku. Morali smo pristupiti fajlu *setup-config.php* koji se nalazi u direktorijumu *wp-admin* kako bismo vršili ovu instalaciju. Moraćemo i da napravimo konekciju ka bazi da bi ovo uopšte funkcionisalo. U *WordPress* postoji *wp-config.php* koji će nam pomoći u ovome, ali i pre toga treba kreirati bazu na serveru.

*Slika 95: Prikaz kreirane baze za wordpress*



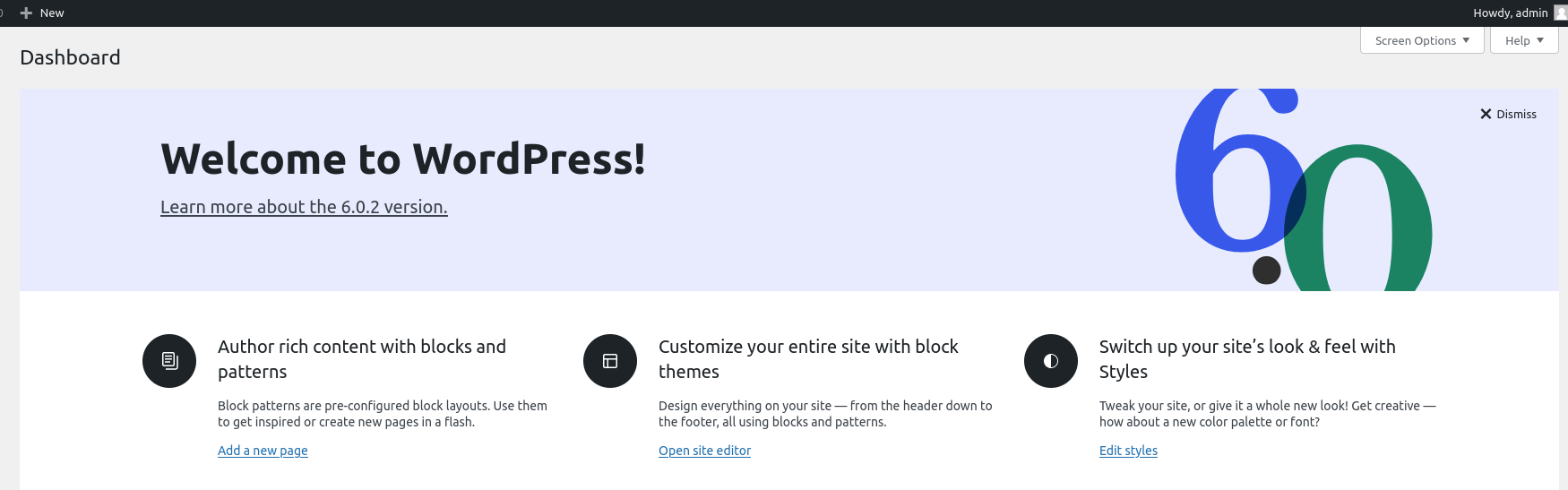
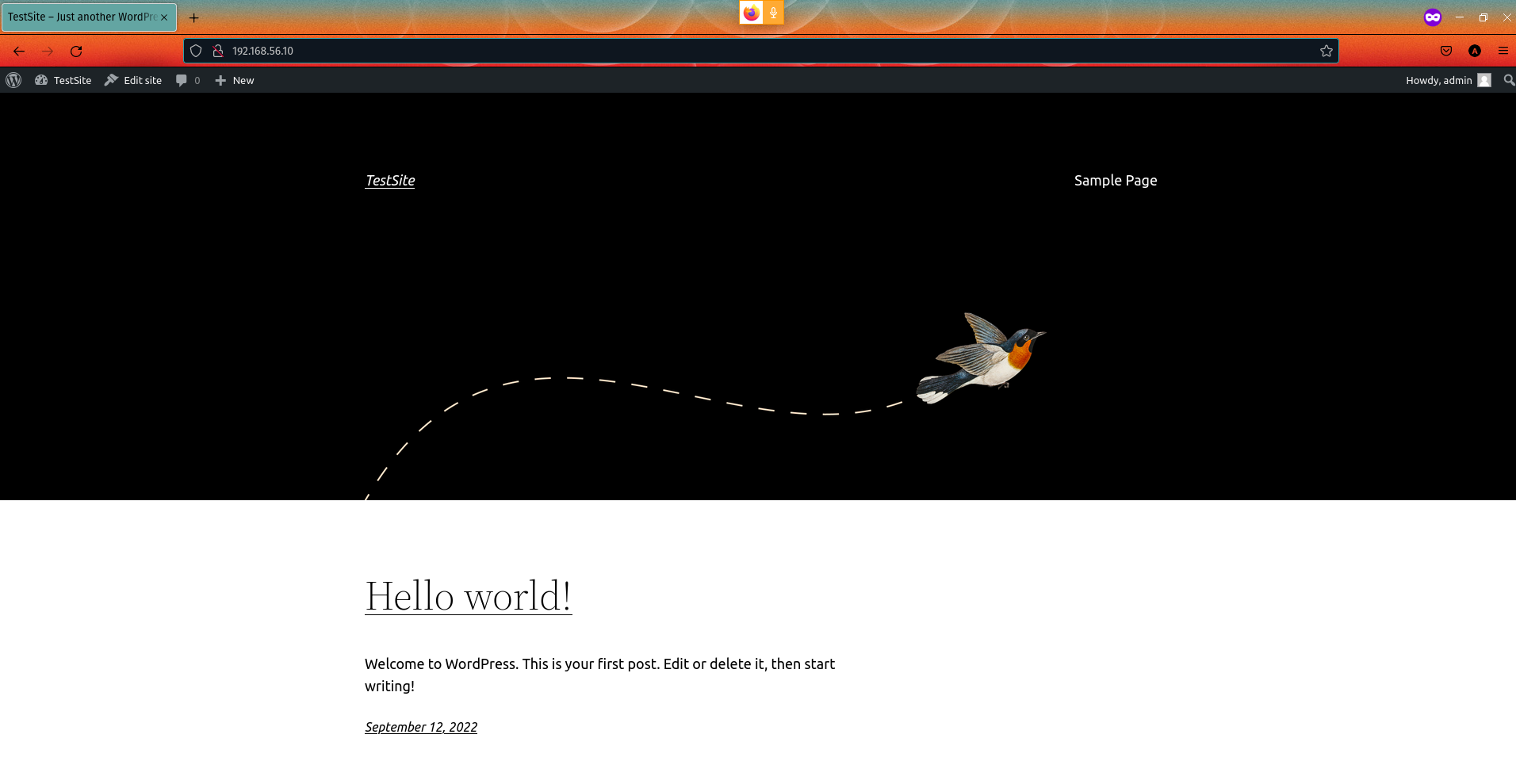
*Slika 96: WordPress konfiguracioni fajl*



Kada prođemo kroz upitnik koji nam daje *WordPress* prilikom instalacije, dobijamo primer *wp-config.php* fajla koji bi trebalo da kreiramo na *root* nivou *WordPress* instalacije i da unesemo zadati sadržaj.

Nakon popunjavanja *wp-config.php* i pokretanja instalacije dobijamo još jedan upitnik sa kojim završavamo instalaciju (preskačemo ga ovde pošto je samo forma koju korisnik popunjava). Nakon uspešne instalacije i kada unesemo zadate informacije koje *WordPress* traži dolazimo do sledeće stranice sa kojom je završena instalacija:

*Slika 97: Admin stranica i početna stranica WordPress nakon instalacije*

*Wordpress* nam nudi mnoštvo mogućnosti, ali ima i svoje mane. Ukoliko je klijent u mogućnosti pribegava se *custom* (ručno pisanim) sajtovima što hakerima otežava da kompromituju domen. *WordPress* sam po sebi je siguran, ali može da se poremeti malicioznim *injection code* napadima češće nego neki *custom* kôd napisan za istu svrhu.

# 11. Shell skripte

## 11.1 Uvod

*Shell* skripte su niz komandi koje se izvršavaju u cilju da se automatizuju neki zadaci na serveru.

Same po sebi, skripte su jednostavne kao i način na koji se pišu, dosta se pozajmljuje iz jezika C na kojem je napisano i samo *Linux* jezgro (kernel).

Osnovno razmišljanje iza ovih skripti jeste da ako znamo da koristimo komande, možemo pisati skripte.

## 11.2 Početak pisanja skripti

Svaka skripta počinje sa *shebang* i putanjom do *shella* koji želimo da skripta koristi, a to izgleda ovako:

**#!/bin/bash**

**# Početak neke skripte**

Ova kombinacija simbola *#!* govori našem interpreteru kako da izvršava skriptu, ovo može biti i recimo *#!/bin/python*, ako želimo da se skripta izvršava kao *Python* kôd. Ako se pogleda na drugu liniju, možemo videti sličan početni simbol *#* s tim što se ovo ne čita kao početak za interpreter već kao komentar u samom fajlu.

Recimo da želimo da napravimo skriptu za pretragu fajlova većih od 1M *home/alex/Videos* direktorijuma upotrebom *find* komande. Da ne bismo morali stalno da pišemo punu komandu, napisali bismo sledeće:

#!/bin/bash

#Find skripta

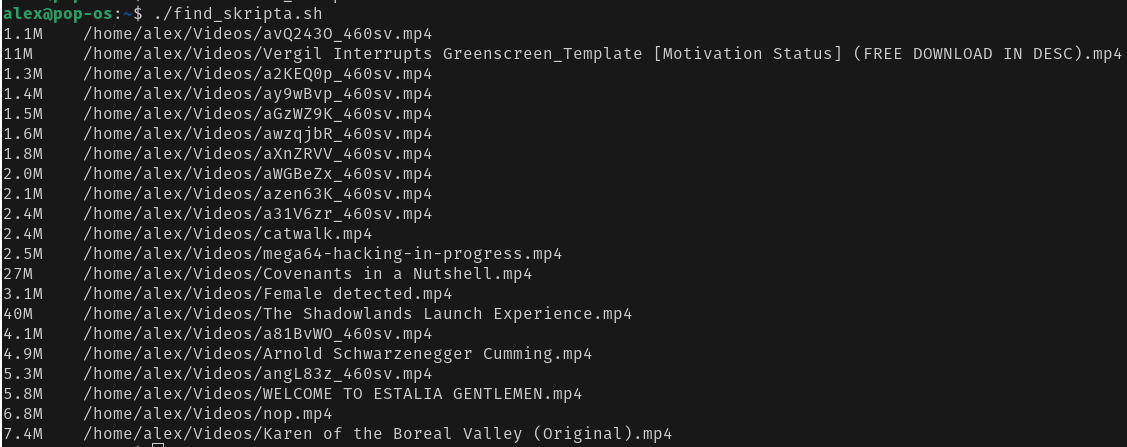
find /home/alex/Video -maxdepth 1 -type f -size +1M -exec du -h {} + | sort

Nakon napisane skripte nephodno je dati određene dozvole koje omogućavaju korisniku da pokreće skriptu, treba dati dozvole izvršavanja.

To se može izvršiti na sledeći način ***chmod +x find\_skripta.sh***. Ovo će omogućiti svim korisnicima koji su trenutno registrovani na operativnom sistemu da pokreću ovu skriptu.

Kada izvršimo skriptu, dobijamo sledeći rezultat:

Slika 98: Rezultat izvršavanja find\_skripta.sh



Skripte se mogu pokretati na dva načina: bilo dolaskom do direktorijuma gde se nalazi ili upotrebom pune putanje do fajla.

*./find\_skripta.sh*

ili

Možemo dati pun *path* (full path) do skripte

*/home/alex/find\_skripta.sh*

Rezultat je isti u oba slučaja.

### 11.2.1 Varijable

U realnom slučaju skripta je korisna, ali ne želimo da sve bude čvrsto postavljeno, recimo, želimo da menjamo vrednosti pretrage.

U ovim slučajevima korisne su varijable koje će nam omogućiti ovu slobodu. Pisanje varijabli u *shell* skripti ima svoju konvenciju, a to je da imena moraju biti pisana velikim slovima *IMEVARIJABLE*. Poželjno je da to radimo ako želimo da delimo našu skriptu sa nekim, u suprotnom nije potrebno.

Forma pisanja varijable je sledeća:

*MESTO\_PRETRAGE=”unosimo nešto šta želimo”*

Prikazaćemo ovo i na primeru, koristićemo prethodno napravljenu skriptu da demonstriramo upotrebu varijabli:

#!/bin/bash

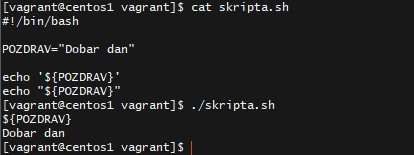
find "${MESTO\_PRETRAGE}" -maxdepth 1 -type f -size +1M -exec du -h {} + | sort

Varijabla sadrži putanju koju *find* treba da koristi, a unutar same *find* komande pozvana je ova varijabla kako bi se primenila na mestu gde se zahteva putanja pretrage sa *${IME\_VARIJABLE}*. Vitičaste zagrade su dobra konvencija ako želimo da razdvojimo dve varijable. Recimo da nam se putanja sastoji iz dva dela,

*“${MESTO\_PRETRAGE}${MESTO\_PRETRAGE\_NIZE}”*

dupli navodnici govore da treba da se čitaju varijable ako ih ima, tako da ne bismo trebali da ih mešamo sa jednostrukim navodnicima gde govorimo skripti da isčitava sve kako piše. Upotrebom jednostrukih navodnika dobićemo ime varijable kao neki *string*, što je loše ako želimo da upotrebimo varijablu u našoj skripti.

*Slika 99: Upotreba jednostrukih i dvostrukih navodnika*



### 11.2.2 Uzimanje unosa iz terminala

Skripte koje nemaju interakciju sa korisnikom su korisne, ali šta ako nam trebaju neke informacije svaki put kada se skripta pokreće? Ovde je korisno da uzimamo argumente sa terminala prilikom pokretanja skripte ili da koristimo *read* komandu.

Želimo da napišemo skriptu koja će samo odštampati naše ime, to bi trebalo da izgleda ovako:

#!/bin/bash

echo "Pozdrav ${1}"

*Slika 100: Prikaz upotrebe argumenta*



Skripta je uzela prvi argument *${1}* koji joj je prosleđen, i odštampala ga je kako smo napisali komandu ispisa.

Kao što je prethodno pomenuto, možemo da nateramo skriptu da traži naš unos kako bi obavila neki zadatak, a to može da se uradi na sledeći način:

Koristićemo prethodno napisanu skriptu sa *find* komandom kako bi se predstavila upotreba:

#!/bin/bash

# Zahtev unosa od korisnika za pretragu

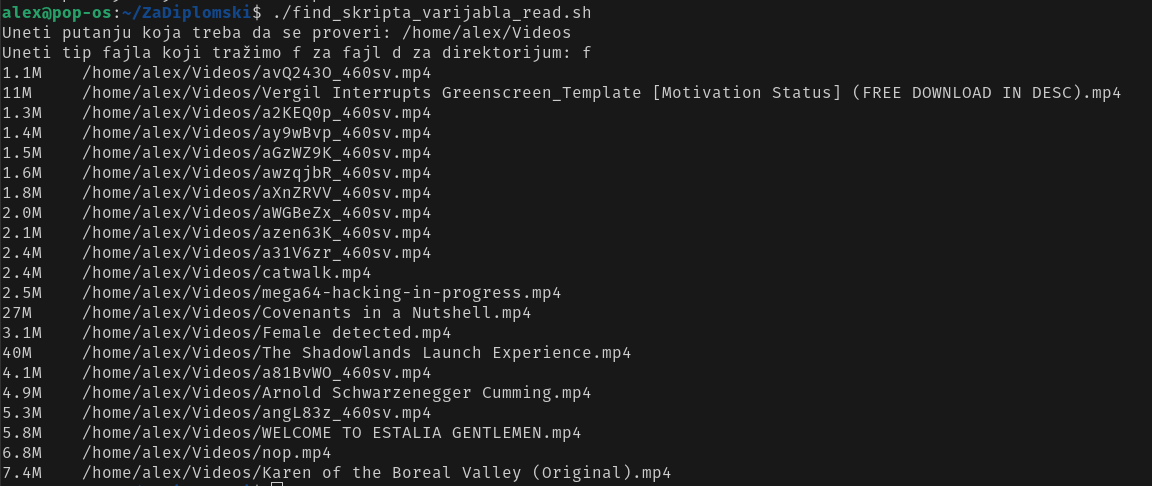
read -p "Uneti putanju koja treba da se proveri: " MESTO\_PRETRAGE

read -p "Uneti tip fajla koji tražimo f za fajl d za direktorijum: " TIP

find "${MESTO\_PRETRAGE}" -maxdepth 1 -type "${TIP}" -size +1M -exec du -h {} + | sort

Rezultat će biti sledeći:

*Slika 101: Rezulat upotrebe find skripte sa varijablom*



Dobija se isti rezultat, ali upotrebom *prompta* koji traži da unesemo nešto kako bi se izvršila skripta.

### 11.2.3 Kondicionali

Kondicionalni izrazi omogućavaju delu koda da se izvrši ukoliko je neki uslov ispunjen. Suprotno tome daju rezultat i ukoliko uslov nije ispunjen. Kondicionali rade oponašajući logiku čovekovog razmišljanja. Na primer, hoćemo da vidimo sve fajlove koji imaju ime *\*.mp4.* Implikacija je da tražimo sve što počinje sa bilo čim, a završava se sa tipom fajla *.mp4*. Ukoliko je ovaj uslov ispunjen, dobićemo ispis da postoje fajlovi koji imaju tip *.mp4*, ukoliko uslov nije ispunjen, dobićemo ispis da ne postoje ovakvi fajlovi.

Drugim rečima, kondicional o kojem je ovde reč je *if* grananje, koji je poznat pojam i u drugim programskim jezicima kao što su *Python, Perl, C, C++* ...

Osnovna forma pisanje *if*  kondicionala je sledeća:

if [ neki uslov ]

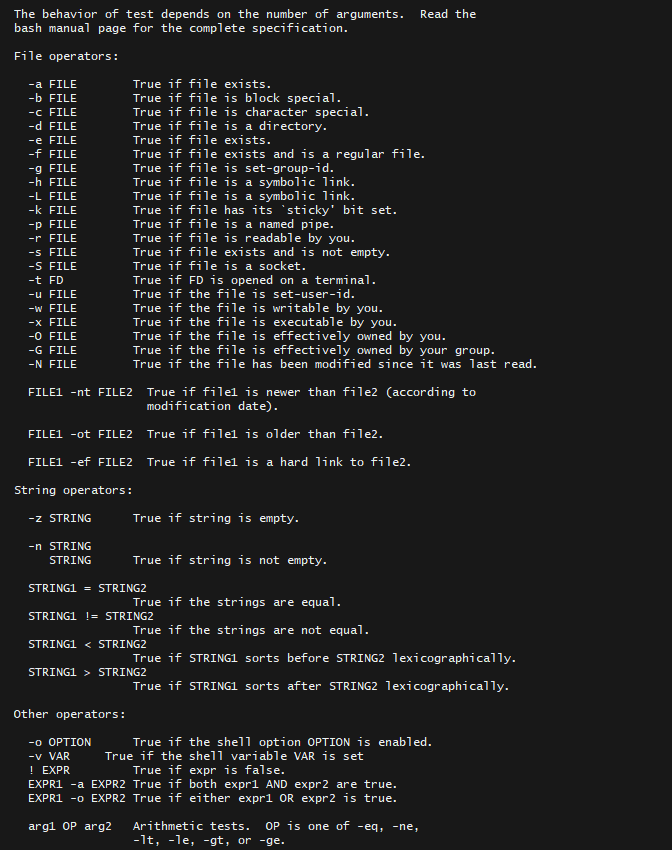
then

komande

fi

Dodatna pomoć koju nam *Linux* sistemi nude su upotrebe specijalnih karaktera unutar uslova, koje možemo dobiti prilikom poziva ugrađene komande *help test*. Ovi parametri olakšavaju ispitivanje određenih kriterijuma koje postavimo.

*Slika 102 : Parametri koji mogu da se upotrebe unutar if*



Kroz naredne primere predstavićemo upotrebu *if* kondicionala kako bismo testirali da li korisnik zapravo pokreće skriptu kao *root* korisnk.

#!/bin/bash

# Skripta za pretragu direktorijuma ili fajla

# Provera da li je korisnik zpravo root

if [[ "${UID}" -ne 0 ]]; then

echo "Skripta nije pokrenuta kao root korisnik " >&2

exit 1

fi

# Zahtev unosa od korisnika za pretragu

read -p "Uneti putanju koja treba da se proveri: " MESTO\_PRETRAGE

read -p "Uneti tip fajla koji tražimo f za fajl d za direktorijum: " TIP

find "${MESTO\_PRETRAGE}" -maxdepth 1 -type "${TIP}" -size +1M -exec du -h {} + | sort

Upotrebom kondicionala koji ispituje da li *UID* nije jednak nuli možemo da zaključimo da li je ovaj fajl pokrenut kao *root* korisnik ili običan korisnik (UID root je 0 a UID korisnika je 1000 ili u tim brojkama).

*Slika 103: Pokretanje kao običan korisnik*



*Slika 104: Pokretanje skripte kao root korisnik upotrebom sudo komande*



U primeru je navedena samo jedna upotreba *if* kondicionala. U zavisnosti od toga šta programer skripte želi da ostvari mogućnosti su mnogo veće.

### 11.2.4 Petlje

Petlje kao i kondicionali su česta pojava prilikom pisanja skripti. Ova specijalna funkcionalnost omogućava delu koda, ili skripte, da se izvršava više puta, što pomaže piscu da skrati ponavljajuće zadatke. Postoje dve vrste petlje, *while* i *for* petlje, gde se *while* petlje izvršavaju sve dok je uslov koji je postavljen zadovoljen, dok se *for* petlje izvršavaju samo onoliko puta koliko je navedeno.

Sintaksa pisanja *while* petlje unutar *bash* skripti je sledeća:

while [[ uslov ]]

do

script

done

Sintaksa *for* petlji u *bash* skriptama:

for $i in (može da stoji niz ili neka komanda koja će nam dati neki niz)

do

skripta

done

Primer upotrebe petlje za razrešavanje zadatka skladištenja rezervne kopije *.sh* fajlova:

#!/bin/bash

#Skripta za backup .sh fajlova

read -p "Unesite kako želite direktorijum za backup da se zove: " IME\_BACKUP

DATUM=$(date +%Y%m%d)

if [[ ! -d ${IME\_BACKUP} ]]

then

mkdir ${IME\_BACKUP} >&2

for i in \*.sh #Pronađi sve fajlove koji se završavaju sa .sh

do

[[ -e "$i" ]] || break #prekida izvršavanje ukoliko nema fajla

mv "$i" ${IME\_BACKUP}

done

tar cvf backup${DATUM}.tar ${IME\_BACKUP}

rm -rf ${IME\_BACKUP}

exit 0

else

echo "Direktorijum već postoji" >&2

exit 1

fi

*Slika 105 : Rezultat skripte sa for loop*



Naravno, ova skripta je mogla da se izvede i samo jednostavnim *mv \*.sh* u neki *backup* fajl i zatim da se spakuje. Ideja je bila da se prikaže kako bi neki realni zadatak bio izveden upotrebom petlji.

## 11.3 Regularni izrazi

Često u radu sa programskim jezicima ili *Linux* sistemima nailazimo na pojam *regularni izrazi*. Formalna definicija za njih bila bi sekvenca karaktera koji se intepretiraju kao pretraga ili obrada koja treba da se izvrši na fajlu.

Ovi izrazi imaju značajnu ulogu i prilikom pisanja skripti koje vrše neku automatizaciju, jer ne može baš sve da se izvrši upotrebom jedne komande, već, ponekad, moramo da podešavamo izlaz.

U narednoj tablici predstavljeni su neki meta karakteri i kako oni utiču na izlaz *grep* komande:

*Tabela 1: Specijalni karakteri regex*

|  |  |  |
| --- | --- | --- |
| BR | Komanda upotrebe | Rezultat |
| 1. | **grep user fajl.txt** | *Grep* će pretražiti pojam *user* u fajlu *fajl.txt* i daće nam sve pronađene linije koje sadrže ovaj pojam |
| 2. | **grep user \*.txt** | *Grep* će tražiti pojam *user* u fajlovima koji se završavaju sa *.txt*  Često se ovaj meta karakter naziva *wildcard* |
| 3. | **grep '^A' fajl.txt** | *Grep* će tražiti sve linije koje počinju slovom A |
| 4. | **grep '2$' fajl.txt** | *Grep* će prikazati sve linje koje se završavaju brojem 2 |
| 5. | **grep '1..' fajl.txt** | *Grep* će prikazati linije koje sadrže 1 i bilo koja 2 karaktera iza njega uračunavajući *space* koji se tretira kao karakter |
| 6. | **grep '.2' fajl.txt** | *Grep* će prikazati linje koje koje sadrže 2 i bilo koji karakter ispred njega uračunavajući *space* |
| 7. | **grep '^[AI]' fajl.txt** | *Grep* će prikazati sve linije koje počinju sa A ili I (velikim slovima samo) |
| 8. | **grep '[^0-9]' fajl.txt** | *Grep* traži linije koje sadrže najmanje jedan alfanumerički karakter |
| 9. | **grep '[A-Z][A-Z] [A-Z]' fajl.txt** | *Grep* traži reč koja sadrži veliko slovo, pa opet veliko slovo, razmak i ponovo veliko slovo |
| 10. | **grep '[a-z]{8}' fajl.txt** | *Grep* traži reč koja sadrži malo slovo ali kao 8 uzastopnih malih slova |

## 11.4 Sed

*Sed* ili neinteraktivni *stream edito*r popularna je alatka kod sistemskih administratora, zato izdvajamo neke komandne osnove.

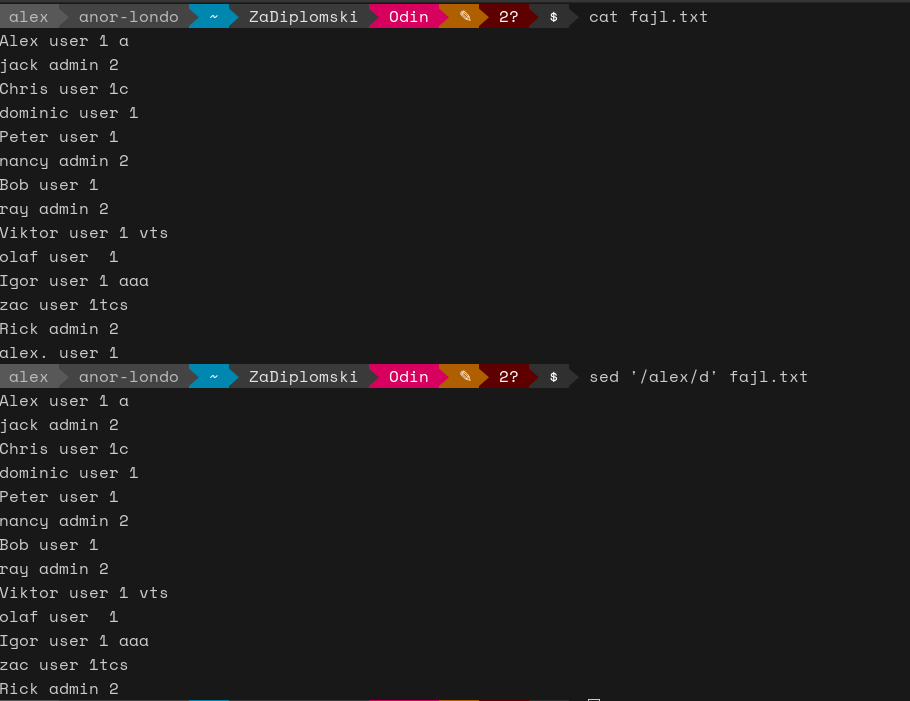
Osnovna upotreba jeste manipulacija tekstom iz komande linije ili *shell*, tj. upravljamo *streamovima*. *Sed* sve promene koje unesemo prikazuje na ekranu gde je pokrenut. Čuvanje novog modifikovanog fajla može se obaviti i usmeravanjem u novi fajl ili prepravkom već postojećeg fajla.

Pošto *sed* ima mnoštvo komandi koje mogu da se upotrebljavaju prilikom rada sa njim, izdvajamo ovde neke osnovne koje se vrlo često koriste.

Recimo, u fajlu *fajl.txt* nalaze se imena korisnika i administrator. Za sada želimo samo da obrišemo pojave imena *alex*, upotrebom komande:

$ sed ‘/alex/d’ fajl.txt

*Slika 106: Brisanje imena iz fajla na standardnom izlazu upotrebom sed komande*



Može da se primeti da prethodno postoji korisnik *alex*. *User 1* kada smo ga prikazali komandom *cat*, nakon upotrebe *sed* ova linija više ne postoji.

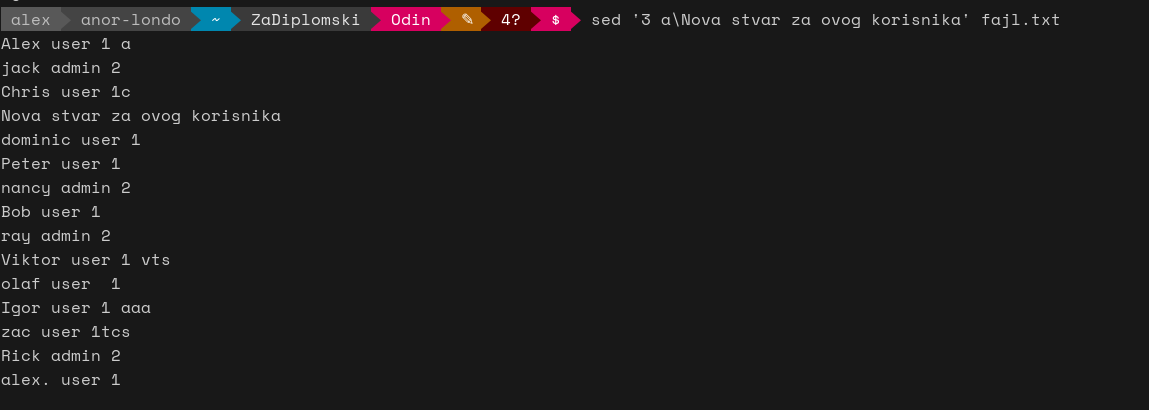
Naravno, fajl je ostao neizmenjen jer nismo *sed* komandi rekli da to uradi. U suštini, oblikovali smo standardni izlaz po našoj želji.

Još jedna česta upotreba jeste da dodajemo nešto u fajl, ovo se može izvršiti upotrebom *append* oznake u *sed* komandi.

Koristimo sledeću komandu za unos u fajl tj. standardni izlaz:

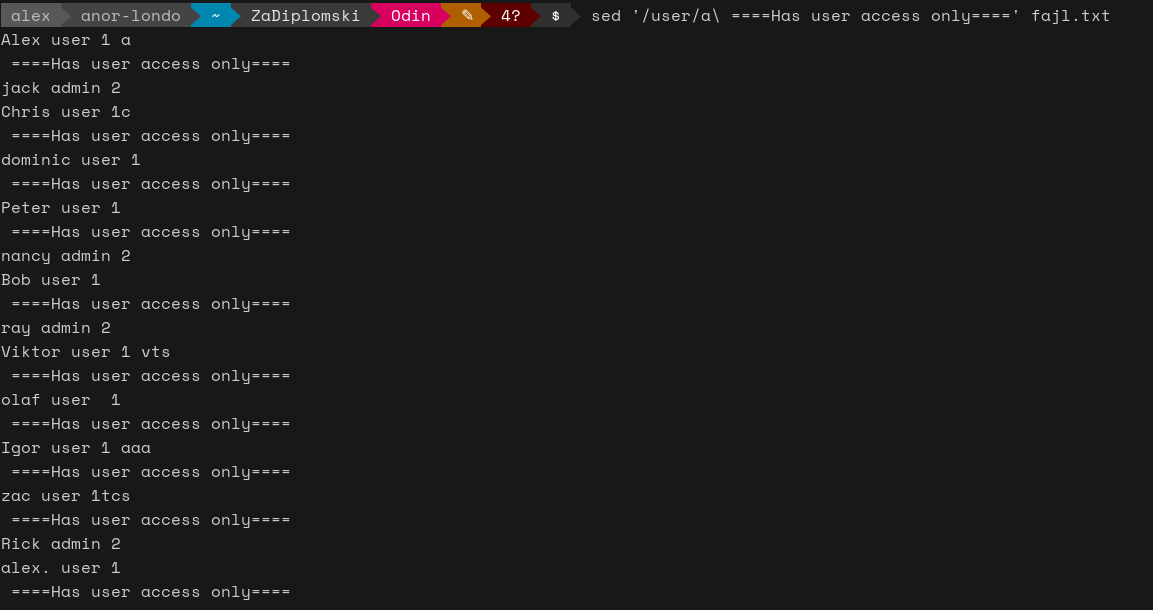
$ sed ‘3 a\ komentar za korisnika’ fajl.txt

*Slika 107: Upotreba append u sed*



U ovom primeru rekli smo *sed* da na trećoj liniji doda liniju sa tekstom *Nova stvar za ovog korisnika*, isto tako možemo staviti za neku reč koju tražimo.

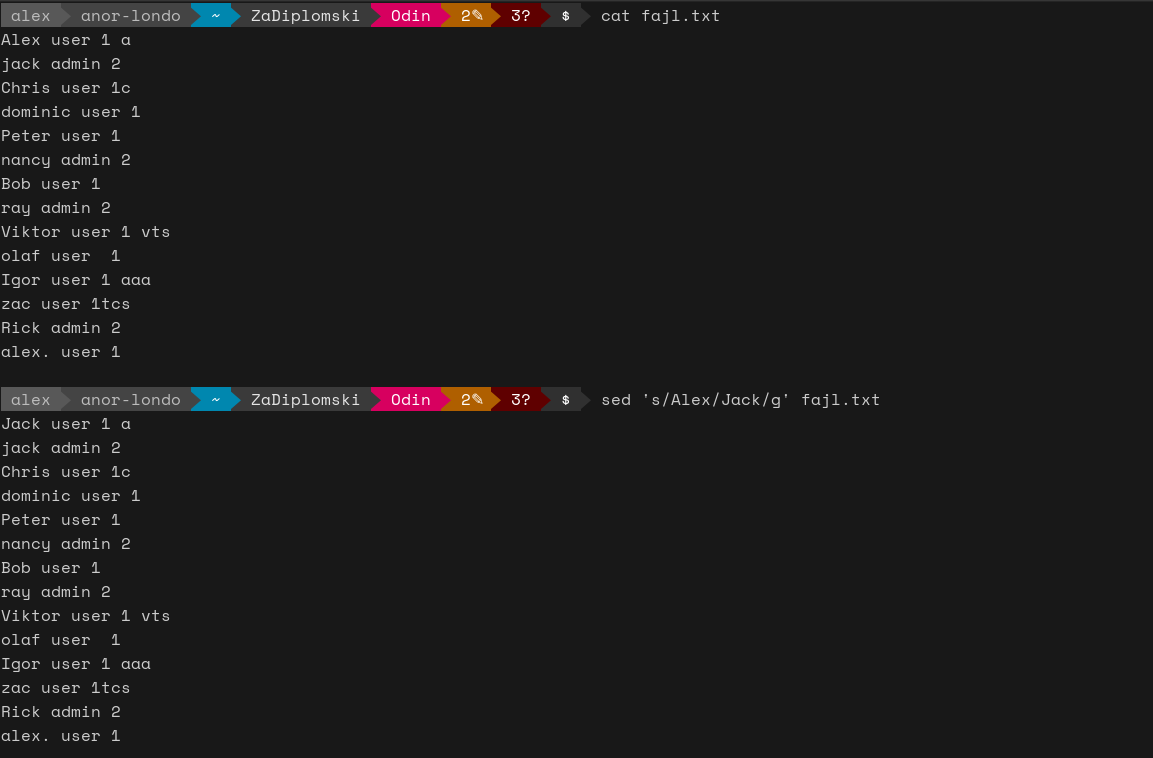
*Slika 108: Upotreba append parametra u sed sa pretragom reči*



Upotrebom komande *sed* sa pretragom ključne reči komanda je dodala *“====Has user access only====”* ispod svake linije koja sadrži *user* kao ključ pretrage. Dodatno na ovo postoji i *insert* koji samo dodaje na specifičnu liniju njegov *flag* u *sed* je *i*. *Sed* može da radi i sa meta karakterima kao i *grep* koji je pomenut u prethodnom odeljku *Regularni izrazi*. Sistemski administratori često upotrebljavaju *sed* kada žele da zamene neku liniju drugačijim unosom:

$ sed ‘s/Alex/Jack/g’ fajl.txt

*Slika 109: Upotreba zamene u sed*



*Alex* korisnik u listi zamenjen je sa *Jack* kada smo specificirali u *sed*. Naravno, kao i sve do sada ovo je samo na standardnom izlazu, *fajl.txt* ostaje nepromenjen. Ovo su samo neke od mogućnosti *sed*. S obzirom da ova alatka može da se koristi i za skriptovanje, njene mogućnosti su mnogo veće nego ovde navedene, koje samo predstavljaju mali deo onoga što može.

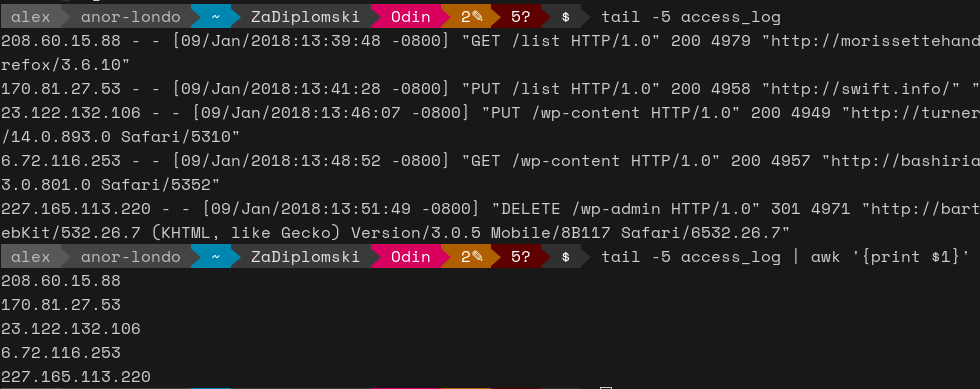
## 11.5 AWK

*AWK* je sopstveni programski jezik koji može da se izvršava u terminalu ili u skripti, prevalentno se koristi za obradu podataka i generisanje izveštaja.

Pošto je *AWK* programski jezik dotaći ćemo se samo nekih upotreba jer osnovna upotreba ne zahteva neko veliko programersko znanje.

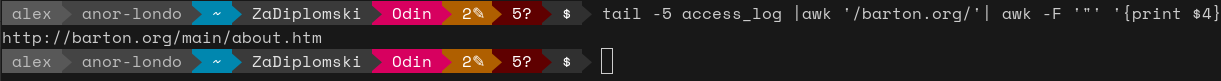
Na primer, želimo samo da ištampamo polje sa *IP* adresama u nekom *access* logu (log za beleženje posete i pristupa stranicama):

*Slika 110: Upotreba AWK komande*



Sa slike možemo uočiti da smo odštampali samo prvo polje ili *$1* koje smo zadali upotrebom *print* komande unutar *AWK*.

*Slika 111: Pretraga ključne reči sa AWK*



Prilikom pretrage ključne reči koja je jedna od mogućnosti *AWK* (Slika 111), koristili smo **|** (pipe) kako bismo preusmerili izlaz i modifikovali kako će izgledati na standardnom izlazu. Dobili smo samo stranicu koju smo tražili iz fajla. U ovome leži moć *AWK* komande, koja može da uradi i mnogo kompleksnije stvari, ali ovo bi bilo dovoljno znanja za nekog ko želi samo da prepravi izlaz komande kako njemu odgovara.

# 12. Firewall i nadgledanje sistema

## 12.1 Uvod

Česta stvar na koju nailazimo na nekom aktivnom serveru je *firewall* ili zaštitna barijera koja reguliše ulazni i izlazni saobraćaj na našem serveru. Od *firewall* implementacija imamo *firewallcmd* i *ufw* (*Debian*), *csf* (*CentOS*), *iptables*… i mnoge druge.

Na *Linux* operativnim sistemima kao što je *CentOS*, *iptables* je jedna od implementacija koja dozvoljava sistemskom administratoru da podesi pravila koja će vršiti filtriranje saobraćaja. Svaki set pravila podešen je u nekoliko tabela koje definišu kako se pravila izvršavaju i kako se upravlja saobraćajem koji dolazi na server. Osnovna upotreba nad već podešenim tabelama jeste blokiranje nepoželjnog saobraćaja, zatvaranje portova ka spoljašnjem svetu ili dozvola određenim *IP* adresama da pristupe ovim portovima, takođe pomaže prilikom *DOS* (Denial of Service) napada.

## 12.2 Filtriranje paketa

Na rudamentarnom nivou *firewall* koji filtrira pakete je ruter. Njegov zadatak unutar mreže jeste da proveri pakete koji se šalju ili primaju i prosledi ih na odgovarajući način. Bitno je izdvojiti da se paket sastoji iz dva dela:

* podatak
* zaglavlje

Zaglavlja paketa sadrže *IP* adresu mašine koja šalje podatke i mašine koja prima podatke, veličinu paketa i druge značajne informacije za komunikaciju. Samim tim što svaki paket sadrži ove informacije *firewall* ne mora da pregleda podatke paketa, već samo na osnovu zaglavlja zna šta treba da radi sa paketom. Kao što je prethodno navedeno, *firewall* koristi informacije iz zaglavlja paketa kako bi saznao šta treba da radi sa paketom. Pri tome paketi moraju da poštuju određena pravila kako bi bili pušteni kroz *firewall*. Na *Linux* serverima pravila se konfigurišu upotrebom *IP* lanca (IP chains) ili *IP* tabela.

## 12.3 IP lanci u firewall konfiguracijama

*IP* lanci su skup pravila koja se obrađuju kada paket pokušava da prođe *firewall*. Početna konfiguracija sadrži tri stalna lanca:

* *INPUT* (ulaz)
* *OUTPUT* (izlaz)
* *FORWARD* (prosleđivanje)

*Input* lanac, kao i što ime kaže, reguliše dolazne pakete, *output* lanac reguliše izlazne pakete, a lanac *forward* upravlja paketima koji se rutiraju (prosleđuju) kroz sistem.

### 12.3.1 Upravljanje pravilima IP lanca

Pravila *IP* lanca mogu da navode:

* -s izvor paketa ili *source*
* -p protokol
* -d odredište ili *destination*
* -j gde treba novo kreirano pravilo poslati
* *Port*

Prilikom rada sa *IP* lancima može se takođe primetiti da postoje dva tipa pravila, ona koja utiču na ceo lanac i ona koja utiču na pravila u okviru lanca.

Opcije koje utiču na ceo lanac:

* -N kreira novi lanac
* -X briše prazan lanac
* -P menja ponašanje ugrađenog lanca
* -L prikazuje pravila u lancu
* -F briše pravila lanca
* -Z podešava brojače bajtova i paketa na 0, na svim pravilima u okviru lanca

Opcije koje utiču samo na pravila u okviru lanca:

* -A dodaje novo pravilo u lancu
* -I ubacuje novo pravilo na neku poziciju u lancu
* -R menja pravilo na nekoj poziciji
* -D briše pravila na osnovu zadatog kriterijuma ili pozicije.

Na primer, želimo da blokiramo neku *IP* adresu ka našem serveru:

ipchains -A input -s 127.0.0.1 -j DENY

Pravilo se dodaje u lanac sa -A i dodaje se u ulazni stalni lanac (input), -s izvor koji se dodaje je *IP* adresa koja je specificirana i poslednji deo nam govori da se pravilo šalje ka *deny* tj. da se odbijaju bilo kakvi ulazni paketi od 127.0.0.1

Isto ovo može da se uradi i za neki specifični servis na serveru, npr. za često korišćeni servis na serverima *FTP* (File transfer protocol).

ipchains -A input -p tcp -i eth0 --dport ftp -j REJECT

Iz komande iznad može se primetiti par sličnosti: prvo što je odrađeno jeste da se pravilo dodaje u *input* stalni lanac upotrebom *-A*. Zatim, upotrebom protokola pravilo se podešava na *tcp* ( tcp ima specifične portove koje dozvoljava i koristi se kao protokol u ovom primeru); *-i* se odnosi na interfejs ili uređaj, u ovom slučaju to je *eth0* (veza ka spoljnom svetu); *–dport* govori koji servis se blokira ovde je *ftp* i *–j*, kao i u prethodnom primeru, govori šta pravilo treba da uradi, ovde se odbija bilo kakav zahtev ka *ftp* servisu iz spoljnog sveta.

Sva ova podešavanja mogu se obaviti unutar konfiguracionog fajla koji se nalazi na *CentOS* sistemima */etc/ipchains.conf*, ili upotrebom *ipchains-save* mogu se preusmeriti bilo koja pravila koja su napisana unutar terminala u stalni fajl, a to je *ipchains.conf*.

ipchains-save>/etc/ipchains.conf

## 12.4 IP tabele u firewall konfiguracijama

*IP* tabele su malo poboljšana verzija već postojeće implementacije. Za razliku od *IP* lanaca koji nema neki definisan redosled, u *IP* tabelama sve je podešeno da se dešava po nekom redu, tj. *IP* tabele su implementacija sa stanjima gde se prate otvorene konekcije.

IP tabele imaju tri podrazumevane tabele: *nat, filter* i *mangle*.

* *NAT* tabele koriste se kako bi se usmeravala konekcija ka drugim mrežnim interfejsima.
* *Filter* tabele obavljaju zadatak filtriranja paketa, i upravalja kretanjem paketa na mreži. Takođe dozvoljava i blokira konekcije u zavisnosti od svoje postavke.
* *Mangle* tabele koriste se za menjanje zaglavlja paketa.

Za razliku od *IP* lanaca, tabele pored klasičnog ulaza (input), izlaza (output) i prosleđivanja (forwarding) imaju *POSTROUTING* i *PREROUTING*.

*POSTROUTING* i *PREROUTING* dodati su od strane *nat* tabele. *PREROUTING* dozvoljava menjaje paketa pre nego što dođu na ulazni (input) lanac, dok *POSTROUTING* dozvoljava menjanje paketa nakon što izađu iz izlaznog lanca (output).

### 12.4.1 Upravljanje pravilima IP tabela

Pošto su *IP* tabele proširenje već postojećih *IP* lanaca, neka pravila su slična dok *nat* tabele imaju svoja proširenja na već postojeću semantiku.

Pravila koja mogu da se navode su sledeća:

* -s kao i kod IP lanaca implicira na izvor ili *source*
* -p stoji za protokol pravila kojeg podešavamo
* -d je destinacija
* -j i poslednje kao i kod *IP* lanaca kada su ispoštovana sva ostala pravila usmerava se na odgovarajući lanac.
* *Port*

Kao što je prethodno rečeno, *IP* tabele su proširenje već postojeće implementacija s tim što postoje pravila koje utiču na celu tabelu, ceo lanac i pravila u okviru lanca. Tabela koja je uvek podrazumevana, sem ako nije specifično navedeno je *filter tabela*.

Opcije koje utiču na celu tabelu:

* -t ova opcija omogućava specificiranje tabele nad kojom se pravilo izvršava

Opcije koje utiču na čitav lanac:

* -N pravi novi lanac
* -X briše prazan lanac
* -E menja ime lanca
* -P menja ponašanje već ugrađenog lanca
* -L prikazuje pravila već podešenog lanca
* -F briše pravila iz lanca
* -Z postavlja brojače paketa i bajtova na 0 na svim pravilima u okviru lanca.

Opcije koje utiču samo u okviru lanca:

* -A dodavanje pravila u lanac
* -I unos pravila na poziciju unutar lanca
* -R zamena već postojećeg pravila na poziciji u lancu
* -D brisanje pravila na nekoj poziciji u lancu ili na osnovu nekog već postojećeg kriterijuma

Primer upotrebe pravila unutar *iptables*:

iptables -A INPUT -s 127.0.0.1 -j DROP

Ovo pravilo ispušta sve pakete koji dolaze sa IP adrese 127.0.0.1

Recimo da imamo sličnu situaciju kao i sa *IP* lancima, gde hoćemo da sprečimo *FTP* servis da komunicira sa spoljašnjim svetom. Pisanje tog pravila može da se izvede na sledeći način:

iptables -A INPUT -p tcp -i eth0 --source-port ftp -j REJECT

*IP* tabele imaju svoju konfiguraciju, kao i lanci, i nalaze se unutar direktorijuma *etc/iptables.conf*. Pravila se čuvaju upotrebom komande *iptables-save* tako što se usmeravaju u konfiguracioni fajl.

iptables-save > /etc/iptables.conf

Takođe, jedna od opcija *iptables* je povratak fabričkih podešavanja za tabele, a to je upotrebom komande *iptables-restore* preusmeravanjem na konfiguracioni fajl.

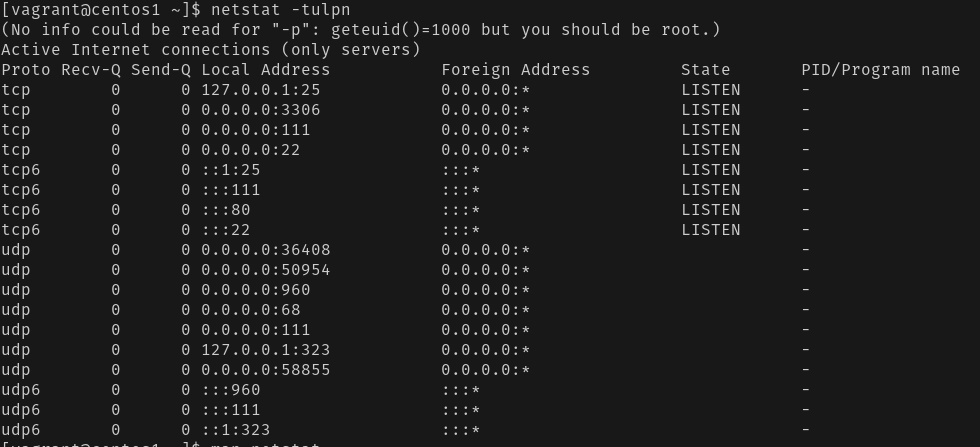
iptables-restore</etc/iptables.conf

Prilikom prvobitnog podešavanja bilo kakvih *firewall* pravila dobro je proći kroz dokumentaciju i uzeti inicijalne skripte koje pomažu pri formiranju osnovne zaštite. *Red Hat* nudi mnoštvo *how to* koji pomažu administratorima pri formiranju ovih pravila i kriterijuma.

## 12.5 Nadgledanje sistema

U radu sa serverom, veliki deo posla svodi se na nadgledanje nekih servisa koji možda ne rade ili želimo da utvrdimo njihovu funkcionalnost da rade ono što im je zadatak. Postoji mnoštvo komandi koje su nam dostupne za neke određene zadatke, recimo želimo da vidimo koji portovi su aktivni i koji servisi na njima slušaju. Možemo koristiti sledeću komandu kada želimo da vidimo koji port je aktivan za neki servis *‘$ netstat -tulpn’* (iako nije fabrički na sistemima, alternativa bi bila *ss*, takođe i *netstat* je deo *net-tools* paketa)

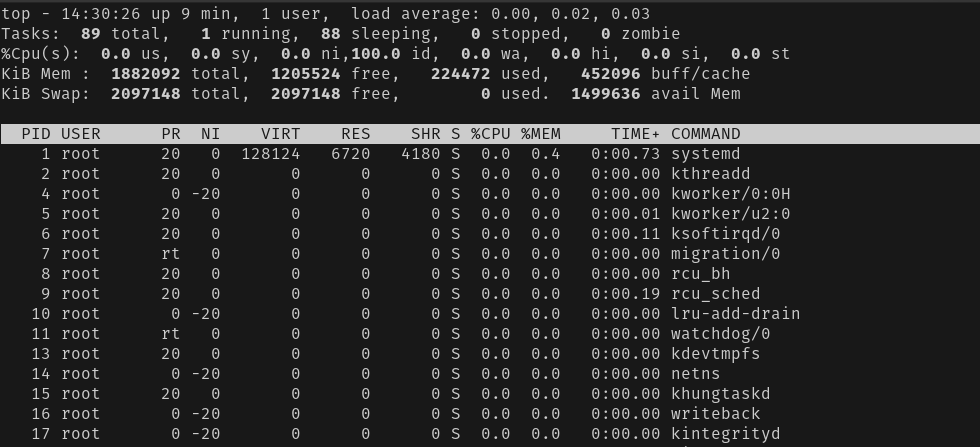
*Slika 112: Upotreba netstat komande*



Predstavljeni su aktivni portovi na serveru, uz malo objašnjenje šta specifično tražimo: Flagovi: -t (tcp) –u (udp) -l (listening tj. sluša ili it aktivan) –p (program da prikaže pid programa koji je aktivan, recimo ssh koji je zapisan ovako 696/sshd) –n (numerička vrednost).

Vrlo česta komanda bi bila *top* koja nam daje informacije o svim aktivnim servisima. Mogućnosti ove komande dolaze do izražaja kada pokušavamo da rešimo neki problem na serveru, na primer da li postoji servis koji pravi problem ili kad moramo da znamo kome pripada pokrenuta komanda bio to *root* ili neki drugi korisnik. Prilikom inicijalnog otvaranja može se koristi i za proveru opterećenja na serveru koji možemo videti pod *load average*.

*Slika 113: Upotreba top komande*



Komanda je izuzetno korisna ako neki servis pravi problem, pa se može videti otprilike kako se sistem ponaša. Često ćemo naići i na problem sa upotrebom disk prostora, koji je takođe nešto što može da utiče na generalnu funkcionalnost servera.

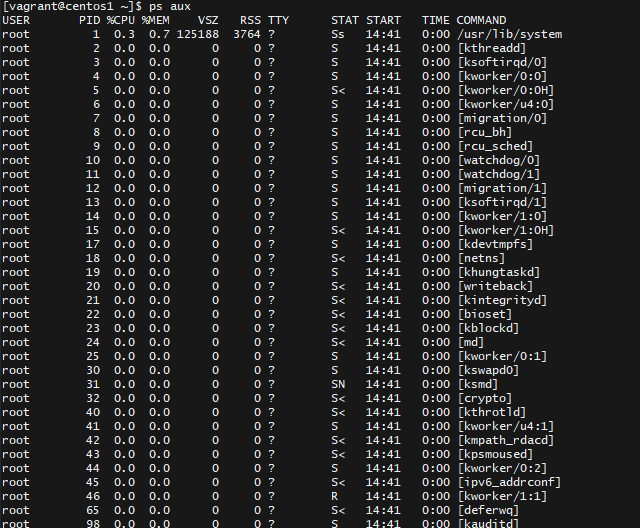
Još jedna prilično popularna komanda koja je informativnog karaktera kao i *top* je *ps.* Ova komanda prikazuje sve aktivne procese, ali u statičnom obliku u odnosu na top koja dinamički proverava statuse procesa.

*Slika 115: Upotreba komande ps*

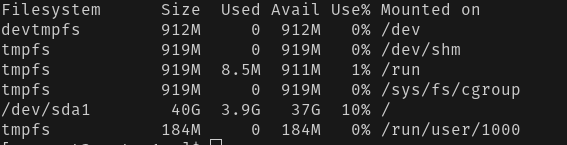


Ova komanda može da se proširi upotrebom flagova *aux* koji dodaju na sadržaj izlaza:

*Slika 116: Proširena komanda ps*



*Slika 117: Prikaz rezultata df komande*



Sledeća dobra komanda koja nam može dati neki generalni pogled na stanje je *df*. Može se primetiti da su izlistani svi mogući direktorijumi i njihovo zauzeće. Generalno, ovo nam je dobar indikator gde da počnemo, na realnim serverima, sem */root* direktorijuma, imamo i *tmp* i *var* direktorijum koji su izdvojeni zasebno kako bi se olakšalo upravljanje njima.

Sistemski administrator raspolaže mnogo većom paletom komandi nego ovde navedeno, bilo to komande koje dolaze sa operativnim sistemom ili aplikacije koje su naknadno postavljene. Ovde je pređen samo mali deo mogućnosti Linux operativnog sistema, i kako mogu da se pregledaju neki specifični problemi.

# Zaključak

*Linux* je veoma prevalentan operativni sistem, barem u serverskom svetu, a ima i primene u korisničkom svetu kroz mnoštvo distribucija i opcija koje su dostupne korisnicima koji žele da koriste ovaj sistem kao svoj radni sistem.

Smatram da je nephodno bilo za programera, ili nekog budućeg sistemskog administratora, da bude upoznat sa ovim sistemom i nekim njegovim funkcionalnostima. Moj cilj u ovom radu bio je da prikažem osnovno podešavanje koje bi možda neko prvi put radio na nekom serveru, a i da predstavim česte teme sa kojima se možemo susresti na ovom polju.

Nadam se da će u budućnosti ovaj rad biti koristan kao referenca za neki značajniji projekat u vezi sa *Linux* sistemima i polazna tačka u shvatanju suštine i ideje ovog operativnog sistema.

# Literatura

**[1] Knjige**

**- Dean, Adam K. 2018.** *Administriranje Linux sistema.* s.l. : Packt Publishing, 2018.

**- Naik, Ganesh. 2018.** *Linux Shell Skriptovanje.* s.l. : Packt Publishing, 2018. T. II.

**- Popović, Ranko, Branović, Irina i Šarac, Marko. 2011.** *Operativni sistemi.* Beograd : Univerzitet Singidunum, 2011.

**- Veinović, Mladen i Jevremović, Aleksandar. 2020.** *Računarske Mreže, sedmo izdanje.* Beograd : Univerzitet Singidunum, 2020.

**[2] Internet i elektronski izvori**

**- Wallen, Jack. 2022.** Classic SysAdmin,Understanding Linux File Permissions. [Na mreži] 6 Januar 2022. [Posećeno dana: 9.11.2022.] https://www.linuxfoundation.org/blog/blog/classic-sysadmin-understanding-linux-file-permissions.

**- Wordpress. 2022.** *How to install wordpress.* [Na mreži] 2022. [Posećeno dana: 12.10 2022.] https://wordpress.org/support/article/how-to-install-wordpress/.

**- Red Hat. 2022.** CentOS documentation. [Na mreži] 2022. [Posećeno dana: 10.10.2022.] https://docs.centos.org/en-US/docs/.

**- Cobbaut, Paul. 2015.** Linux Networking. [Na mreži] 2015. [Posećeno dana: 15.11.2022.] https://linux-training.be/networking/index.html.