Projektovanje elektronskih sistema

Predavanje 7

Linux OS

Doc.dr Borisav Jovanović

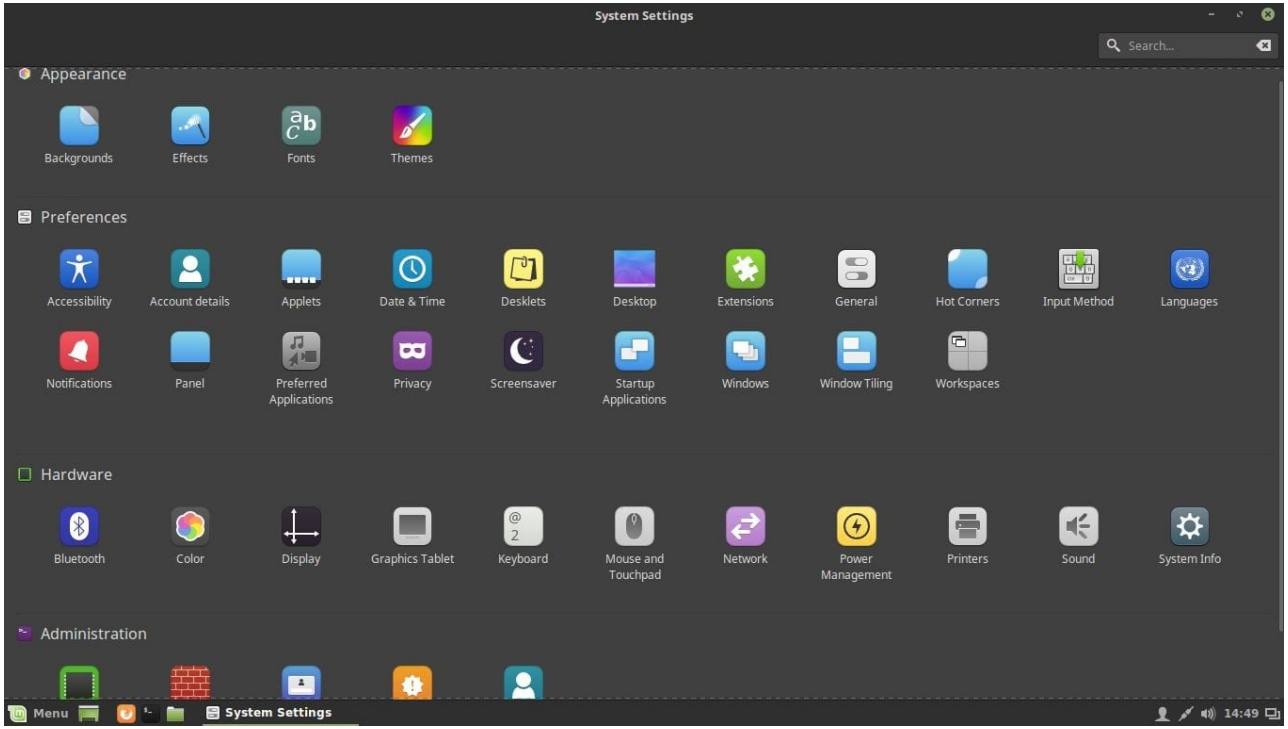
preuzeto iz predavanja prof. Milunke Damnjanovic i prof. Miluna Jevtica

Sadržaj:

- Prednosti Linux OS
- Arhitektura Linux OS
- Fajl sistem
- Zaštita fajlova
- Procesi
- Komande Linux OS
  - Rad sa direktorijumima
  - Tekst editori
  - Rad sa fajlovima
  - Promena pristupnih dozvola
  - Pretraživanje fajlova
  - Arhiviranje podataka
  - Administriranje procesa
  - Kreiranje izvršnih fajlova

Prednosti Linux OS

- Linux je vrlo popularan OS pisan na jeziku C.
- Koristi se kako u laboratorijama, univerzitetima, u industriji.
- Raspoložive su verzije za upotrebu na računarima opšte namene, na specijalizovanim računarima, na personalnim računarima, na grafičkim radnim stanicama, serverima.
- Linux je besplatan. Instalacija dolazi bez troškova.
- Pored samog OS, aplikacije su besplatne i *open source*.



- Windows OS je jedan od OS sa najjednostavnijim grafičkim okruženjem.
- Iako se rad na Windows lako uči, u tom pogledu ne zaostaju ni Linux distribucije poput Ubuntu-a, Elementary OS-a, Linux-a Mint,...

- Linux je pouzdaniji OS u poređenju sa Windows-om.
- Zajednica Linux programera je aktivna tako da periodično izlaze glavna ili manja ažuriranja OS.
- Linux sistemi poznati su po tome što mogu godinama da rade bez ijednog kvara ili čak bez ponovnog restartovanja. Ovo je važno posebno za serverske sisteme.

Linux sistemi troše manje sistemskih resursa (RAM-a, prostor na disku itd.) u poređenju sa Windows-om.

Proizvođači hardvera shvatili popularnost Linux-a i proizvode hardver i upravljačke programe (drajvere) koji su su kompatibilni sa Linux-om.

- Windows ima na raspolaganju veliki broj komercijalnih softvera.
- Linux, s druge strane, koristi **open-source** softver koji je besplatan.
- Linux poseduje jednostavne menadžere paketa koji pomažu u instaliranju i deinstaliranju željenih softverskih aplikacija.
- Za programere, Linux terminal nudi superiorno okruženje u poređenju sa Windows-om. GNU kompajleri i uslužni programi su korisni alati za programiranje.
- Često se dešava da Windows iznenada pokaže poruku da računar treba ponovo pokrenuti.
- Na Linux OS, kada se ažurira softver ili instalira/deinstalira softver, tzv. reboot računara nije potreban.

- Microsoft Windows je ranjiv na malvere, trojanske programe i viruse.
- Linux je skoro neranjiv i svakako sigurniji OS. Zato, Linux ne zahteva upotrebu komercijalnih antivirusnih/ anti-malvare paketa.
- Za razliku od Windows-a, on ne generiše log fajlove i ne prenosi (uploaduje) podatke sa PC-a.
- Korisnik treba da bude dobro upoznat sa Windows politikom privatnosti.

- Linux je prisutan svuda, od najmanjeg uređaja do najvećeg superračunara. To može biti automobil, ruter, telefon, medicinski uređaji, avion, TV, satelit, sat ili tablet uređaj.
- Linux podržava programske jezike Python, C / C ++, Java, Perl, Rubi, itd.
- Nudi širok spektar aplikacija korisnih u svrhu programiranja.
- Linux terminal je pogodniji za upotrebu u odnosu na Windows-ovu komandnu liniju.
- Biblioteke razvijene za Linux, kao i menadžeri paketa pomažu programerima da posao programiranja obavljaju brže i lakše.
- Mogućnost pisanja skripti također jedan od razloga zašto se programeri odlucuju za Linux OS.

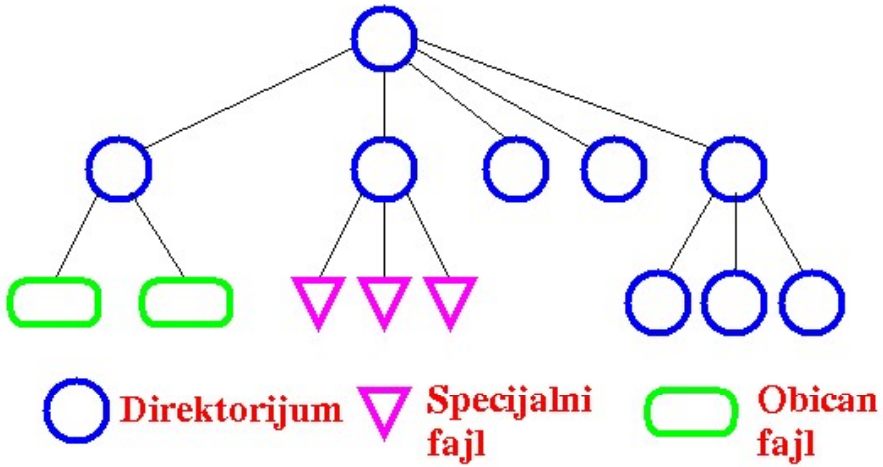
Arhitektura Linux OS

- Oko hardvera se nalazi niz slojeva koji izoluju korisnika od mašine.
- Prvi sloj oko hardvera je *jezgro (kernel)* koje ima zadatak da više slojeve izoluje od sloja ispod, to jest da ih učini hardverski nezavisnim.
- Sledeći sloj čini komandni interpreter *(shell)*. Zadatak ovog sloja jeste **interakcija sa korisnikom,** prihvatanje korisničkih zahteva, njihovo interpretiranje i predavanje nižem sloju (kernelu) na izvrsavanje.
- U najvišem sloju nalaze se različite komande i korisnički programi čiji je zadatak direktno pružanje usluga korisniku.

- Četiri njegove osnovne komponente: *kernel, shell, komande i fajl sistem*
- Kernel Linux operativnog sistema kontroliše pristup računaru, upravlja fajl sistemom, resursima računara, diskovima, stampačima, komunikacionim linijama i drugim uređajima.
- Koristi hijerarhijski fajl sistem sto omogućava lako održavanje i efikasnu implementaciju.
- Koristi konzistentan format za fajlove, posmatra ih kao niz bajtova sto olakšava pisanje aplikacionih programa

Fajl sistem

- *File* u Linux OS predstavlja kolekciju znakova.
- Veličina fajla je jednaka ukupnom broju znakova koje fajl sadrži.
- Fajl sistem je organizovan hijerarhijski u obliku stabla.
- U osnovi stabla je direktorijum poznat kao ***root***. Grane stabla predstavljaju pod-direktorijume dok fajlovi predstavljaju listove stabla fajl sistema.



Postoje tri vrste Linux fajlova:

- obični fajlovi,
- direktorijumi i
- fajlovi specijalnih jedinica.

- Obični fajlovi predstavljaju skup 8-bitnih znakova.
- Običan fajl moze sadržati izvorni program, izvršni program, tekst nekog dokumenta, slogove baze podataka, ili bilo koji drugi tip podataka.
- Bajtovi običnog fajla se mogu interpretirati kao znakovi nekog teksta, binarne instrukcije ili kao naredbe programa.
- Svaki običan fajl ima sledeće atribute:
  - ime fajla,
  - jedinstven broj fajla nazvan *inode*,
  - veličinu u bajtovima,
  - vreme kreiranja,
  - vreme poslednje izmene,
  - vreme poslednjeg pristupa,
  - pristupne dozvole i
  - vlasnika i grupu.

Postoje dva tipa običnih fajlova:

- tekstualni i
- binarni fajlovi.

Tekstualni fajlovi sadrze samo vidljive ASCII znakove.

Bajt binarnog fajla može imati bilo koju vrednost (od 0 do 255).

Zaštita fajlova

Zaštita fajlova se koristi za:

- Sprečavanje neovlašćenog pristupa, korišćenja ili brisanja podataka,
- Onemogućavanje slučajnog ili nenamernog brisanja,
- Omogućavanje da više korisnika koriste podatke iz istog fajla ali da svaki od njih ima različita prava nad njim.
- Čuvanje sadržaja fajla u slučaju kvara ili pada sistema.

Prava koja jedan korisnik moze imati nad fajlom nazivamo ***pristupnim dozvolama (access permissions)*.**

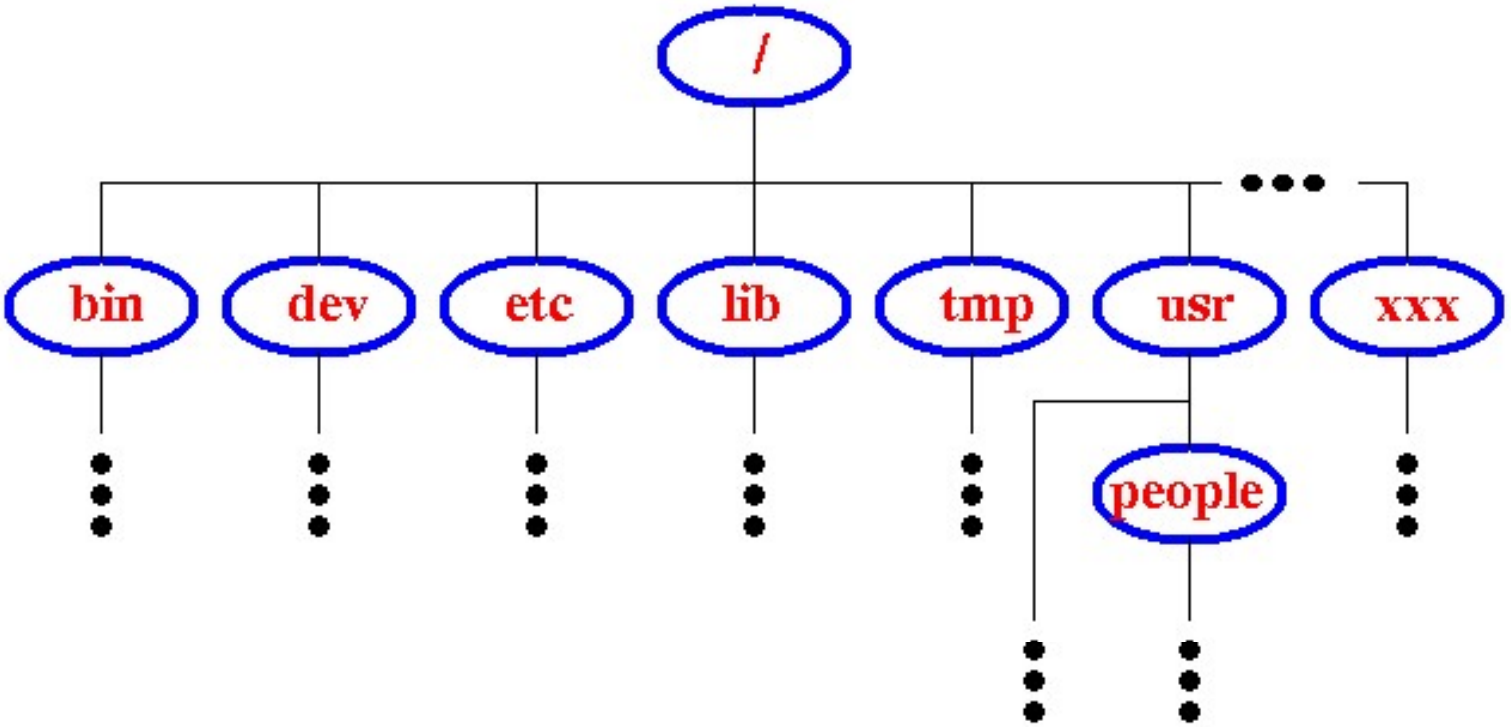
Za svaki fajl u LINUX OS je moguće kontrolisati sledeće tipove pristupnih dozvola:

- **Read** dozvoljeno čitanje fajla;
- **Write** dozvoljen upis u fajl
- **Execute** dozvoljeno izvršavanje fajla

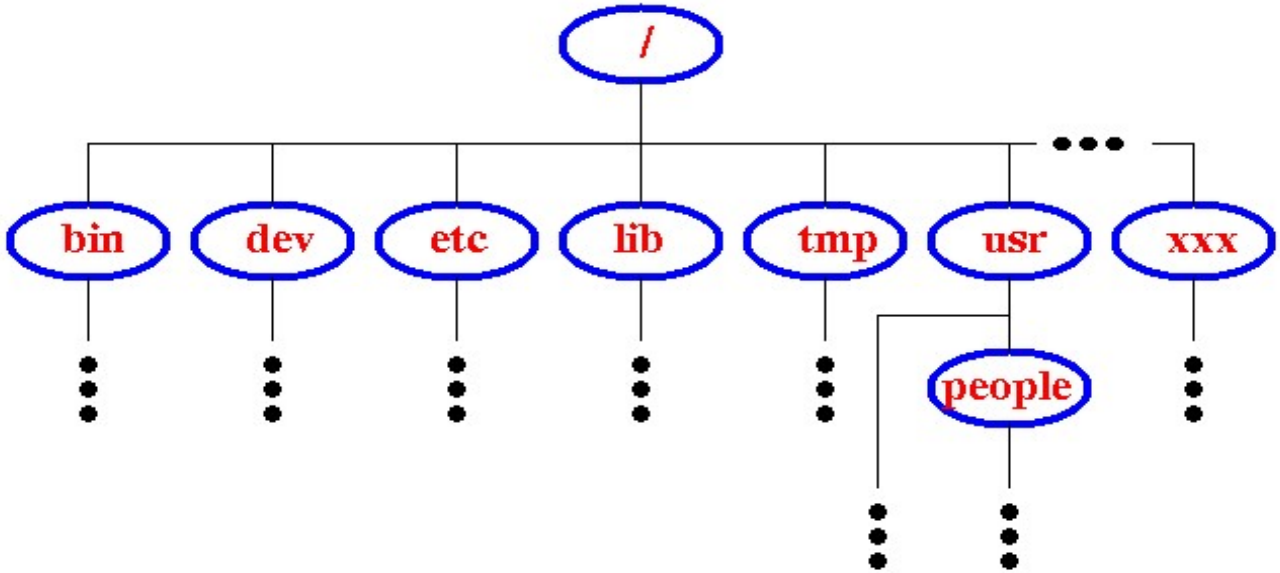
- Linux OS razlikuje dve vrste korisnika: root I obicne korsnike.
- Korisnik, poznat kao *root* ili *super korisnik (super user)* poseduje sva prava na sistemu. Njemu su dozvoljene sve akcije kao što su, na primer, promena konfiguracionih fajlova, instalacija novog i brisanje vec instaliranog softvera, pa čak i promena kernela. Privilegije ovakvog korisnika se dodeljuju pouzdanoj i stručnoj osobi koja se naziva ***sistem administratorom***.
- Svi ostali korisnici čine klasu ***običnih*** korisnika koji imaju ograničena prava.
- Svi korisnici Linux sistema, računajući i *super* korisnika, su podeljeni u grupe. Kriterijum koji se koristi za podelu korisnika je obično tip zadataka za koje su oni zaduženi.
- Na primer, skup korisnika koji rade na jednom zajedničkom projektu, čine grupu. Pri tome, jedan korisnik moze biti član jedne ili vise grupa (ako radi na većem broju projekata, na primer).

- Gledano sa strane fajla, svi korisnici OS se dele u tri klase:
  - vlasnik (*owner*),
  - članovi grupe (*group*) i
  - ostali (*other*).
- Vlasnik fajla je korisnik koji je kreirao fajl i on poseduje pravo da određuje pristupne dozvole fajlu. Ovo pravo poseduje i **super korisnik**. Članovi grupe jesu korisnici koji pripadaju grupi kojoj pripada i vlasnik fajla. Svi drugi korisnici se tretiraju kao *ostali*.
- Za svaku klasu korisnika fajla je moguće definisati zasebne pristupne dozvole. Na primer, uobičajeno je da vlasnik poseduje sva prava nad fajlom (čitanje, upis i izvršavanje, ako se radi o izvršnom fajlu), članovima grupe se dodeljuje pravo čitanja (bez prava menjanja sadržaja), dok ostali korisnici ne poseduju nikakvo pravo pristupa fajlu.
- Kontrola pristupa fajlu je relativno jednostavna i obavlja se sledećim komandama: chown, chgrp i chmod.

- Linux je posebno interesantan po tome sto se direktorijumi tretiraju na isti način kao i obični fajlovi s tim sto je u njih smeštena lista fajlova lociranih na tom direktorijumu.
- I ovde, kao i u drugim OS, direktorijumi omogucavaju hijerarhijsku organizaciju celog fajl sistema.
- Slično kao kod običnih fajlova, direktorijum ima tri tipa pristupnih dozvola: čitanje (*read*), upis (*write*) i izvršenje (*execute*) s tim sto se one ovde malo drugačije interpretiraju.
- Dozvola upisa i izvršenja odredjuje da li fajlovi mogu biti dodavani ili brisani sa direktorijuma. Dozvola izvršenja i čitanja određuje da li se sadrzaj direktorijuma moze listati.
- Takođe, gledano sa strane direktorijuma, postoje tri klase korisnika: vlasnik, clanovi grupe i ostali.
- Pristupne dozvole direktorijuma postavlja vlasnik, to jest korisnik koji je kreirao direktorijum.



- Prikazana je tipična struktura direktorijuma.
- Na vrhu stabla je direktorijum koji se naziva korenom (**root**) i označava se kosom crtom (/).
- Ispod root direktorijuma prikazani su direktorijumi koji su standardno prisutni u svim Linux varijantama.



- Fajl sistem skoro obavezno sadrzi direktorijume koji su prikazani na slici:
- **bin** - direktorijum sadrži najcesce korišćene Linux komande,
- **dev** - sadrži specijalne fajlove,
- **etc** - sadrzi najčesce korišćene komande i fajlove vezane za administraciju i održavanje sistema,
- **lib** - je direktorijum gde su smeštene biblioteke,
- **tmp** je direktorijum za čuvanje podataka privremenog karaktera,
- **usr** direktorijum sadrzi korisničke direktorijume (kod nekih realizacija se oni nalaze ispod **/usr/people** ili **/home** direktorijuma).
- Direktorijum **usr** takođe sadrži aplikativne programe, sistemske fajlove i direktorijume, ali koji su prvenstveno vezani za pružanje usluga korisnicima.

Pristupni put, je sekvenca imena direktorijuma na čijem se kraju nalazi ime ciljnog direktorijuma ili fajla.

Sva imena u putu su razdvojena kosom crtom. Ako put pocinje kosom crtom znaci da je formiran od korena, to jest **root** direktorijuma, i tada se naziva *potpunim* ili *apsolutnim* putem.

Put se takodje moze kreirati relativno u odnosu na radni (*working*), ili tekuci direktorijum i tada se on naziva *relativnim*.
UNIX operativni sistem nudi specijalne skracenice za parent i radni direktorijum:

Skraćeno ime za radni direktorijum. Na primer, ./xxx je ime fajla xxx u tekućem direktorijumu.

Skraćeno ime za ***parent*** direktorijum radnog direktorijuma.
Na primer, ../.. se odnosi na direktorijum koji je dva nivoa iznad radnog.

- Link je mehanizam kojim se omogućava da se više imena dodeli jednom fajlu.
- Postoje dve vrste linkova: ***hard* i simbolički (*soft*)** linkovi.
- Hard linkom se dva ili više imena vezuju za jedan isti fizički fajl, odnosno za jedan isti *inode*.
- > ln original linkname
- Simbolički link obezbedjuje isto to ali predstavlja fajl koji sadrzi put do drugog fajla u sistemu.
- > ln -s original linkname
- > cp -s original linkname

- Jedna od glavnih inovacija Linux sistema je koncept *pipe*-a.
- Pipe dozvoljava protok podataka u jednom pravcu izmedju dva procesa.
- Pipe predstavlja jedan od mehanizama UNIX-a za interprocesnu komunikaciju.
- Najčesće je to način da se izlaz jedne komande koristi kao ulaz u drugu, tako da se one izvrsavaju kao sekvenca nazvana *pipeline*.
- U ovoj sekvenci, komande se odvajaju uspravnom crtom, na primer:

- **Socket** je poseban tip fajla koji, slično kao i pipe, omogucava interprocesnu komunikaciju.
- U suštini, socket predstavlja generalizaciju pipe-a u smislu da omogucuje komunikaciju dva procesa koji ne moraju da se izvrsavaju na istoj mašini.
- Najčesci primer koriscenja socket-a je kod komunikacije dva procesa koji se izvršavaju na različitim sistemima a medjusobno komuniciraju preko računarske mreze.

Procesi

Proces je osnovna jedinica Linux-a koja se izvrsava.

Postoji vise tipova procesa:

- **interaktivni procesi** -procesi koji se zahtevaju i kontrolisu preko terminala. Mogu se izvrsavati u *foreground* ili *background* modu.
- **batch procesi** -procesi koji se izvrsavaju u batch obradi, tj. definisan je redosled procesa koji se izvrsavaju sekvencijalno.
- **daemons** -server procesi koji se izvrsavaju u background-u i cekaju dok neki drugi proces ne zahteva njihove usluge.

Svaki proces poseduje sledece atribute:

- **Process ID** -jedinstveni identifikacioni broj procesa
- **Nice number** -broj koji definise prioritet procesa u zauzimanju resursa racunara
- **TTY** -oznaka terminala sa koga je proces startovan

- Shell je komandni interpreter UNIX operativnog sistema i moze se posmatrati kao sloj koji se nalazi izmedju kernela i korisnika.
- alat koji izvršava korisničke komande
- Interaktivan je s obzirom da ima direktnu komunikaciju sa korisnikom.
- On prihvata korisničke zahteve, interpretira ih i predaje kernelu na izvrsavanje.
- Komande se unose u tekstualni terminal (prozor u grafičkom okruženju ili čisto tekstualna konzola)
- Rezultati se ispisuju u terminal (nema potrebe za grafikom)

Komande Linux OS

Programi koji se izvrsavaju odmah, bez kompajliranja, nazivaju se **izvrsnim programima ili komandama**.

Sve komande UNIX-a se mogu svrstati u neku od sledecih klasa:

- **obrada teksta** -tekst editori (ed, ex, vi), spell checker, tekst formateri i slicno,
- **upravljanje podacima** -kreiranje, organizacija i brisanje fajlova i direktorijuma,
- **elektronske komunikacije** - vise programa (write, mail) koji omogucavaju razmenu poruka medju korisnicima,
- **programska okolina** -mogucnost definisanja komforne programske okoline koriscenjem razlicitih usluznih programa,
- **razvojno okruzenje** -kompajleri i interpreteri raznih programskih jezika, linkeri, debageri,...
- **ostali usluzni programi** -graficka podrska, X Windows.

U Linux-u sve se tretira kao običan fajl:

- ‘normalni’ (tekstualni) fajlovi,
- Izvršni fajlovi (binarni fajlovi ili Shell skripte),
- direktoriumi,
- device files (fajlovi uređaja),
- pipes,
- Simbolički i hard linkovi (reference na fajlove)

Rad sa direktorijumima

Komanda pwd prikazuje sadržaj tekućeg direktorijuma.

Sintaksa: **pwd**

Komanda ls prikazuje, lista imena svih fajlova. Lista sadržaj nekog direktorijuma.

Sintaksa: **ls [-alR] [file/directory]**

Opcije:

-a prikazuje pored običnih takođe i skrivene fajlove/foldere koji počinju tačkom

-l dugi listing format. Prikazuje dozvole, korisnike i grupe, vremenske oznake, veličinu fajla, itd.

-R prikazuje fajlove u trenutnom direktorijumu i u svim pod-direktorijumima rekurzivno.

primeri:

```
% ls
```

proba.c novi.c Makefile

```
% ls -aF
```

.login .profile proba.c novi.c a.out* tmp/

```
% ls -l
```

-r--r--r-- 1 root sys 1145 Oct 23 18:41 READ.ME

-rw-rw-rw- 1 tanja user 85 Nov 7 14:47 boja.c

-rwxrwxrwx 1 tanja user 1046 Nov 15 13:01 create

- Pored imena, dugački listing za svaki fajl ili direktorijum daje i sledeće informacije: tip fajla, pristupne dozvole, broj hard linkova, vlasnika fajla, grupu kojoj fajl pripada, velicinu u bajtovima, datum i vreme kreiranja, i na kraju, ime fajla.

Komanda: mkdir – kreira novi direktorijum,

Komanda: rmdir - briše direktorijum (koji je već prazan).

Sintaksa:

mkdir directory

rmdir directory

Primeri

```
>  ls -la
>  mkdir program
>  mkdir razno
>  mkdir projektovanje
 > ls -a ./projektovanje
>  rmdir projektovanje
>  ls -la
>  rmdir razno
>  ls -la
```

Komanda: cd

Promena tekućeg (radnog) direktorijuma.

Komanda kojom se omogućava kretanje po stablu fajl sistema.

Sintaksa: **cd [directory]**

Primer: Unutar tekućeg direktorijuma program kreirajte nove direktorijume: src, object, bin. Idite u src, a zatim se vratite u glavni direktorijum

```
> pwd
> cd program
> pwd
> mkdir source
> mkdir object
> mkdir bin
> cd ./source
> pwd
> ls -a ./../../
> cd ./../..
> pwd
```

Tekst editori

Linux OS standardno prate nekoliko editora teksta.
Tekst editor **vi** je podržan od strane svih verzija Linux-a.

VI editor ima dva moda rada: Komandni i Insert mod.

Komandni mod VI editora koristi se za unos komandi. Svi znaci otkucani sa tastature se tretiraju kao komande. Možete pomerati kursor**,** izvršavati **cut, copy, paste.** U ovom se modu snimaju sve promene, zatvara fajl.

Insert mod se koristi za unos teksta, tasteri imaju normalno značenje.

Prelazak iz komandnog u insert mod se ostvaruje većim brojem komandi za dodavanje teksta.
Recimo, možete koristiti tastera 'i' na tastaturi. Napuštanje insert moda i povratak u komandni obavlja uvek pritiskom na <Esc> taster.

Tasteri u Command modu

- **i** – Insert kursor **(prelaz u insert mod)**
- **a** – Upisi posle kursora **(prelaz u insert mod)**
- **A** Pisi na kraju linije **(prelaz u insert mod)**
- **ESC** Završi *insert* mod

Kretanje po fajlu preko tastera **h, j, k, l**

Snimanje i prestanak rada

- :w snimi fajl koji ostaje otvoren
- :q zatvori bez snimanja
- :wq snimi i zatvori

primer: Kreirati fajl Hello.cpp file koristeći vi editor u folderu./program/source

> pwd

> cd ./program/source

> vi Hello.cpp

Pritisni **i** za prelaz u Insert mod

Napiši sledeći sadržaj:

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello, World!";
    return 0;
}
```

Pritisni **ESC** za prelazak u Command mod

Komanda **:wq** za snimanje i zatvaranje fajla

Komanda **cat –** prikazuje sadržaj tekstualnog fajla
Syntax: **cat** ***file***

Komanda **more** – omogućava pregledavanje većih fajlova strana po stranu.
Važne komande su **b** za skrolovanje i **q** za prekidanje - quit.
Sintaksa : **more** ***file***

primer: Prikaži/ kreiraj Hello.cpp, Test.txt files koristeći komande cat i more

```
>  cat Hello.cpp
>  cat > Test.txt
This is a test.
Ctrl+D
>  more Hello.cpp
>  more Test.txt
```

Rad sa fajlovima

Komanda **cp** se koristi za kopiranje fajlova i direktorijuma. Originalni fajl/direktorijum se ne menja. Opcijom **-r** direktorijumi se rekurzivno kopiraju (sa svim podfolderima).

Sintaksa:

```
cp file1 file2
cp file1 [file2 ...] directory
cp -r dir1 dir2
cp -r dir1 [dir2 ...] directory
```

primer: Kopirati Hello.cpp u novi Hello2.cpp u istom folderu

./program/source . Onda kopirati Hello2.cpp u folder ./program/bin

```
> pwd
> cp Hello.cpp Hello2.cpp
> ls ./.
> cp Hello2.cpp ./../bin/
> ls ./../bin
> cat ./../bin/Hello2.cpp
```

Komanda **mv** - rename promena imena ili premeštanje fajlova idirektorijuma. Slično kao kod **cp**, samo originalni fajl se briše.

Sintaksa:

```
mv file1 file2
mv file1 [file2 ...] directory
mv dir1 dir2
mv dir1 [dir2 ...] directory
```

Primer: Neka je početni folder ./program/source. Premesti Hello2.cpp iz ./program/bin u ./program/object. Onda, promeni ime Hello2.cpp u Hello3.cpp.

```
> pwd
> ls -la
> mv  ./../bin/Hello2.cpp  ./../object/
> mv ./../object/Hello2.cpp  ./../object/Hello3.cpp
> ls ./../object/
> cat ./../object/Hello3.cpp
```

Komanda **rm koristi se za brisanje fajlova i foldera**. Posle brisanja, obrisani fajlovi se ne mogu povratiiti! Zato ovu komandu treba koristiti oprezno.
Sintaksa:

rm [-irf] file(s)/directory(ies)

Opcije:

-i brisanje tek nakon potvrde

-r podfolder biće rekurzivno obrisani (sa svim podfolderima)

-f force: brisati bez bezbednosnih upita.

Vežba: Iz tekućeg direktorijuma*./program/source*. obrisati *Hello3.cpp* iz ./program/object/. Vratiti se u glavni folder.

```
> pwd
> ls -a
> rm -i ./../object/Hello3.cpp
> ls ./../object/
> ls ./../bin/
> cd ./../..
> pwd
```

Pristupne dozvole (access permissions).

Tri različita prava pristupa:

r - read: dozvola čitanja sadržaja datoteke ili za direktorijuma.

w - write: dozvolla modifikovanja datoteka (uključujući brisanje).

x - execute: dozvola izvršavanje binarnih datoteka (naredbi, programa) i iz komandne linije

Prava pristupa su pojedinačno definisana za:

- u- vlasnika
- g – grupe kojoj vlasnik pripada
- o – ostalim korisnicima
- a – svim korisnicima

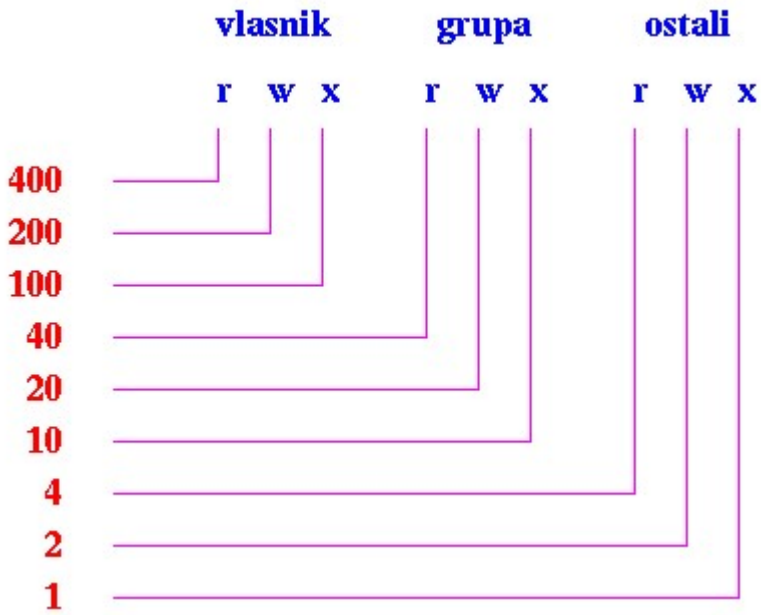
I simbolički

Prava pristupa mogu da se promene koristeći **chmod**.

Sintaksa: **chmod [ugoa][+-=][rwx] file(s)/directory(ies)**

```
% chmod a=r,u+w dat1
% ls -l dat1
 -rw-r--r-- 1 borko user 1046 Nov 15 13:01 dat1
% chmod a-rwx *
% ll
 ---------- 1 borko user 1046 Nov 15 13:01 dat1
% chmod u+rw,g+r,o+r *
% chmod u+rw,go+r *
% ll
 -rw-r--r-- 1 borko user 1046 Nov 15 13:01 dat1
```

II oktalno



primeri:

```
% chmod 444 dat1
% ls -l dat1
 -r--r--r-- 1 borko user 1046 Nov 15 13:01 dat1
% chmod 644 dat1
% ls -l dat1
 -rw-r--r-- 1 borko user 1046 Nov 15 13:01 dat1
```

Simbolički linkovi

- Simbolički link je specijalna datoteka koja predstavlja referencu na ime druge datoteke ili direktorijuma.
- Korisno da se smanji zauzeće memorije na disku kada dve datoteke imaju isti sadržaj.

Kako prepoznati simboličke linkove? Komandom **ls -l** .

ls –l prikazuje <naziv_linka> -> <naziv_datoteke>

- Komanda za pravljenje simboličkog linka:

> **ln –s <naziv_datoteke> <naziv_linka>**

- Pravljenje linka na datoteku u drugom direktorijumu, sa istim imenom

> **ln –s ./../ <naziv_datoteke>**

- Brisanje linka

> **rm <naziv_linka>**

Poređenje fajlova i direktorijuma

Sintaksa: **cmp -ls** *file1 file2*

Komanda za poređenje sadržaja dva fajla. Ukoliko su identični, nema komentara, ali ukoliko ima razlika, ispisuje se broj bajtova i broj linija u kojima se nalaze razlike.

- **-l** ispisuje razlike izmedju dva fajla,
- **-s** vraca samo kod (0-iste, 1-razlicite, 2-ne moze da pristupi argumentu).

```
% cd ./program/source
% cmp Hello.cpp Hello2.cpp
```

Komanda **diff** poredi sadržaj dva fajla. **diff** je u stanju da upoređuje i direktorijume. Osim ovoga, daje širi i precizniji izveštaj o eventualnim razlikama. Ako su isti fajlovi , ne daje ništa na izlazu.

Sintaksa: **diff file1 file2**

Pretraživanje fajlova

Komanda **find** se rekurzivno kreće kroz hijerarhiju fajl sistema počev od direktorijuma ***starting_dir*** tražeći fajlove koji zadovoljavaju logički izraz ***matching_criteria*** i nad takvim fajlovima izvrsava akcije.

find *starting_dir* [*matching_criteria_&_actions*]

Primeri:

```
% find ~borko/ -name Test.txt -print
/home/mobaxterm/program/source/Test.txt
```

Traži se od home direktorijuma fajl sa imenom ***Test.txt.*** Pronađen je u direktorijumu **~borko/program/source/**.

```
% find /usr/people/borko \(-name a.out -o -name core \ -o -name
   ".BAK.*"\) -atime +14 -exec rm -f '{}' ';`
```

Ispod direktorijuma /usr/people/borko, pronadji sve fajlove a.out, core i one čije ime pocinje sa .BAK., kojima nije pristupano više od 14 dana i izbriši ih.

Komanda **grep** pretražuje fajlove i trazi stringove u njima.

- Svaka linija koja zadovoljava *expression* štampa se na standardni izlaz.
- U slučaju da se istovremeno pretražuje više fajlova, ispred svake linije štampa se i ime fajla iz koje potiče.
- Važna opcija je [-i], koja primorava grep da ignoriše svaku razliku između velikih i malih slova.

Primer: Rekurzivno traži tekst „text“ u svim * .txt datotekama u direktorijumu **~borko/program/**

```
%grep 'text.' -r ~borko/program/
%grep 'text' -r ~borko/program/ *.txt
%grep 'include' -r  ~borko/program/ *.cpp
```

Arhiviranje

Arhiviranje (tar) - komanda za zapis, čitanje i listanje sadržaja arhive.

tar [*arguments*] [*name*]

Argumenti:

- -c – (compress) zapis od početka arhive.
- -r - zapis na kraju arhive (dodavanje fajlova starom sadržaju).
- -t - listanje sadržaja arhive.
- -x – (extract) čitanje sadržaja arhive i zapis na disk.
- -f - definisanje arhive (naredni argument je ime fajla arhive).
- -v - za vreme snimanja ili čitanja arhive ispisuju se imena fajlova koji se trenutno obrađuju (*verbose mod*).
- name - ime fajla ili direktorijuma koji se arhivira ili restaurira (uključuju se i podirektorijumi).

Primer: napraviti **(c- create)** arhivu - fajl **(f- file)** *program.tar.gz* koristeći kompresiju gunzip **(z)** direktorijuma *./ program*. Uključiti *Verbose* mod (v).

```
>cd ~borko
>tar -zcvf program.tar.gz   ./program
```

Obrisati folder. /program

```
>   rm –r ./program
>   ls -la
```

Ekstrahovati originalni direktorijum (./program) iz arhive *program.tar.gz* koristeći dekomresiju podataka gunzip (z).

```
> tar -zxvf program.tar.gz
```

Administriranje procesa

Proces je pokrenut program i sastoji se od samog programa i odgovarajućeg okruženja, koje se sastoji od svih potrebnih dodatnih informacija potrebnih za osiguravanje ispravnog toka.

Karakteristike procesa su (između ostalog):

- jedinstveni ID procesa (PID),
- PID nadređenog procesa (PID),
- broj korisnika i grupe vlasnika
- i prioritet procesa.

- **su [-] [*name*]**

- **Omogućava privremeni login na root bez odjavljivanja sa svog naloga.**
- **Da bi ovo bilo moguće, neophodno je poznavanje administratorski password.**
- **Odjavljivanje sa ovog nakloga se obavlja < Ctrl >D ili logout komandom.**

primeri:

```
% su
password: xxxxxx
```

Komanda ps

Štampa na ekranu informacije o trenutno aktivnim procesima.

Sintaksa: **ps [-al] [-u user]**

- Ako se koristi bez opcija, prikazuju se samo procesi korisnika koji se izvode u trenutnom terminalu.

Važne opcije:

-a , prikazati sve procese dodeljene bilo kom terminalu

-l, prikaz dugog formata. Prikazuju se dodatne informacije o vlasniku, roditeljskom procesu itd.

-u prikazuju se svi procesi koji su u vlasništvu određenog korisnika.

% ps

Daje listu procesa koji su startovani sa tekućeg terminala.

% ps -u *user_name*

Daje listu aktivnih procesa korisnika *user_name*.

Komanda kill

Komanda koja ubija aktivirane procese.

• ***PID*** **- identifikator procesa**

•Ubija proces sa zadatim PID brojem.

•Može da izvrši samo vlasnik procesa ili root

Sintaksa: **kill [-9] PID**

Važna opcija:

- **-9 za „tvrdoglave“ procese koji se ne mogu prekinuti običnom kill naredbom.**

Primer: Izlistajte procese i završite neki nepotrebni proces pomoću naredbe kill

```
> whoami
> ps –u user
> kill 12999
> ps –u user
```

Kreiranje izvršnih fajlova

Komanda **g++**

Sintaksa

g++ [-c] [-g] [-O] file.cpp [-o outputfile]

GNU C ++ kompajler je uključen u svaku Linux distribuciju.

Važne opcije:

- -o ime izvršnog objekta (programa). Podrazumevano „a.out“.
- -c samo kompajlirati, ne povezivati.
- -g uključuje informacije o otklanjanju grešaka. Neophodno za kasnije otklanjanje grešaka. Povećava veličinu i vreme izvršavanja programa.
- -O optimizacija (podrazumevano: srednja optimizacija)

- Primer: Kompajlirati Hello.cpp file, napraviti izvršni fajl i pokrenuti program

```
> cd ./program/source/
> rm Hello2.cpp Test.txt
> g++ Hello.cpp
> ls
> ./a.out
> rm a.out
> g++ Hello.cpp -o ./hello
> cd ./../bin
> ./hello
> pwd
> g++ -c ./../source./Hello.cpp
> ls
> g++ Hello.o -o hello
> ./hello
```