# Napredna administracija operacijskog sustava na poslužitelju

Debian

L201

OBRAZOVNI PROGRAMI srca

**priručnik za polaznike**

srce
Sveučilište u Zagrebu
Sveučilišni računski centar

Ovu su inačicu priručnika izradili:

Autor: mr. sc. Branimir Radić

Recenzent: Darko Culej

Urednik: Dominik Kenđel

Lektor: dr. sc. Jasna Novak Milić

Sveučilište u Zagrebu

Sveučilišni računski centar

Josipa Marohnića 5, 10000 Zagreb

edu@srce.hr

ISBN 978-953-8172-99-1 (meki uvez)

ISBN 978-953-382-000-2 (PDF)

Verzija priručnika L201-20221117

**Sadržaj**

# Uvod

**Trajanje poglavlja:**
**10 min**

Po završetku ovoga tečaja moći ćete:

- prilagoditi jezgru operacijskog sustava i proces pokretanja
- uspostaviti softversko RAID polje
- uspostaviti datotečni sustav na fizičkom i logičkom spremištu
- nadzirati stanje pojedinih procesa na sustavu
- izraditi skripte za automatizaciju
- izraditi Debian pakete i prilagoditi postojeće

Ovo je tečaj napredne administracije u Debian linuxu. U tečaju je dodatno razrađeno administrativno upravljanje operacijskim sustavom linux (Debian): koraci pokretanja sustava i napredne izmjene pokretanja, zaustavljanja i izmjena postavaka pokretanja u linuxu, rad sa datotečnim sustavima i spremišnim sustavima te njihovi tipovi i mogućnosti, nadzor aktivnosti na sustavu kao i prilagodba sustava izradom osnovnih skripti i provođenje osnovnih izmjena paketa.

Tečaj je namijenjen polaznicima sa osnovnim predznanjem rada u linuxu i u komandnoj liniji. Osnovno poznavanje rada na računalima je obavezno za uspješno praćenje sadržaja.

# 1. Jezgra i pokretanje operacijskog sustava Linux

**Trajanje poglavlja:**
**215 min**

Po završetku ove cjeline moći ćete:

- razumjeti ulogu komponenti jezgre operacijskog sustava Linux
- prilagoditi i programsko prevesti jezgru operacijskog sustava Linux
- instalirati zakrpe u jezgru operacijskog sustava Linux
- razumjeti rad **systemd**
- raditi s i razumjeti jedinice (engl. unit) i mete (engl. target)
- izmijeniti konfiguraciju **systemd.**

U ovoj se cjelini obrađuje jezgra operacijskog sustava *Linux*. U cjelini je opisan postupak primjene zakrpa jezgre i programsko prevođenje jezgre. Cjelina obrađuje i postupak pokretanja sustava, kao i postupak izmjene postavki pokretanja sustava te sustav **systemd** koji upravlja pokretanjem jedinica.

## 1.1. Jezgra operacijskog sustava Linux

### 1.1.1. Datoteke jezgre

Jezgra, programski prevedena iz izvornog kôda, u izvršnom se obliku sastoji od velikog broja datoteka. Najvažnije datoteke koje sačinjavaju jezgru pripadaju dvjema distinktnim kategorijama:

1. **Središnja datoteka jezgre** – u ovoj se datoteci nalaze sve komponente jezgre osim modula. Program za učitavanje operacijskog sustava (engl. *bootloader*) učitava ovu datoteku pri pokretanju sustava te ona upravlja radom sustava od trenutka njezinoga učitavanja u memoriju. Središnja se datoteka smješta u direktorij /**boot**. Središnja datoteka jezgre može imati različita imena, a značenja tih imena objašnjena su u tablici 1.
2. **Moduli** – datoteke koje sadrže module jezgre nisu dio središnje datoteke i standardno se smještaju u direktorij /**lib/modules**. Poddirektoriji u ovom direktoriju povezani su na različite inačice jezgre instalirane na sustavu. Moduli se učitavaju i uklanjaju iz jezgre na zahtjev.

| Ime varijable | Značenje |
|---|---|
| **vmlinux** | Nekomprimirana datoteka jezgre, uobičajeno generirana kao jedan od koraka izrade nove inačice jezgre, pri čemu ne smije biti posljednji. Ova se datoteka ne može koristiti za pokretanje sustava zbog toga što joj nedostaju neki elementi. |
| **vmlinuz** | Komprimirani oblik datoteke **vmlinux** koji je dodatno prilagođen dodavanjem ključnih elemenata koji omogućavaju pokretanje sustava ovom inačicom. Standardno, ovo ime koriste *Linux* distribucije za programski prevedene i prilagođene datoteke jezgre spremne za korištenje. |
| **zImage** | Zastarjeli komprimirani format sličan formatu **vmlinuz**. Veličina ovoga tipa je ograničena na 512 KiB i zbog toga je neprimjenjiva za današnje jezgre. |
| **bzImage** | Komprimirani oblik datoteke jezgre. Ovaj format izrađen je kako bi se prevladala ograničenja formata **zImage**. I danas se koristi, uglavnom za lokalno prilagođene i prevedene inačice datoteka jezgre. |
| **kernel** | Generičko ime koje se koristi na nekim sustavima koji koriste GRUB 2. U pitanju je standardno preimenovana **bzImage** datoteka jezgre. |

Tablica 1 – Značenje imena datoteka jezgre

### 1.1.2. Programsko prevođenje jezgre

Prvi (pred)korak pri izradi nove datoteke jezgre jest identifikacija hardvera. Naime, da bismo znali koji su upravljački programi i moduli potrebni središnjoj jezgri za rad, trebamo znati koji hardver sačinjava sustav.

Jednostavno pravilo pri konfiguraciji jezgre jest da se u slučaju nesigurnosti oko potrebe neke mogućnosti ili modula isti dodaju u konfiguraciju. Nepotrebno programsko prevođenje takve mogućnosti ili modula će dodati nekoliko sekundi programskomu prevođenju i potrošiti malo diska u prevedenoj inačici, ali nekorišteni moduli ne troše dodatnu memoriju i procesor pri radu sustava. Smatra se da je bolje potrošiti nekoliko dodatnih sekundi u prevođenju nego stvoriti nefunkcionalnu jezgru.

Identifikacija hardvera može se provesti čitanjem dokumentacije hardvera, korištenjem konfiguracije postojeće jezgre kao modela, vizualnim pregledom komponenti (kada se ne radi o virtualnom hardveru) ili nekim specijaliziranim alatom poput **lshw**.

Nakon identifikacije hardvera potrebno je identificirati namjenu jezgre. Jednom kada je poznato na kojem hardveru i za koju namjenu će jezgra biti korištena, može započeti prvi korak programskoga prevođenja jezgre – konfiguracija.

### Izbor konfiguracijskih postavki jezgre

Pri programskom prevođenju jezgre postoji veći broj mogućnosti nego pri prevođenju standardnih paketa. U Tablici 2 nabrojane su i objašnjene standardne mogućnosti naredbe `make`, kojima se provodi programsko prevođenje jezgre.

| Mogućnost | Objašnjenje |
|---|---|
| **mrproper** | Brisanje starih privremenih i konfiguracijskih datoteka. |
| **oldconfig** | Koristi se stara konfiguracija na način da se ažuriraju samo nove postavke. |
| **silentoldconfig** | Sličan opciji oldconfig, ali s manje detaljnim prikazom na ekranu. |
| **defconfig** | Stvara se konfiguracija korištenjem osnovnih postavki za konfiguraciju hardvera. |
| **allmodconfig** | Kreira se konfiguracija koja maksimalno implementira modularnost. |
| **config** | Potpuna konfiguracija kroz tekstualno sučelje. |
| **menuconfig** | Potpuna konfiguracija kroz tekstualnu simulaciju GUI sučelja. |
| **xconfig** | Potpuna konfiguracija kroz GUI. |
| **gconfig** | Potpuna konfiguracija kroz GUI zasnovan na GTK+. |

Tablica 2 – Mogućnosti naredbe make

Rezultat poziva naredbe `make help` u direktoriju u kojem se nalaze datoteke jezgre za prevođenje daje opširni pregled mogućnosti naredbe `make` za inačicu jezgre smještenu u tom direktoriju. Prikaz je informativniji od gornje tablice i specifičan isključivo za tu jezgru.

Najpregledniji način konfiguracije jest preko grafičkog sučelja, opcijom **xconfig**. Kada na računalu ne postoje paketi grafičkoga sučelja, tada je gotovo jednako dobra mogućnost **menuconfig**.

Mogućnosti pri konfiguraciji jezgre se mijenjaju često, razvojem novih inačica jezgre. Brojnost svih mogućnosti čini beskorisnim i nepraktičnim opis svih mogućnosti. Glavne se kategorije mijenjaju rijetko i u trenutačno aktualnoj inačici jezgre (4.8.2) to su kategorije:

Korisnikova ljuska (u većini slučajeva je to bash).

| Kategorija | Objašnjenje |
|---|---|
| **General Setup** | Mogućnosti visoke razine koje uglavnom konfiguriraju podržane standarde i način na koji se pokreće sustav. |
| **Bus Options (PCI, etc.)** | Definira se podrška za vrste sabirnica. |
| **Executable File Formats/Emulations** | Definira se podrška za različite vrste izvršnih datoteka i podrška za emulaciju izvršavanja pri |

| | |
|---|---|
| | pokretanju nekih izvršnih datoteka (staroga formata). |
| **Device Drivers** | Definicija podrške za uređaje. Velik broj mogućnosti za uređaje različitih vrsta i generacija. Moguće je isključiti velik broj mogućnosti, ali se mora precizno znati svaki dio hardvera i biti siguran da u budućnosti neće biti nepredviđenih izmjena hardvera. |
| **Firmware Drivers** | Mogućnosti vezane za *firmware*. |
| **File Systems** | Podrška za različite vrste datotečnih sustava i particijskih shema. Neki datotečni sustavi trebaju dodatne pakete, ali bez podrške u jezgri neće raditi. |
| **Kernel Hacking** | Mogućnosti važne uglavnom razvojnomu timu jezgre. |
| **Virtualization** | Mogućnosti za konfiguraciju određenih virtualizacijskih značajki koje omogućavaju pokretanje drugih operacijskih sustava unutar pokrenutoga *Linuxa*. |
| **Networking Support** | Konfiguracija za određene mrežne protokole. Ne odnosi se na mrežne uređaje (oni pripadaju segmentu „Device Drivers“). |
| **Processor Type and features** | Konfiguracija o tome kako *Linux* upravlja CPU-om, neke od postavki su nužne ako želimo koristiti ranije spomenutu virtualizaciju. |
| **Power management and ACPI options** | Postavke za minimalizaciju potrošnje energije, poput podrške za hibernaciju (u RAM-u i na hard disku). |
| **Security options** | Sigurnosne mogućnosti poput podrške za određene sigurnosne module i *SELinux* postavke. |
| **Cryptographic API** | Neke značajke jezgre zahtijevaju prisutnost određenih kriptografskih modula. Moguće je omogućiti/onemogućiti brojne module. |

Tablica 3 – Kategorije pri konfiguraciji jezgre

**Programsko prevođenje jezgre**

Nakon uspješnog izbora postavki mogu se programskim prevođenjem izraditi datoteke jezgre i prilagoditi sustav za korištenje nove jezgre. Postupak programskoga prevođenja jednostavan je poput prevođenja bilo kojega drugog softvera. Posebnost kod prevođenja jezgre jest sama instalacija jezgre, zbog toga što je potrebno kopirati datoteku jezgre u direktorij /**boot**, instalirati module u za to predviđeni direktorij, pripremiti inicijalnu RAM datoteku, podesiti **GRUB** ili drugi program za učitavanje sustava i ponovno pokrenuti sustav.

Naredba `make` kreirat će središnju datoteku jezgre i sve module. To je zbog toga što je naredba `make` ekvivalent naredbi `make all`. Naredbom `make modules` se mogu generirati moduli.

**Instalacija jezgre** sastoji se od jednostavnog kopiranja datoteke jezgre u direktorij /**boot**/. Jednom kada je kopirana u direktorij /**boot**/, dobra je ideja preimenovati datoteku tako da ime sadržava i verziju jezgre i na taj način omogućiti postojanje više jezgri paralelno.

```
# cp ./arch/x86/boot/bzImage /boot/bzImage-4.8.10
```

**Instalacija modula** provodi se naredbom `make modules_install`. Instalacija se sastoji od kreiranja poddirektorija u /**lib/modules** (za potrebe trenutačno aktivne jezgre to bi bio /**lib/modules/4.8.10**) i kopiranja modula u taj direktorij i njegove poddirektorije. Za rad sustava potreban je valjani popis ovisnosti modula. Taj se popis nalazi u modules.dep datoteci koja se nalazi u direktoriju modula te se regenerira za trenutačno aktivnu jezgru izvođenjem naredbe depmod.

**Dodavanje nove jezgre u GRUB konfiguraciju**

**GRUB Legacy**

Najbrži način dodavanja nove jezgre jest pronalaženje konfiguracije koja definira trenutačno aktivnu jezgru, kopiranje toga segmenta i prilagođavanje zapisa novoj jezgri. Tipičan zapis koji definira jednu jezgru jest:

```
title Debian (3.13-1-486)
  root (hd0,0)
  kernel /bzImage-3.13-1-486 ro root=/dev/sda5
  initrd /initrd.img-3.13-1-486
```

i nalazi se u središnjoj konfiguracijskoj datoteci /**etc/grub/grub.cfg**.

S obzirom na to da se lokacija *root* particije neće mijenjati s novom jezgrom, taj bi zapis izmijenili za korištenje nove, u ranijim primjerima prevedene jezgre na ovaj način:

```
title Debian (4.8.10)
  root (hd0,0)
  kernel /bzImage-4.8.10 ro root=/dev/sda5
  initrd /initrd.img-4.8.10-amd64b
```

Dakle, potrebno je definirati tri parametra:

1. ime nove jezgre (*title*)
2. datoteku jezgre
3. inicijalnu RAM datoteku.

Ostali parametri najčešće su identični za sve jezgre na sustavu, a to su:

- particija na kojoj se nalazi *root* direktorij i
- uređaj na kojem se nalazi ta particija.

**GRUB 2**

**GRUB 2** koristi sonda-skripte za **update-grub** i **grub-mkconfig** kako bi automatski generirao valjanu datoteku /**boot/grub/grub.cfg** koja je ujedno njegova središnja konfiguracijska datoteka. Poželjno je prije izvršavanja ovih naredbi napraviti sigurnosnu kopiju /**boot/grub/grub.cfg** datoteke. Zatim će se pokrenuti sonda-skripta te se pomoću naredbe **diff** može odrediti koje su se izmjene dogodile u odnosu na izvornu datoteku.

```
# update-grub2
Generating grub configuration file ...
Found background image: /usr/share/images/desktop-base/desktop-grub.png
Found linux image: /boot/vmlinuz-4.8.10
Found initrd image: /boot/initrd.img-4.8.10
Found linux image: /boot/vmlinuz-3.13-1-486
Found initrd image: /boot/initrd.img-3.13-1-486
done
```

Kao što je vidljivo iz primjera, naredba `update-grub2` javlja koje je datoteke jezgre i datoteke inicijalnoga RAM datotečnoga sustava pronašla u direktoriju **/boot/**. U skladu s pronađenim, obavljaju se izmjene na datoteci /**boot/grub/grub.cfg**. Važno je napomenuti da se neće promijeniti datoteke u direktoriju /**etc/grub.d/**. Te datoteke ostaju nepromijenjene, a ono što se mijenja jest rezultat koji dobiju sonda-skripte. Dakle, generirana datoteka je rezultat predložaka smještenih u direktoriju /**etc/grub.d/** te /**etc/default/grub** datoteke i podataka prikupljenih sondama.

### 1.1.3. Instalacija zakrpa i prilagodba jezgre

Zakrpa za jezgru je tekstualna datoteka u kojoj se nalaze razlike između dviju različitih inačica jezgre. Zakrpe se kreiraju pomoću naredbe `diff` i ono što se primjenom zakrpe zapravo dogodi jest da se izvorna verzija jezgre pretvori u ciljanu verziju. Za uspješno primjenjivanje zakrpe potrebno je znati za koju je izvornu verziju zakrpa namijenjena. Ta informacija treba biti dostupna kao meta zapis u zakrpi ili mora biti jasna iz samog imena datoteke zakrpe. Primjerice, zakrpa imena **patch-4.10-rc4** namijenjena je za jezgru **4.10, rc1** do **rc3** i te će inačice podići na inačicu **4.10-rc4**.

Važno je napomenuti da se zakrpe primjenjuju na datoteke jezgre prije programskog prevođenja. Svrha zakrpa je prije svega smanjivanje mrežnoga prometa. Zakrpe su primarno namijenjene za situacije kada se želi kontinuirano imati najnovija inačica jezgre, kako se ne bi trebao svaki put pribavljati sav izvorni kôd. Takve su situacije rijetke u produkciji i koriste se samo na sustavima gdje se testiraju napredne mogućnosti jezgre.

Zakrpu primjenjujemo naredbom `patch`. Naredba se primjenjuje na datoteke u trenutačnom direktoriju, a kao ulazni argument prihvaća datoteku zakrpe. Standardno je da se u definiciji zakrpe očekuje da se nalazimo u direktoriju u kojem je direktorij s jezgrom. Ako se nalazimo u samom direktoriju jezgre, moramo koristiti mogućnost **–p1** naredbe `patch` kako bi se ignorirala jedna razina datotečnoga stabla, kako je prikazano u primjeru:

```
$ ls
linux-4.8.11  linux-4.8.11.tar.xz  patch-4.8.11-12
$ cd linux-4.8.11/

$ patch -p1 < ../patch-4.8.11-12
patching file Makefile
patching file arch/parisc/Kconfig
patching file arch/parisc/kernel/cache.c
…
```

Budući da je zakrpa jezgre samo set **diff** zapisa o razlikama datoteka između dvije inačice, moguće je jednostavno provesti postupak u suprotnom smjeru odnosno „deinstalirati“ zakrpu. Mogućnost naredbe `patch` za deinstalaciju zakrpe jest –R.

```
$ patch -p1 -R < ../patch-4.8.11-12
patching file Makefile
patching file arch/parisc/Kconfig
patching file arch/parisc/kernel/cache.c
…
```

Kako je prikazano u primjeru, izlaz naredbe se ne mijenja. Jednostavno se na istim datotekama izvornoga kôda jezgre primjenjuju izmjene u suprotnom smjeru. Od mogućnosti naredbe `patch` još je značajna mogućnost **–b** koja čuva izvorne oblike datoteka dodajući im **.orig postfix**. Korištenje i dobiveni rezultat prikazani su u primjeru:

```
$ patch -p1 -b < ../patch-4.8.11-12
patching file Makefile
patching file arch/parisc/Kconfig
patching file arch/parisc/kernel/cache.c
…
$ ls -R |grep .orig
Makefile.orig
exynos4210-origen.dts
exynos4412-origen.dts
…
```

### 1.1.4. Izrada programskoga paketa s novom jezgrom

Kada se prevedena jezgra planira koristiti na više računala, iz aspekta održavanja pogodno je napraviti **programski paket** (RPM ili deb, ovisno o sustavima na koje će biti instalirana). Danas su u izvornom kôdu jezgre dostupne mogućnosti **rpm-pkg**, **binrpm-pkg, deb-pkg** ili **bindeb-pkg** za naredbu `make`. Naravno, ove mogućnosti, poput **xconfig**, trebaju odgovarajuće alate instalirane na računalu kako bi uspješno kreirale paket jezgre. Paketi kreirani korištenjem ovih mogućnosti bit će smješteni u direktoriju **/usr/src** na distribuciji **Debian** i u **/root/rpmbuild** na distribucijama koje koriste RPM.

Ovako izrađeni paket može se kopirati na bilo koje računalo kompatibilne arhitekture i hardvera te instalirati kao bilo koji drugi paket jezgre. Ipak, potrebno je biti posebno oprezan kada se želi izraditi paket koji će biti instaliran na različitom hardveru. Jezgra tada treba biti prevedena s podrškom za **sav** hardver koji postoji na računalima za koja se izrađuje paket jezgre. Što više različitih komponenti poput mrežnih kartica, matičnih ploča, procesora i sličnoga postoji, to je teže uspješno konfigurirati jezgru koja će raditi na svim računalima.

Mogućnosti naredbe `make` za izravnu izradu paketa navedene su u tablici u nastavku.

| Mogućnost | Objašnjenje |
|---|---|
| **rpm-pkg** | Izrada **.RPM** paketa jezgre, izradit će se izvršni i paket izvornoga kôda. |
| **binrpm-pkg** | Izrada samo **.RPM** paketa izvršnoga kôda. |
| **deb-pkg** | Izrada **.deb** paketa jezgre, izradit će se izvršni i paket izvornoga kôda. |
| **bindeb-pkg** | Izrada samo **.deb** paketa izvršnoga kôda |

Tablica 4 – Mogućnosti naredbe make za izradu paketa

## 1.2. Pokretanje operacijskog sustava Linux

### 1.2.1. systemd

**systemd** je init sustav koji se koristi u *Linux* distribucijama za upravljanje korisničkim prostorom i servisima. Ime **systemd** pridržava se Unix konvencije imenovanja daemona dodavanjem slova *d* na kraj imena servisa. **systemd** je konfiguriran na način da je kompatibilan s upstart i SysV init skriptama. Na brojnim sustavima je **systemd** danas standardan sustav.

Novi koncept koji **systemd** donosi jest koncept **jedinica**. Jedinice su predstavljene konfiguracijskim datotekama smještenim u direktorijima navedenim u donjoj tablici.

| Direktorij | Objašnjenje |
|---|---|
| **/lib/systemd/system** | Konfiguracijske datoteke jedinica distribuiranih s instaliranim paketima. |
| **/run/systemd/system/** | Konfiguracijske datoteke jedinica kreirane pri radu. Sadržaj ovoga direktorija je većeg prioriteta od prethodno navedenog u tablici. |
| **/etc/systemd/system/** | Konfiguracijske datoteke jedinica kreirane naredbom systemctl enable. Datoteke u ovom direktoriju su najvišega prioriteta. |

Tablica 5 – Direktoriji u koje se smještaju konfiguracije jedinica

**systemd** podržava devet glavnih mogućnosti:

1. **Aktivacija putem utičnica (engl. *socket*)** – pri pokretanju sustava **systemd** kreira utičnice za sve servise koji podržavaju takvu aktivaciju. U trenutku kada je servis pokrenut dodjeljuje mu se njegova utičnica. **systemd** brine o porukama na definiranim utičnicama

tako da se ne gubi komunikacija, već se u slučaju nedostupnosti servisa njemu upućene poruke spremaju u red i dostavljaju u trenutku kada je servis aktivan. Koriste se jedinice utičnica.

2. **Aktivacija putem sabirnice (engl. *bus*)** – servisi koji koriste D-Bus virtualnu sabirnicu za među-procesnu komunikaciju mogu biti aktivirani kada im je putem te sabirnice upućen zahtjev. Koriste se D-Bus servisne datoteke.
3. **Aktivacija putem uređaja (engl. *device*)** – servisi koji to podržavaju mogu se pokrenuti kada se priključi i aktivira određeni uređaj. Koriste se jedinice uređaja.
4. **Aktivacija zasnovana na putanji (engl. *path)*** – moguće je pokretanje servisa koji to podržavaju na promjenu određene datoteke ili direktorija. Koriste se jedinice putanje.
5. **Slike stanja sustava (engl. *snapshot)*** – **systemd** može privremeno pohraniti stanje svih servisa ili vratiti ranije dinamički pohranjeno stanje svih servisa. Za pohranu stanja sustava koriste se jedinice slike stanja.
6. **Upravljanje točkama za montiranje (engl. *mount*) i automatsko montiranje (engl. *automount*)** – systemd koristi mount i automount jedinice za upravljanje i nadzor.
7. **Agresivna paralelizacija (engl. *aggressive parallelization)*** – zbog aktivacije putem utičnica systemd može u paralelu pokrenuti sve servise dokle god su dostupne sve utičnice. Uz servise koji podržavaju aktivaciju na zahtjev, moguće je paralelnim pokretanjem značajno skratiti vrijeme potrebno za pokretanje sustava.
8. **Transakcijska logika (engl. *transactions)*** – pri pokretanju ili zaustavljanju jedinice systemd, na osnovi ovisnosti kreira privremenu transakciju i provjerava konzistenciju. Ako postoje nekonzistentnosti, systemd će ih prvo neinvazivno probati ukloniti, a tek nakon toga javiti grešku.
9. **Kompatibilnost sa SysV initom (engl. *backward compatibility)*** – systemd podržava SysV init skripte opisane u „Linux Standard Base Core Specification“. Ovo olakšava nadogradnju na systemd servisne jedinice. Te su skripte posebne jer započinju zaglavljem koje dodatno definira ovisnosti servisa i redoslijed pokretanja.

**systemd – kompatibilnost**

Iako je visoko kompatibilan sa SysV initom i Upstartom, ipak postoje značajne razlike u kompatibilnosti koje je važno navesti.

**systemd** posjeduje tek ograničenu podršku za razine izvođenja. Analogno razinama izvođenja, **systemd** podržava mete, a mete su zapravo skupine jedinica. Mete su kasnije objašnjene i uspoređene s razinama izvođenja. Zbog kompatibilnosti systemd dolazi s naredbom `runlevel`, ali ne postoje odgovarajuće mete koje su ekvivalenti ranije postojećih razina izvođenja. Zbog toga se može lako dogoditi da izvođenje naredbe `runlevel` vrati vrijednost, odnosno bude neuspješno. Zbog toga se savjetuje izbjegavanje korištenja naredbe `runlevel` sa sustavom systemd.

Središnji alat u sustavu **systemd** jest **systemctl**. **Systemctl** ne komunicira sa servisima koje ne pokreće alat **systemctl**. Pri pokretanju neke mete **systemctl** pohrani **pid** (process id) kojim se nadzire stanje mete. Stoga, ako je servis pokrenut neovisno od **systemctl**, tada se njime ne može nadzirati stanje toga procesa niti izmijeniti stanje procesa.

**systemd** gasi samo aktivne servise pri zaustavljanju sustava, a ne sve servise koji se nalaze u /**etc/rc0.d/** kako je bilo ranije. Ne dopušta interakciju s aktivnim servisima putem standardnog

inputa, već pri pokretanju postavlja standardni input na /**dev/null**. Servisi ne nasljeđuju kontekst (varijable okoline) od korisnika koji ih pokreće. Kontekst koji im dodijeli systemd neovisan je o pokretaču i/ili njegovoj okolini.

Kada se koriste SysV init skripte, systemd će pri pokretanju pročitati i interpretirati LSB zaglavlje skripte. U tom se zaglavlju nalaze pravila pokretanja servisa poput popisa servisa i uređaja o kojima dani servis ovisi i slično.

Sve servisne jedinice imaju definirano vrijeme čekanja (engl. *timeout*) od 5 minuta. To se vrijeme može produžiti eksplicitno ga definirajući u konfiguraciji za pojedine servise. Svrha ovoga vremena jest da zaštiti sustav od potencijalnog blokiranja jednim servisom.

**systemd – upravljanje servisima**

Ranije korištene **init** skripte smještene u **direktoriju /etc/init.d/** zamijenjene su servisnim jedinicama. Datoteke servisnih jedinica završavaju sa **.service**. Načini na koje se koristi alat **systemctl** u usporedbi s alatima **service** i **chkconfig** prikazani su u donjim tablicama.

| Korištenje service | Korištenje systemctl | Objašnjenje |
|---|---|---|
| **service <ime> start** | systemctl start <ime>.service | Pokretanje servisa. |
| **service <ime> stop** | systemctl stop <ime>.service | Zaustavljanje servisa. |
| **service <ime> restart** | systemctl restart <ime>.service | Ponovno pokretanje servisa. |
| **service <ime> condrestart** | systemctl try-restart <ime>.service | Ponovno pokretanje servisa samo ako je već pokrenut. |
| **service <ime> reload** | systemctl reload <ime>.service | Učitavanje nove konfiguracije. |
| **service <ime> status** | systemctl status <ime>.service<br><br>systemctl is-active <ime>.service | Provjera je li servis pokrenut. |
| **service --status-all** | systemctl list-units --type service --all | Prikaz status svih servisa. |

Tablica 6 – Standardne mogućnosti naredbi service i systemctl

| Korištenje chkconfig | Korištenje systemctl | Objašnjenje |
|---|---|---|
| **chkconfig <ime> on** | systemctl enable <ime>.service | Uključivanje servisa. |
| **chkconfig <ime> off** | systemctl disable <ime>.service | Isključivanje servisa. |
| **chkconfig --list <ime>** | systemctl status <ime>.service<br><br>systemctl is-enabled <ime>.service | Provjera statusa (uključen/isključen) servisa. |
| **chkconfig --list** | systemctl list-unit-files --type service | Ispis svih servisa sa statusom. |
| **-** | systemctl list-dependencies --after | Ispis servisa koji se pokreću prije određene jedinice. |
| **-** | systemctl list-dependencies --before | Ispis servisa koji se pokreću nakon određene jedinice. |

Tablica 7 – Standardne mogućnosti naredbi chkconfig i systemctl

Osnovne naredbe za upravljanje sustavom i servisima navedene su u sljedećoj tablici.

| Najsličnija naredba | systemctl naredba | Objašnjenje |
|---|---|---|
| **chkconfig --list \| grep :on** | systemctl list-units | Popis jedinica koje se nalaze u memoriji systemd. |
| **NEMA** | systemctl daemon-reload | Ponovno učita konfiguraciju systemd. Potrebno je izvesti naredbu svaki put kada je izmijenjena neka konfiguracijska datoteka jedinica. |
| **chkconfig --list** | systemctl list-unit-files | Popis svih konfiguriranih jedinica zajedno sa statusom. |
| **init <razina_izvođenja>** | systemctl isolate | Pokretanje jedinice zajedno sa svim jedinicama o kojima ta jedinica ovisi uz gašenje svih ostalih. |
| **NEMA** | systemctl list-units --type=<tip> | Prikaz svih jedinica određenoga tipa. |

Tablica 8 – Osnovne systemctl naredbe za upravljanje sustavom i servisima i njihovi ekvivalenti

Svi gornji primjeri koriste puno ime servisa (ime.service), ali naredba `runlevel` `systemctl` podržava korištenje bez sufiksa, na primjer „`systemctl start ime`“, a podržava i aliase. Za ispis svih aliasa koristi se mogućnost „`show ime –p Names`“, na primjer„`systemctl show nfs-server.service –p Names`“

**Systemd – zaustavljanje, gašenje i hibernacija sustava**

U donjoj tablici prikazane su standardne operacije te način njihova izvođenja u sustavu **systemd.**

| Stara naredba | Naredba u systemd | Objašnjenje |
|---|---|---|
| **halt** | systemctl halt | Zaustavljanje sustava. |
| **poweroff** | systemctl poweroff | Gašenje sustava. |
| **reboot** | systemctl reboot | Ponovno pokretanje sustava. |
| **pm-suspend** | systemctl suspend | Pauziranje sustava. |
| **pm–hibernate** | systemctl hibernate | Hibernacija sustava. |
| **pm-suspend-hybrid** | systemctl hybrid sleep | Pauziranje i hibernacija sustava. |

Tablica 9 – Standardne operacije upravljanja sustavom u sustavu systemd

Zadnje tri naredbe navedene u prvom stupcu nisu standardni dio distribucije, već se instaliraju s paketom **pm-utils** i služe pauziranju sustava, hibernaciji sustava ili pauziranju i hibernaciji sustava. Kod sustava **systemd** te su naredbe standardni dio paketa i ne treba ništa dodatno instalirati.

Ostale naredbe u prvom stupcu su i dalje dostupne, ali one pozivaju njihove systemctl ekvivalente iz drugog stupca. Prednost starih naredbi je što podržavaju neke dodatne mogućnosti koje njihovi **systemctl** ekvivalenti ne podržavaju. Primjerice, naredba `shutdown` podržava odgođeno isključivanje sustava sljedećim naredbama:

```
# shutdown --poweroff 15:35
# shutdown --halt +11
```

Prva naredba definira trenutak isključenja poslužitelja u nekom trenutku u budućnosti. Taj je trenutak definiran u 24-satnom formatu za 15 sati i 35 minuta. Druga naredba pokreće isključivanje sustava odgođeno za 11 minuta. Također je moguće otkazati isključivanje sustava naredbom `#shutdown -c` .

Više o mogućnostima ovih naredbi može se pronaći u **man** stranicama svake pojedine naredbe.

### 1.2.2. Prilagodba pokretanja operacijskog sustava Linux

Ranije korišteni **SysV init** i **Upstart** implementirali su predefinirane razine izvođenja (engl. *runlevel*) za određene načine rada sustava. Te razine izvođenja bile su numerirane od 0 do 6, a definirale su skup servisa koji se u svakoj od tih razina pokreće.

U systemd razine izvođenja zamijenile su jedinice meta (engl. *target units*) ime kojih završava s „.target“. Svrha tih jedinica meta jest grupiranje drugih systemd jedinica kroz lanac ovisnosti. Važno je napomenuti da unutar tih jedinica meta mogu biti i druge jedinice meta.

Standardno se **systemd** distribuira s jedinicama meta koje logički odgovaraju ranije implementiranim razinama izvođenja. Za poboljšanje kompatibilnosti postoje i aliasi koji se zovu isto kao ranije postojeće razine izvođenja. Primjerom su ispisane takve jedinice meta:

```
# systemctl list-unit-files|grep runlevel
systemd-update-utmp-runlevel.service       static
runlevel0.target                           disabled
runlevel1.target                           disabled
runlevel2.target                           static
runlevel3.target                           static
runlevel4.target                           static
runlevel5.target                           static
runlevel6.target                          disabled
```

Potrebno je još jednom napomenuti da su ovo aliasi, a da se jedinice meta izvorno drugačije zovu. U donjoj tablici prikazane su sve jedinice meta koje su ekvivalenti razinama izvođenja.

| Razina izvođenja | Jedinice meta | Objašnjenje |
|---|---|---|
| **0** | runlevel0.target, poweroff.target | Isključivanje i zaustavljanje sustava. |
| **1** | runlevel1.target, rescue.target | Pokretanje *rescue* ljuske. |
| **2** | runlevel2.target, multi-user.target | Pokretanje sustava u višekorisničkom modu bez GUI-ja. |
| **3** | runlevel3.target, multi-user.target | Pokretanje sustava u višekorisničkom modu bez GUI-ja. |
| **4** | runlevel4.target, multi-user.target | Pokretanje sustava u višekorisničkom modu bez GUI-ja. |
| **5** | runlevel5.target, graphical.target | Pokretanje sustava u višekorisničkom modu s GUI-jem. |

| | | |
|---|---|---|
| **6** | runlevel6.target, reboot.target | Isključivanje i ponovno pokretanje sustava. |

Tablica 10 – Mete koje su ekvivalenti razinama izvođenja

Ove jedinice meta moguće je prilagoditi svojim potrebama baš kao i razine izvođenja.

Još će neko vrijeme biti moguće koristiti naredbe `telinit` i `runlevel`, za prijelaz između razina izvođenja, ali ih je bolje izbjegavati jer će uskoro prestati biti dostupne.

Umjesto naredbe `runlevel` treba koristiti systemctl **list-units --type target.** Kao što je prikazano primjerom, ove dvije naredbe daju vrlo različit izlaz:

```
# runlevel
N 5
# systemctl list-units --type target
UNIT                   LOAD   ACTIVE SUB    DESCRIPTION
basic.target           loaded active active Basic System
cryptsetup.target      loaded active active Encrypted Volumes
getty.target           loaded active active Login Prompts
graphical.target       loaded active active Graphical Interface
local-fs-pre.target    loaded active active Local File Systems (Pre)
local-fs.target        loaded active active Local File Systems
multi-user.target      loaded active active Multi-User System
network-online.target loaded active active Network is Online
network.target         loaded active active Network
paths.target           loaded active active Paths
remote-fs-pre.target   loaded active active Remote File Systems (Pre)
remote-fs.target       loaded active active Remote File Systems
rpcbind.target         loaded active active RPC Port Mapper
slices.target          loaded active active Slices
sockets.target         loaded active active Sockets
sound.target           loaded active active Sound Card
swap.target            loaded active active Swap
sysinit.target         loaded active active System Initialization
timers.target          loaded active active Timers

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type

19 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
#
```

Za prelazak između razina izvođenja umjesto naredbe `telinit` treba koristiti **systemctl isolate** [name.target]. Na primjer, umjesto `telinit 5` može se koristiti **systemctl isolate runlevel5.target**. Obje naredbe će pokrenuti ekvivalent razine izvođenja 5.

Prilagodba sustava potrebama korisnika izvodi se ciljanim izmjenama meta koje se pokreću pri pokretanju sustava.

Izmjene sustava čine se daleko kompliciranijima za izvesti kod sustava **systemd** nego što su bile u slučaju **SysV init**. Naime, korištenje meta je hijerarhijski organizirano i zbog toga manje pregledno. Pogledajmo na primjeru kako to izgleda. Naredbom `systemctl get-default` dolazi se do naziva mete koja se pokreće u osnovnim postavkama:

```
$ systemctl get-default
graphical.target
```

Pregledom mete **graphical.target** dobije se popis servisa koji će biti pokrenuti u osnovnim postavkama.

```
$ cat /lib/systemd/system/graphical.target
[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
After=multi-user.target
Conflicts=rescue.target
Wants=display-manager.service
AllowIsolate=yes
```

Pojasnimo sadržaj datoteke. Za metu **graphical** treba (i pokreće se nakon) meta **multi-user**, u konfliktu je s **rescue** metom i želi **display-manager** metu. *Wants* direktiva je nešto manje striktni oblik *Requires* direktive i uzrokuje da se u njoj imenovana meta pokuša pokrenuti u paraleli s **graphical**, ali ako to nije moguće, **graphical** će se i dalje pokrenuti.

Da bi se vidjelo koji se sve servisi pokreću u osnovnim postavkama, potrebno je pogledati definiciju meta **multi-user** i **display-manager**, zatim pogledati koje oni mete pozivaju i tako dok se ne pristupi samo jednostavnim metama koje pozivaju određene servise. Konačni rezultat, odnosno popis svih servisa koji se aktiviraju u osnovnim postavkama može se dobiti naredbom:

```
$ systemctl list-unit-files --type=service|grep enabled
accounts-daemon.service                   enabled
anacron-resume.service                    enabled
anacron.service                           enabled
atd.service                               enabled
avahi-daemon.service                      enabled
bluetooth.service                         enabled
cron.service                              enabled
cups-browsed.service                      enabled
cups.service                              enabled
dbus-org.bluez.service                    enabled
dbus-org.freedesktop.Avahi.service        enabled
dbus-org.freedesktop.ModemManager1.service enabled
```

```
dbus-org.freedesktop.nm-dispatcher.service enabled
display-manager.service                    enabled
gdm.service                                enabled
getty@.service                             enabled
hwclock-save.service                       enabled
ModemManager.service                       enabled
NetworkManager-dispatcher.service          enabled
NetworkManager.service                     enabled
pppd-dns.service                           enabled
rsyslog.service                            enabled
ssh.service                                enabled
sshd.service                               enabled
syslog.service                             enabled
vboxadd-service.service                    enabled
vboxadd-x11.service                        enabled
vboxadd.service                            enabled
```

Za lociranje svih meta koje pokreću ove servise trebalo bi proučiti datoteke s postavkama približno jednakoga broja meta koliko je pokrenuto servisa. Ovako opsežan i zahtjevan postupak brzo bi osudio **systemd** na propast i zaborav. Ipak, u systemd je implementirana jednostavnija logika za sve standardne operacije poput izmjene zadanoga načina pokretanja, izmjene i dodavanja meta i servisa i slično. Neki standardni procesi opisani su u poglavlju 1.2.5.

### 1.2.3. Oporavak od pogrešaka

Kada se sustav ne uspijeva oporaviti od pogreške ni nakon ponovnog pokretanja, potrebno je izmijeniti način pokretanja. Izmjenom načina pokretanja privremeno se zaobilazi problem, a trajno rješenje kao i način na koji zaobići problem ovise o tipu uzroka problema. Izmjena načina pokretanja treba biti samo privremeno rješenje ili pokušaj isprobavanja pretpostavke jer će se pri idućem planiranom ili ne planiranom pokretanju sustava greška vratiti.

Probleme primarno razlikujemo ovisno o fazi pokretanja u kojoj se manifestiraju:

1. u radu programa za učitavanje sustava (engl. *boot loader*)
2. pri pokretanju/učitavanju jezgre
3. pri pokretanju servisa (systemd).

**Napomena**

Važno je napomenuti da što koji medij radi uvelike varira o distribuciji kojoj taj medij pripada i inačici distribucije. Tako neke distribucije nude mogućnosti za konfiguraciju postavki poput mapiranja tipkovnice ili mrežne konfiguracija, a druge konfiguriraju minimalne mogućnosti.

#### Oporavak od pogrešaka – u radu programa za učitavanje sustava

U slučajevima kada je program za učitavanje sustava krivo konfiguriran ili postoje pogreške u njegovu radu, potrebno je koristiti medij za pokretanje sustava. Mediji svih *Linux* distribucija mogu se koristiti za pokretanje sustava u *rescue* načinu rada. Također je važno napomenuti da se ti

mediji mogu koristiti na bilo kojem sustavu, a ne samo na sustavima gdje je instalirana ta *Linux* distribucija. Postupak korištenja medija za pokretanje sustava sastoji se od tri koraka:

1. Prvo se sustav pokrene s BIOS postavkama koje najprije učitavaju sustav s proizvoljnog medija CD/DVD/USB. Pri pokretanju se odabire „*rescue*“ mogućnost pokretanja.
2. Korijenski direktorij sustava često se automatski prepoznaje i postavlja se negdje u stablo direktorija inicijalnoga korijenskog direktorija (u „*rescue*“ modu taj je korijenski direktorij smješten u RAM).

Kada je korijenski direktorij sustava koji popravljamo *montiran* na neki direktorij (za primjer se koristi direktorij „/mnt/stari_root“), potrebno je naredbom `chroot` u trenutnoj ljusci promijeniti korijenski direktorij za trenutnu ljusku.

Primjer izvođenja naredbe `chroot` u danom scenariju:

```
# mount /dev/sda3 /mnt/stari_root
# ls /root/
# chroot /mnt/stari_root
# ls /root/
Desktop  Downloads
```

U gornjem primjeru se korijenski direktorij nalazi na /**dev/sda3**. Kao što je vidljivo, nakon izvođenja naredbe chroot mijenja se položaj korijenskoga direktorija tako da je sada dostupan direktorij *root* korisnika s njegovim sadržajem na odgovarajućem mjestu.

Sada je sve na sustavu dostupno i na očekivanom mjestu te je moguće iz zapisa u datotekama prepoznati grešku i ukloniti ju.

**Oporavak od pogrešaka – pri pokretanju/učitavanju jezgre**

Jedna od mogućih pogrešaka pri učitavanju jezgre jest nemogućnost lociranja korijenskoga direktorija na datotečnom sustavu. Ako nastane takva pogreška, često nije potrebno koristiti medij za pokretanje sustava. Dovoljno je pokrenuti sustav s drugim postavkama jezgre. Ta mogućnost ponekada **nije moguća**. Na primjer, u slučajevima kada je program za učitavanje sustava konfiguriran s pogrešnim parametrima za jezgru, a nije korištena initrd datoteka.

Potpuni oporavak od pogreške kada sustav ne uspijeva locirati korijenski datotečni sustav jest pokretanje sustava i ispravak parametara jezgre iz (evidentno) pogrešnih u točne parametre. Na primjer, pogrešni uređaj za boot particiju iz /**dev/sda1** promijeniti u /**dev/hda1**.

**Oporavak od pogrešaka – pri pokretanju servisa**

Pogreške nastale pri pokretanju servisa smanjile su se uvođenjem sustava **systemd**. Ipak, pogreške su i dalje moguće te je kao i ranije kod inita potrebno promijeniti razinu izvođenja, odnosno metu. Za tu svrhu koristi se mogućnost „**systemd.unit=<unit>“** pri pokretanju servisa.

Uz brojne druge dostupne mogućnosti koje pomažu lociranju pogreške, za uklanjanje/zaobilaženje pogreške važna je još i mogućnost „**systemd.confirm_spawn=yes/no**“ kojom se omogućuje da systemd za svaki pokrenuti proces traži potvrdu od korisnika prije pokretanja.

Ovdje je važno napomenuti da su rijetki servisi koji će onemogućiti pokretanje sustava. Kod većine servisa će se u slučaju nemogućnosti pokretanja servisa sustav uspješno pokrenuti te potom javiti na terminal i zapisati u datoteku da se servis nije uspio pokrenuti.

Samo u rijetkim slučajevima bit će potrebno koristiti **medij za pokretanje sustava** (CD/DVD proizvoljne distribucije u „*rescue*“ modu).

### 1.2.4. Prilagodba initramfs

**Inicijalni RAM datotečni sustav** (u daljnjem tekstu datoteka **initramfs**) sadržava prvi, ali i privremeni, korijenski direktorij koji je potreban za proces učitavanja jezgre. Datoteka **initramfs** je ovisna o jezgri i jezgra učitava datoteku **initramfs** u RAM kako bi se iskoristila za učitavanje svih potrebnih modula za učitavanje pravoga korijenskog direktorija. Ovaj je sustav nasljednik initrd.

Iako im je funkcija ista, postoje neke razlike između **initrd** i **initramfs** datoteka:

- značenje – **initrd** je „initial RAM disk“, a **initramfs** je „initial RAM file system“
- format – **initrd** je blok uređaj, a **initramfs** je zasnovan na datotekama
- kreiranje – kod **initrd** se kreira slika datotečnoga sustava, a **initramfs** je arhiva datoteka.

Važno je napomenuti da se korištenje datoteke **initramfs** mora omogućiti pri konfiguraciji jezgre. Pri konfiguraciji je moguće i definirati ime datoteke **initramfs**, ali preglednije i intuitivnije je koristiti zapis u konfiguraciji programa za pokretanje sustava.

Alati za izradu datoteke **initramfs** variraju ovisno o korištenoj *Linux* distribuciji. Dva su osnovna alata:

- mkinitrd (Red Hat i srodne distribucije) i
- mkinitramfs (Debian i srodne distribucije).

S obzirom na to da je fokus ovoga tečaja na distribuciju Debian, opisat će se naredba `mkinitramfs`. Primjer izvođenja naredbe je:

```
# mkinitramfs -o /boot/initrd.img-4.8.10-amd64b 4.8.10-amd64
```

Ova naredba kreira datoteku **initramfs /boot/initrd.img-4.8.10-amd64b** na osnovi postavki za inačicu jezgre **4.8.10-amd64**.

Bez zadnjeg parametra (4.8.10-amd64) naredba `#mkinitramfs -o /boot/initrd.img-4.8.10-amd64b` kreirala bi datoteku **initramfs** istog imena kao i prethodna naredba, ali bi se koristila trenutačno aktivna jezgra.

U Tablici 11 nabrojane su i objašnjene standardne mogućnosti naredbe `mkinitramfs` za izradu datoteke **initramfs**.

| Mogućnost | Objašnjenje |
|---|---|
| **-d <direktorij>** | Postavlja se konfiguracijski direktorij umjesto standardnog /etc/initramfs-tools. |
| **-k** | Zaustavlja automatsko uklanjanje privremenoga direktorija korištenoga pri kreiranju datoteke. |
| **-c <metoda_kompresije>** | Definiranje načina kompresije datoteke. |
| **-o <ime_datoteke>** | Postavlja ime datoteke **initramfs**. |
| **-r <root_particija>** | Definira na kojoj se particiji nalazi *root* direktorij |
| **-v** | Pokretanje naredbe s više informacija o izvođenju |
| **--supported-host-version=<inačica>** | Provjera mogućnosti kreiranja datoteke **initramfs** za zadanu inačicu aktivne jezgre |
| **--supported-target-version=<inačica>** | Provjera mogućnosti kreiranja datoteke **initramfs** za zadanu inačicu jezgre |
| **<verzija>** | Zadnja mogućnost naredbe koja definira za koju se inačicu kreira datoteka **initramfs**; ako nije zadana kreira se za trenutačno aktivnu jezgru |

Tablica 11 – Mogućnosti naredbe mkinitramfs

Naredbe za kreiranje *initramfs* datoteka su:

- `mkinitramfs` i
- `update-initramfs`.

Naredba `update-initramfs` koristi naredbu `mkinitramfs` koja je namijenjena za napredne korisnike.

Naredba `update-initramfs` čuva sažetak kreiranih **initramfs datoteka**, kreira njihove sigurnosne kopije i pri radu koristi postavke iz /**etc/initramfs-tools/update-initramfs.conf** konfiguracijske datoteke.

Konfiguracijska datoteka koja upravlja ponašanjem naredbe `mkinitramfs` je /**etc/initramfs-tools/initramfs.conf**.

Slijede primjeri izvođenja naredbe `update-initramfs`:

```
# update-initramfs -u -k all
update-initramfs: Generating /boot/initrd.img-3.16.0-5-amd64
update-initramfs: Generating /boot/initrd.img-3.16.0-4-amd64
update-initramfs: Generating /boot/initrd.img-3.2.0-4-amd64
```

U primjeru je prikazano kako se s `update-initramfs` mogu ažurirati postojeće datoteke ***initramfs***. Ova se naredba najčešće izvodi kada je nastala neka greška na sustavu ili kada se provode izmjene jezgre ili hardvera nakon kojih je potrebno prilagoditi datoteku ***initramfs***.

Datoteka **initramfs** potpomaže ključan korak u učitavanju jezgre i kao takva se ne može značajno prilagođavati. Također, sadržaj datoteke ne utječe na kasnije ponašanje sustava.
Za **initramfs** datoteku važno je da podržava sve module potrebne za uspješno učitavanje jezgre. Alati poput **mkinitramfs** postoje kako bi se moglo provesti *debugiranje* u slučaju greške, ali za prilagodbu sustava potrebama korisnika treba prilagoditi **systemd** koji upravlja pokretanjem servisa.

### 1.2.5. Prilagodba sustava systemd

**systemd – izrada i izmjena systemd datoteka jedinica**

Datoteke jedinica smještaju se po instalaciji u direktoriju /**lib/systemd/system**. Datoteke u tom direktoriju ne trebaju se mijenjati ni od koga, uključujući administratora sustava.

Datoteke u direktoriju /**etc/systemd/system** imaju prednost nad datotekama u direktoriju /**lib/systemd/system**. Izmjene nad izvornim jedinicama trebaju se provoditi tako da se u direktoriju /**etc/systemd/system** kreiraju nove istoimene datoteke jedinica (engl. *unit files*) koji će nadglasati sadržaj datoteka u direktoriju /**lib/systemd/system**. Korištenje direktorija /**etc/systemd/system** je najsigurniji način za provođenje izmjena nad izvornim datotekama, budući da u sustavu systemd sadržaj toga direktorija ima prioritet nad svim drugim direktorijima. Ako su izmjene koje želimo provesti minimalne, uvijek je moguće jednostavno kopirati datoteku iz /**lib/systemd/system** i nju izmijeniti.

Sve novo-kreirane jedinice, neovisno o njihovu tipu, smještaju se u direktorij /**etc/systemd/system**/. Format imena datoteke je **<ime>.<tip>**, gdje je ime referenca koja će se kasnije koristiti za operacije nad jedinicom, a tip je jedna od standardnih vrijednosti navedenih u donjoj tablici.

| Tip/sufiks datoteke | Objašnjenje |
|---|---|
| **.service** | Servis. |
| **.target** | Meta – Skup jedinica. |
| **.automount** | Jedinica za automatsko montiranje uređaja. |
| **.device** | Datoteka uređaja upravljanog od jezgre. |
| **.mount** | Opis jednoga montiranja sustava. |

| | |
|---|---|
| **.path** | Datoteka ili direktorij u datotečnom sustavu. |
| **.scope** | Eksterno kreirani proces. |
| **.slice** | Skupina hijerarhijski organiziranih jedinica koje upravljaju sistemskim procesima. |
| **.snapshot** | Stanje sustava pohranjeno od systemd upravljačkoga programa. |
| **.socket** | Utičnica za komunikaciju između procesa. |
| **.swap** | Uređaj ili datoteka priručne memorije. |
| **.timer** | Systemd vremenski brojač. |

Tablica 12 – Tipovi jedinica u sustavu systemd

Ovisno o tipu jedinice variraju postavke koje mogu odnosno moraju biti u datotekama jedinica.

Za određeni servis mogu postojati jedinice više tipova. Tako, primjerice, za **ssh** postoji **servis** i **utičnica**. Navedeno je prikazano u donjem primjeru, gdje tražimo sve jedinice imena „sshd“:

```
# ls -R /etc/systemd/system /lib/systemd/system|grep ssh
sshd.service ssh.socket
```

Za potrebe konfiguracije pojedine jedinice mogu se koristiti brojne definicije, a broj i važnost tih definicija varira među različitim tipovima jedinica. Standardno za minimalnu potpunu definiciju jedinice potrebna su tri konfiguracijska segmenta:

1. **[Unit]** – sastoji se od generičkih mogućnosti koje su neovisne o tipu jedinice.
2. segment ovisan o tipu **[Service]** ili **[Socket]** ili… – sastoji se od direktiva specifičnih za upravo taj tip jedinice. Oznaka odgovara tipu jedinice. Tako će za servise taj segment biti označen sa [Service].
3. **[Install]** – informacije o instalaciji jedinice koji su važni naredbama **systemctl enable** i **systemctl disable**.

Ove su definicije izdvojene u donjim tablicama: tablica 13 sadrži ključne definicije potrebne za sve jedinice, a tablica 14 najvažnije definicije očekivane u konfiguraciji jedinice servisa. Tablica 15 prikazuje važne instalacijske mogućnosti.

| Definicija (za jedinice) | Objašnjenje |
|---|---|
| **Description** | Opis jedinice. |
| **Documentation** | Popis URL-ova s dokumentacijom jedinice. |
| **After** | Strogo definira da se jedinica ne može pokrenuti dok nisu aktivne jedinice navedene u tom popisu. Pokretanje ove jedinice neće pokrenuti jedinice u „After“ popisu. Suprotno od „Before“. |
| **Requires** | Definira jedinice koje će se pokrenuti pri pokretanju jedinice u čijem su popisu. Kada se neka od |

| | |
|---|---|
| | navedenih jedinica ne uspije pokrenuti, ne pokreće se ni ova jedinica. |
| **Wants** | Slabija ovisnost od prethodne dvije. Može se tumačiti kao poželjno i nema izravni utjecaj na pokretanje jedinica. |
| **Conflicts** | Suprotnost od „Requires“. Ako je neka od jedinica u ovom popisu aktivna, tada je nemoguće pokrenuti ovu jedinicu. |
| **Before** | Strogo definira da se jedinice navedene u ovom popisu ne pokreću dok ne završi pokretanje ove jedinice. |

Tablica 13 – Definicije koje su dio svih jedinica

| Definicija (za datoteke) | Objašnjenje |
|---|---|
| Type | Definira način pokretanja i time utječe na funkcionalnosti **ExecStart** i s njime povezane mogućnosti. Moguće vrijednosti su: ***simple***, ***forking***, ***oneshot***, ***dbus***, ***notify*** i ***idle***. |
| ExecStart | Definira naredbu ili skripte koje se izvršavaju kada se jedinica pokreće. Moguće je dodatno definirati **ExecStartPre** i **ExecStartPost** za dodatne naredbe koje će se izvršiti prije i nakon pokretanja jedinice respaktivno. |
| ExecStop | Definira naredbu ili skripte koje se izvršavaju kada se jedinica zaustavlja. |
| ExecReload | Definira naredbu ili skripte koje se izvršavaju kada se jedinica ponovno pokreće. |
| Restart | Mogućnost koja definira treba li se jedinica automatski ponovno pokrenuti u slučaju nestandardnoga prekida rada. |
| RemainAfterExit | Vrijednost *true* ove mogućnosti definira da će jedinica biti smatrana aktivnom čak i nakon što izvršavanje svih njezinih procesa završi. |

Tablica 14- Definicije koje su standardne za definiciju servisa

| Mogućnost | Objašnjenje |
|---|---|
| Alias | Popis alternativnih imena jedinice. |
| RequiredBy | Popis jedinica koje ovise o toj jedinici. |
| WantedBy | Popis jedinica za čije je pokretanje poželjno da je ova jedinica pokrenuta. |
| Also | Popis jedinica koje trebaju biti instalirane i deinstalirane zajedno s ovom jedinicom. |

Tablica 15 – Važne instalacijske mogućnosti jedinica

Sva ova tri segmenta nalaze se u definiciji **rsyslog** servisa za upravljanje zapisima o sistemskim događajima. Na primjeru su prikazana gore navedena tri segmenta:

```
root@l201:/lib/systemd/system# cat
/lib/systemd/system/rsyslog.service

[Unit]
Description=System Logging Service
Requires=syslog.socket
Documentation=man:rsyslogd(8)
Documentation=http://www.rsyslog.com/doc/

[Service]
Type=notify
ExecStart=/usr/sbin/rsyslogd -n
StandardOutput=null
Restart=on-failure

[Install]
WantedBy=multi-user.target
Alias=syslog.service
```

Najprije se može uočiti da je središnji segment označen sa [Service] jer je u pitanju servis. Prvi segment daje kratki opis jedinice (**Description**), popis jedinica koje moraju biti aktivne za pokretanje ove jedinice (**Requires**) i popis dokumentacije (**Documentation**).

Drugi segment definira način pokretanja (**Type**), naredbu kojom se servis pokreće (**ExecStart**), kamo se šalje standardni izlaz (**StandardOutput**) i treba li se (i u ovom slučaju kada) servis ponovno pokrenuti (**Restart**).

Treći segment definira koja jedinica treba ovu jedinicu (**WantedBy**) i jedno alternativno ime jedinice (**Alias**).

Nadalje će biti prikazano kako promijeniti ovu postojeću jedinicu. Prvi korak je kopiranje postojeće konfiguracijske datoteku u **direktorij /etc/systemd/system**:

```
# cp /lib/systemd/system/rsyslog.service /etc/systemd/system/
```

Za sada nije ništa izmijenjeno. Zbog većeg prioriteta direktorija /**etc/systemd/system** od sada će biti korištena nova datoteka za upravljanje servisom, ali kako su datoteke istoga sadržaja, neće biti promjena. Potrebno je dodati naredbu `wall` koja svim korisnicima javlja poruku o gašenju servisa. Proizvoljnim uređivačem teksta u /**etc/systemd/system/rsyslog.service** datoteku dodaje se linija u središnji segment:

```
ExecStop=/usr/bin/wall "rsyslog je otišao u krevet. Nitko ne bilježi što radite! Sigurnosni rizik visok!"
```

Ova izmjena još nije aktivna. Tek će se nakon izvršavanja naredbe `systemctl daemon-reload` aktivirati izmjene. Ako se sada izvrše te dvije naredbe, na ekranu će se pojaviti definirana poruka, kao što je vidljivo u primjeru:

```
# systemctl daemon-reload
# systemctl stop rsyslog syslog.socket

Broadcast message from root@l201 (somewhere) (Sat Jan  7 16:32:53 2017):
rsyslog je otišao u krevet. Nitko ne bilježi što radite! Sigurnosni rizik visok!
```

**systemd – segmentacija zapisa**

Kako zapisi kod jedinica s kompleksnim pravilima ne bi bili nepregledni u ogromnim datotekama, uz datoteke za definiciju pojedine jedinice moguće je koristiti direktorije s konfiguracijskim datotekama. Za **sshd.service** bi se tako kreirao direktorij **sshd.service.d/** u istom direktoriju.

Također je moguće kreirati direktorije za pojedine ovisnosti s drugim jedinicama. Tako je, primjerice, moguće kreirati direktorije **sshd.service.wants/** i **sshd.service.requires/.** U te direktorije se smještaju simboličke poveznice na jedinice koje su poželjne za pokretanje i nužne za pokretanje sshd servisa respektivno. Simboličke se poveznice mogu kreirati pri instalaciji sshd, pri pokretanju ili ručno dodati. U donjem je primjeru vidljivo da za metu socket.target postoji čak 7 poželjnih jedinica:

```
root@l201:/lib/systemd/system# ls ./sockets.target.wants/ -l
lrwxrwxrwx  1 root root    14 Aug  2  2015 dbus.socket -> ../dbus.socket
lrwxrwxrwx  1 root root    25 Sep  4 00:35 systemd-initctl.socket -> ../systemd-initctl.socket
lrwxrwxrwx  1 root root    34 Sep  4 00:35 systemd-journald-dev-log.socket -> ../systemd-journald-dev-log.socket
lrwxrwxrwx  1 root root    26 Sep  4 00:35 systemd-journald.socket -> ../systemd-journald.socket
lrwxrwxrwx  1 root root    27 Sep  4 00:35 systemd-shutdownd.socket -> ../systemd-shutdownd.socket
lrwxrwxrwx  1 root root    31 Sep  4 00:35 systemd-udevd-control.socket -> ../systemd-udevd-control.socket
lrwxrwxrwx  1 root root    30 Sep  4 00:35 systemd-udevd-kernel.socket -> ../systemd-udevd-kernel.socket
```

Zaključno, systemd je novo „srce“ ovog operacijskog sustava. Korisnik koji dobro razumije systemd i jezgru napravio je veliki korak prema razumijevanju *Linuxa*.

## 1.3. Vježba: Jezgra operacijskog sustava Linux

1. Prije početka rada odaberite sliku stanja virtualnoga računala (engl. Snapshot) za početak ove vježbe.
2. Prijavite se na računalo kao korisnik **l201**. U GUI-ju pokrenite **Terminal (Activities → Terminal)**. Izvršite su - naredbu da postanete administrator (lozinka: L201).
3. Instalirajte pakete potrebne za izradu prilagođene inačice jezgre. (`apt install wget build-essential dwarves python3 libncurses-dev flex bison libssl-dev bc libelf-dev -y`)

---

4. Uđite u direktorij **/usr/local/.**
5. Pribavite izvorni kôd jezgre (linux) pomoću naredbe **wget** sa kernel.org stranice . (`# wget https://mirrors.edge.kernel.org/pub/linux/kernel/v5.x/linux-5.17.9.tar.gz`)

---

6. Naredbom **tar** otpakirajte pribavljenu jezgru.
7. Uđite u kreirani poddirektorij i pokrenite naredbu

   `# cp -v /boot/config-$(uname -r) .config` te zatim izradu konfiguracije sa `# make oldconfig`.
8. Pokrenite izradu slike jezgre naredbom `# make bzImage`

---

   Nakon nekog vremena, zaustavite konfiguraciju – [CTRL]+[c].
9. Sljedeći bi koraci programskoga prevođenja jezgre trajali predugo, pa je u skrivenom direktoriju već pripremljena jezgra: **/usr/local/src/.skriven**; Uđite u taj direktorij.

---

   Provjerite koje se datoteke jezgre nalaze u direktoriju:

---

10. Instalirajte tu jezgru.

---

11. Naredbom `#update-grub2` dodajte novu jezgru u grub.
12. Pokrenite računalo i provjerite da li je nova jezgra dostupna?

---

13. Pokrenite računalo s tom jezgrom? Je li sustav uspješno pokrenut?

## 1.4. Vježba: Pokretanje operacijskog sustava Linux – izmjena postojeće jedinice

2. Prije početka rada odaberite sliku stanja virtualnoga računala (engl. Snapshot) za početak ove vježbe.
3. Prijavite se na računalo kao korisnik **l201**. U GUI-ju pokrenite **Terminal** (**Activities** → **Terminal**). Izvršite su - naredbu da postanete administrator (lozinka: L201).
4. Kopirajte datoteku **/etc/ssh/sshd_config** u **/etc/ssh/sshd_second_config.**

5. U datoteci **/etc/ssh/sshd-second_config** obrišite liniju „**#Port 22**“ i dodajte liniju „**Port 2222**“.
6. Kopirajte datoteku **/etc/systemd/system/sshd.service** u **/etc/systemd/system** direktorij i preimenujte datoteku u **sshd- second.service**.
7. Uredite datoteku **/etc/systemd/system/sshd-second.service** tako da postojeće linije zamijenite sljedećim linijama:

```
Description=OpenBSD Secure Shell server drugi  daemon
After=network.target auditd.service sshd.service
ExecStart=/usr/sbin/sshd -D -f /etc/ssh/sshd-second_config $SSHD_OPTS
Alias=sshd-second.service
```

8. Naredbom systemctl pokrenite novostvoreni servis. Namjestite da se servis pokreće pri pokretanju računala.

9. Provjerite da je sshd-second pokrenut kao i sshd.

10. Pokušajte se spojiti na lokalno računalo sa **ssh** naredbom – spojite se na standardno **ssh port** i na port **2222** (na kraj naredbe dodajte `-p 2222`).

## 1.5. Vježba: Pokretanje operacijskog sustava Linux – stvaranje nove jedinice

1. Prije početka rada odaberite sliku stanja virtualnoga računala (engl. Snapshot) za početak ove vježbe.
2. Prijavite se na računalo kao korisnik **l201**. U GUI-ju pokrenite **Terminal (Activities → Terminal)**. Izvršite su - naredbu da postanete administrator (lozinka: L201).
3. Stvorite novu datoteku **/etc/systemd/system/emacs.service.**

4. Promijenite prava pristupa te datoteke na 755 **chmod** naredbom:
5. U datoteku dodajte:

```
[Unit]
Description=Emacs text editor Documentation=info:emacs man:emacs(1)
https://gnu.org/software/emacs/
[Service]
Type=notify ExecStart=/usr/bin/emacs --fg-daemon ExecStop=/usr/bin/emacsclient -
-eval "(kill-emacs)" Restart=on-failure [Install] WantedBy=default.target
```

6. Izvršite naredbu za učitavanje novih postavki postojećih jedinica i učitavanje novih jedinica u systemd.

7. Pokrenite emacs jedinicu servisa.

8. Namjestite emacs jedinicu servisa za pokretanje pri pokretanju računala.

9. Provjerite stanje emacs servisa.

   Pokrenut (zaokružite odgovor): DA / NE

   Namješten za pokretanje računalom : DA / NE

   Ime servisa i broj procesa:

10. Ponovno pokrenite sustav pomoću systemctl naredbe.

---

11. Provjerite stanje emacs servisa.

    Pokrenut: DA/NE

    Namješten za pokretanje računalom : DA / NE

    Ime servisa i broj procesa:

# 2. Datotečni sustavi

**Trajanje poglavlja:**
**225 min**

Po završetku ove cjeline moći ćete:

- razlikovati tipove datotečnih sustava na operacijskom sustavu Linux
- stvoriti i prilagoditi datotečni sustav
- provjeriti stanje datotečnog sustava i upravljati programima za oporavak
- namjestiti automatsko montiranje datotečnog sustava

U ovoj se cjelini obrađuju datotečni sustavi operacijskog sustava *Linux*. U cjelini je opisan postupak stvaranja i prilagodbe datotečnih sustava. Cjelina obrađuje i testiranje rada i akcije koje su potrebne u slučaju kvarova datotečnog sustava. Cjelina završava sa opisom konfiguracije za automatsko montiranje datotečnih sustava.

## 2.1. Upravljanje datotečnim sustavima

### 2.1.1. Vrste datotečnih sustava

*Linux* pruža podršku za cijeli niz tipova datotečnih sustava. Podrška postoji i za datotečne sustave koji su primarno razvijani za druge operacijske sustave poput **Windowsa** ili **Dos**. Neki od tih datotečnih sustava ne rade dobro, a oni koji dobro rade često ne podržavaju ni približno sve funkcionalnosti koje *Linux* očekuje od datotečnog sustava. Zbog toga pri planiranju i izradi Linux sustava poželjno je korištenje *Linux*u standardnih datotečnih sustava. Takvi standardni datotečni sustavi su:

- **Ext2fs**, oznaka „ext2“ - *Second Extended File System* (kraće *ext2fs* ili *ext2*) je izvorni standardni *Linux* datotečni sustav. Bio je dominantan u kasnim 1990-ima, ali danas je zastario i zamijenjen je novim datotečnim sustavima. Ipak, jednostavnost koja dolazi iz činjenice da ne koristi journaling čini ga i dalje korisnim za malene uređaje, na kojima se i dalje koristi. Dodatno se i dalje zna koristiti na sustavima gdje čitav uređaj koristi aplikacija koja sama brine o sigurnosnim kopijama u slučaju ispada.
- **Ext3fs**, oznaka „ext3“ - *Third Extended File System* (krače *ext3fs* ili *ext3*) je zapravo ext2 s dodanim journalingom. Rezultat takvog spajanja je datotečni sustav koji je pouzdan i brz, te se brzo oporavlja od ispada sustava.
- **Ext4fs**, oznaka „ext4“ - *The Fourth Extended File System* je zadnja generacija ext datotečnih sustava. Najznačajnije nadogradnje u odnosu na ext3 su mogućnost kreiranja velikih datotečnih sustava (preko 32 tebibyta) i rada sa velikim datotekama (preko 32 tebibyta). Također, ext4 pruža poboljšanje performansa u odnosu na ext3.
- **JFS**, oznaka „jfs“ - *Journaled File System* razvio je IBM za njihov AIX OS i kasnije nadogradio za OS/2. OS/2 verzija je donirana *Linux*-u. Posebnost JFS je vrlo napredan

journaling, ali datotečni je sustav ipak prilagođen potrebama operacijskih sustava za koje je razvijan.
- **XFS**, oznaka „xfs“ - *Extents File System* je datotečni sustav koji je razvio *Silicon Graphics* (SGI) za IRIX OS. XFS je tehnički sofisticiran datotečni sustav koji je robustan, brz i fleksibilan. Nažalost, na *Linux*u nisu podržane sve mogućnosti koje XFS-u daju njegove karakteristike.
- **ReiserFS**, oznaka „reiserfs“ - iz temelja razvijani datotečni sustav koji je od početka razvijan sa ciljem razvoja efikasnog *Linux* journaling datotečnog sustava. Posebno je dobar za rad sa velikim brojem malih datoteka. Danas zastario, a glavni nasljednik mu je **reiser4**.

Ovo su standardni datotečni sustavi koji se koriste na *Linux* operacijskom sustavu. Osim njih često su korišteni i:

- **FAT** – zastarjeli datotečni sustav koji je jedini podržan na starim verzijama Windowsa i na DOS-u i zbog toga se često koristi na prijenosnim medijima te je podržan od svih operacijskih sustava.
- **NTFS** – novija verzija datotečnog sustava za Windowse
- **ISO-9660** – Standardni datotečni sustav za optičke medije, korišten u vrijeme kada su prevladavali CD/ROM optički mediji.
- **UDF** (*Universal Disc Format*) je nova generacija datotečnog sustava za optičke medije koji se standardno koristi za DVD-ROM optičke medije i optičke medije za snimanje.

**Napomena**

Journaling je funkcionalnost datotečnog sustava da prije svake akcije prvo zapiše akciju koja se odvija. Tako će u slučaju bilo kakvog ispada biti moguće rekonstruirati sve akcije koje se nisu zbog ispada uspješno završile. Ovakva funkcionalnost naravno donosi dodatne troškove u potrošnji prostora na datotečnom sustavu i potrošnji vremena za svaku operaciju.

Uz navedene datotečne sustave postoje još i datotečni sustavi posebnog tipa. Primjerice, datotečni sustavi za mrežni pristup, poput ***Server Message Block/Common Internet Filesystem*** (SMB/CIFS) i ***Network Filesystem*** (NFS), koji omogućavaju jednom računalu da pristupi diskovima drugog udaljenog računala. Također, važno je spomenuti i **virtualne datotečne sustave** koji omogućavaju jezgri da uređajima pristupa na isti način kao i datotekama. ***udev*** **datotečni sustav** je primjer virtualnog datotečnog sustava.

### 2.1.2. Izrada datotečnih sustava

Tri su koraka standardna da bi se diskovni uređaj mogao početi koristiti:

1. Kreiranje particije (opcionalno)
2. Izrada datotečnog sustava
3. Montiranje (engl. *mount*)

Svi se ovi koraci obavljaju i pri instalaciji operacijskog sustava *Linux*, ali u ovom će se slučaju obrađivati kako se navedeni koraci provode na već instaliranom, aktivnom sustavu.

Svaka particija sastoji se od neprekinutog prostora na disku. Zbog toga je pri kreiranju particije dovoljno imenovati prvi i posljednji blok, budući da se time jednoznačno definira particija.

### Kreiranje particije

Prvo treba napomenuti da particija ne mora biti kreirana. Datotečni sustav može se kreirati nad cijelim blok uređajem.

Instalirana distrubucija *Linux*a dolazi s nekoliko alata za particioniranje diskova. Najčešće korišteni su alati:

- **fdisk** - najrašireniji i najčešće korišten alat, ali dopušta samo particije do 2 TB
- **parted** - nudi više mogućnosti od fdisk-a, kao što je promjena veličine particije te također dopušta particije do 9.4 ZB (ziliona bajtova, ili $10^{21}$).

Oba se alata pokreću pozivom istoimene naredbe sa dodatnim parametrima nad željenim diskom. Pregled postojećih particija na sustavu moguće je dobiti izvođenjem naredbi sa mogućnosti –l. Slijedi primjer izvođenja tih dviju naredbi:

```
# fdisk -l
Disk /dev/sdb: 2 GiB, 2147483648 bytes, 4194304 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk /dev/sda: 12 GiB, 12884901888 bytes, 25165824 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x386a3d63

Device     Boot   Start      End  Sectors  Size Id Type
/dev/sda1  *       2048   585727   583680  285M 83 Linux
/dev/sda2        585728  6445055  5859328  2.8G 82 Linux swap / Solaris
/dev/sda3       6445056 25163775 18718720  8.9G 83 Linux

# parted -l

Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 12.9GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system     Flags
 1      1049kB  300MB   299MB   primary  ext4            boot
 2      300MB   3300MB  3000MB  primary  linux-swap(v1)
 3      3300MB  12.9GB  9584MB  primary  ext4

Error: /dev/sdb: unrecognised disk label
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 2147MB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
```

```
Warning: Unable to open /dev/sr0 read-write (Read-only file system).  /dev/sr0
has been opened read-only.

Error: /dev/sr0: unrecognised disk label
Model: VBOX CD-ROM (scsi)
Disk /dev/sr0: 59.3MB
Sector size (logical/physical): 2048B/2048B
Partition Table: unknown
Disk Flags:
```

Budući da je **parted** u usporedbi s **fdisk** alatom napredniji te ima dodatne mogućnosti, isti će biti predstavljen u ovom tečaju. Najprije je potrebno kroz niz koraka prikazati standardne operacije nad diskom pomoću alata **parted**.

Prvo se naredbom `parted` ulazi u komandno linijsko sučelje alata:

```
# parted
GNU Parted 3.2
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands
```

Može se uočiti **prompt „(parted)“**, ali i da je u nedostatku parametra parted odabrao prvi dostupni disk - /dev/sda. Budući da su na /dev/sda sistemske particije („/“, „/boot“ i swap) i da postoji dodatni disk /dev/sdb, taj disk se može odabrati naredbom `select`.

```
(parted) select /dev/sdb
Using /dev/sdb
```

Naredbom `print` prikazuje se particijska tablica:

```
(parted) print
Error: /dev/sdb: unrecognised disk label
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 2147MB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
```

Kao što je vidljivo, disk je veličine 2GB i trenutno je prazan - bez particijske tablice. Ovom disku može se dodijeliti oznaka **msdos** naredbom `mklabel`. **msdos** je podrazumijevana vrsta particijske tablice za diskove do 2 TB koji koriste **Master Boot Record (MBR)**.

Dva su standarda za strukturu particija **Master Boot Record (MBR)** i **GUID Partition Table (GPT)**. **MBR** je stariji standard koji je ograničen jer podržava samo diskove veličine do 2 TB. **MBR** podržava maksimalno četiri primarne particije. Također GPT svakoj particiji dodjeljuje globalno jedinstveni identifikacijski broj (engl. globaly unique identifier GUID) koji je dovoljno velik da je izvjesno jedinstven globalno za svaku particiju u upotrebi.

```
(parted) mklabel msdos
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 2147MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start  End  Size  Type  File system
Flags
```

Konačni rezultat jest kreirana nova particija na disku:

```
(parted) mkpart
Partition type?  primary/extended? primary
File system type?  [ext2]? ext2
Start? 1
End? 100
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 2147MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start  End  Size  Type  File system  Flags
 1      1049kB  99.6MB  98.6MB  primary  ext2  lba
```

Prvo se naredbom `mkpart` pokreće postupak kreiranja nove particije:

1. Odabir particije: *primary* ili *extended*. U primjeru je odabran **primary**.
2. Nakon toga odabran je **tip – ext2**
3. Konačno dolazi izbor veličine diska sa izborom prvog i zadnjeg sektora. Izbor 100 sektora rezultira diskom veličine 100MB.

Kao što je vidljivo na kraju primjera, pri ponovnom izvršavanju **naredbe print** na disku se nalazi jedna particija veličine ~100MB. Ovim je kreirana particija i sve što je još potrebno napraviti je napustiti **parted** naredbom `quit`.

### Izrada datotečnog sustava

Izradom particije napravljena je priprema za izradu datotečnog sustava. Završni korak je na željenoj particiji izraditi datotečni sustav naredbom `mkfs`. `mkfs` je samo sučelje za niz alata za kreiranje pojedinih datotečnih sustava. Prikazano je na primjeru:

```
# mkfs -t ext4 /dev/sdb1
mke2fs 1.42.12 (29-Aug-2014)
```

```
Creating filesystem with 96256 1k blocks and 24096 inodes
Filesystem UUID: 8dc3a757-46ee-4c82-b9a6-6385cd3f515d
Superblock backups stored on blocks:
    8193, 24577, 40961, 57345, 73729

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

Naredba u gornjem primjeru poziva naredbu `mke2fs` koja je zajednička za kreiranje ext2/ext3/ext4 datotečnih sustava. Ono što naredba `mkfs` radi jest da na osnovu **mogućnosti –t** kojom se definira tip datotečnog sustava poziva odgovarajuću naredbu za kreiranje tog tipa. Naredbe koje `mkfs` poziva mogu biti smještene u /**sbin**, /**sbin/fs**, /**sbin/fs.d,** /**etc/fs** ili /**etc** direktoriju. Koliko je alata, a time i datotečnih sustava dostupno, ovisi o distribuciji i instaliranoj podršci.

**Montiranje datotečnog sustava**

Izraz koji se standardno koristi za pridjeljivanje datotečnog sustava određenom direktoriju je **montiranje particije**, a dolazi od engleske riječi *mount* i istoimene naredbe `mount` kojom se ta akcija provodi.

Kada *Linux* montira datotečni sustav, podaci o tom montiranju smještaju se u datoteku /**etc/mtab**. Iako je format sličan datoteci /**etc/fstab** (koja će biti kasnije objašnjena), ovu datoteku ne mijenjaju korisnici. Ova datoteka i datoteka /proc/mounts koriste se za bilježenje podataka o montiranim uređajima. Do inačice jezgre 2.6.26. u datoteci /**proc/mounts** se nalazilo malo podataka, ali u novijim jezgrama u toj se datoteci nalazi i više zapisa nego u /**etc/mtab**.

Naredba `df` je brži i standardniji način za pribavljanje podataka o montiranim particijama. Uz ispis montiranih particija naredba daje i informaciju o ukupnoj veličini i iskorištenosti montiranih uređaja. Primjer izvođenja naredbe `df`:

```
# df
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/sda3 9081336 6551312 2045672  77% /
udev        10240       0   10240   0% /dev
tmpfs      310512    4932  305580   2% /run
tmpfs      776276     504  775772   1% /dev/shm
tmpfs        5120       4    5116   1% /run/lock
tmpfs      776276       0  776276   0% /sys/fs/cgroup
/dev/sda1  274407   39594  216125  16% /boot
tmpfs      155256       8  155248   1% /run/user/118
tmpfs      155256       8  155248   1% /run/user/1000
/dev/sr0    57870   57870       0 100% /media/cdrom0
```

Naredba `mount` daje samo pregled montiranih uređaja, ako joj se ne zadaju dodatni parametri. Rezultat izvođenje naredbe odgovara sadržaju /**proc/mounts** datoteke.

U praksi je montiranje datotečnog sustava s naredbom `mount` jednostavno. Potrebno je samo imenovati datotečni sustav i direktorij u koji se taj datotečni sustav montira. To je u ranijim primjerima nekoliko puta napravljeno ovom naredbom:

```
# mount /dev/sdb1 /tmp/mount/
```

Kako bi ova naredba bila uspješno provedena potrebno je da se na /**dev/sdb1** nalazi datotečni sustav podržan od strane jezgre i da direktorij /**tmp/mount** postoji. Ovom se naredbom montira datotečni sustav jednom i moguće ga je koristiti.

U donjoj tablici su navedene češće korištene mogućnosti naredbe `mount`:

| Mogućnosti (kratka/duga varijanta) | Objašnjenje |
|---|---|
| **-a / --all** | Naredba za montiranje svih uređaja navedenih u /etc/fstab. |
| **-r / --read-only** | Ograničenje pristupa sustavu da može samo čitati sa montiranog uređaja. |
| **-w / --rw** | Definira da se po montiranom uređaju može pisati i čitati. Zadana vrijednost u većini distribucija. |
| **-o<mogućnosti>/--options<mogućnosti>** | Definicija dodatnih mogućnosti za montiranje (mogućnosti variraju između različitih datotečnih sustava). |
| **-L<oznaka>** | Definiranje uređaja na osnovi oznake. |
| **-U<uuid>** | Definiranje uređaja na osnovi univerzalnog jedinstvenog identifikatora. |

Tablica 16 - Mogućnosti naredbe mount

### 2.1.3. Prilagodba datotečnih sustava

Nakon što se naredbom izradio datotečni sustav, moguće ga je početi koristiti, ali se on još uvijek može prilagoditi, npr. zbog potrebe za većim datotečnim sustavom ili proširenjem aktivnog datotečnog sustava koji se već koristi.

Prvi je korak proširiti particiju pomoću **parted** alata. U parted se koristi naredba `resizepart`. Korištenje naredbe prikazano je u primjeru:

```
# parted /dev/sdb
GNU Parted 3.2
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: ATA VBOX HARDDISK (scsi)
```

```
Disk /dev/sdb: 2147MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  99.6MB  98.6MB  primary  ext4

(parted) resizepart
Partition number? 1
End?  [99.6MB]? 345
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 2147MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End    Size   Type     File system  Flags
1      1049kB  345MB  344MB  primary  ext4
```

U gornjem primjeru prvo je sa parted napravljen ispis particija kako bi se saznao broj particije kojoj se želi promijeniti veličina. Može se uočiti da je sada tip datotečnog sustava **ext4** (definirano ranije pomoću mkfs naredbe). Zatim se poziva naredba `resizepart` nad particijom s brojem 1. Za veličinu particije postavljen je kraj na 345MB. Treba napomenuti da je izlaskom iz parted promijenjena samo veličina particije, ali ne i datotečnog sustava. Ako bi se u ovom trenutku koristio datotečni sustav, on bi i dalje imao veličinu od ~100MB. Ovo se može prikazati primjerom:

```
# mkdir /tmp/mount
# mount /dev/sdb1 /tmp/mount/
# df -h |grep sdb1
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb1        88M  1.6M   79M   2% /tmp/mount
```

Da bi se moglo koristiti proširenje particije potrebno je proširiti i datotečni sustav. Naredba za promjenu veličine datotečnog sustava je `resize2fs`. Ova se naredba može izvršiti na aktivnom datotečnom sustavu ako je u pitanju proširenje datotečnog sustava. U nastavku je primjer korištenja naredbe:

```
# resize2fs /dev/sdb1
resize2fs 1.42.12 (29-Aug-2014)
Filesystem at /dev/sdb1 is mounted on /tmp/mount; on-line resizing
required
old_desc_blocks = 1, new_desc_blocks = 2
The filesystem on /dev/sdb1 is now 335888 (1k) blocks long.

# df -h |grep sdb1
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb1       314M  2.1M  296M   1% /tmp/mount
```

Kao što je vidljivo u primjeru, datotečni sustav može biti aktivan dok se radi proširenje. Kada se radi smanjivanje datotečnog sustava mogu nastati greške i potrebno je da datotečni sustav ne bude montiran. Pokušaj smanjivanja pomoću naredbe `resize2fs` javit će poruku o grešci. U sljedećem primjeru prikazano je smanjivanje veličine particije pomoću parted (**parted** upozorava pri smanjivanju o mogućnosti gubitaka podataka), a zatim je prikazan pokušaj izvođenja smanjivanja datotečnog sustava pomoću `resize2fs`:

```
# parted /dev/sdb
GNU Parted 3.2
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) resizepart
Partition number? 1
Warning: Partition /dev/sdb1 is being used. Are you sure you want to continue?
Yes/No? y
End?  [345MB]? 155
Warning: Shrinking a partition can cause data loss, are you sure you want to
continue?
Yes/No? y
(parted) q
Information: You may need to update /etc/fstab
```

Čak i nakon naredbe `mount` i dalje nije moguće smanjiti datotečni sustav:

```
# df -h |grep sdb1
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb1       314M  2.1M  296M   1% /tmp/mount
# resize2fs /dev/sdb1
resize2fs 1.42.12 (29-Aug-2014)
Filesystem at /dev/sdb1 is mounted on /tmp/mount; on-line resizing required
resize2fs: On-line shrinking not supported
```

Dakle, kada je datotečni sustav jednom kreiran na particiji, on se više ne može smanjiti. Jedini način za smanjivanje particije je posredno s uklanjanjem i ponovnim kreiranjem particije. Ako na disku postoji prostor, potrebno je kreirati novu particiju i kopirati sve podatke na nju te ju montirati umjesto originalne particije, a originalnu obrisati.

Za pregled i izmjenu parametara, poput ranije navedenih parametara provjere datotečnih sustava, koriste se specijalizirani alati. U donjoj tablici su navedeni alati, njihova namjena i datotečni sustavi za čije podešavanje su namijenjeni:

| Datotečni sustav | Alat za pregled | Alat za izmjenu parametara |
|---|---|---|
| **ext[2-4]** | dumpe2fs | tune2fs |
| **xfs** | xfs_info | xfs_admin |
| **ReiserFS** | debugreiserfs | reiserfstune |

Tablica 17 - Alati za pregled i podešavanje datotečnih sustava

Na primjeru će biti prikazani alati **dumpe2fs** i **tune2fs**. U donjem se primjeru pomoću dumpe2fs pribavljaju podaci o datotečnom sustavu. Zatim se postavljaju novi parametri i ponovno se pregledavaju nakon promjene. Prvo se pregledava i mijenja interval provjere u 21 dan, a zatim i maksimalni broj montiranja između provjere na 11.

```
# dumpe2fs /dev/sdb1|grep Check|grep inter
dumpe2fs 1.42.12 (29-Aug-2014)
Check interval:           0 (<none>)
# tune2fs /dev/sdb1 -i21d
tune2fs 1.42.12 (29-Aug-2014)
Setting interval between checks to 1814400 seconds
# dumpe2fs /dev/sdb1|grep Check|grep inter
dumpe2fs 1.42.12 (29-Aug-2014)
Check interval:           1814400 (3 weeks)
# dumpe2fs /dev/sdb1|grep Maximu
dumpe2fs 1.42.12 (29-Aug-2014)
Setting maximal mount count to 11
# dumpe2fs /dev/sdb1|grep Maximu
dumpe2fs 1.42.12 (29-Aug-2014)
Maximum mount count:      11
```

Naravno, kao i kod alata za provjeru datotečnih sustava potrebno je izmjene vršiti nad datotečnim sustavom kada nije montiran. Ukoliko je potrebno mijenjati parametre sistemskih datotečnih sustava poput „/“ datotečnog sustava, tada se sustav mora podići pomoću optičkog medija, kako je opisano u prvom poglavlju. Alati za pregled i izmjenu parametara, navedeni malo ranije, prisutni su na većini takvih (*rescue)* medija.

### 2.1.4. Provjera i oporavak datotečnih sustava

Brojne pogreške mogu naštetiti integritetu podataka na datotečnom sustavu. Ako se to dogodi, može doći i do gubitka podataka. U nekim slučajevima oštećenja onemogućuju Linuxu montiranje datotečnog sustava. Alati za provjeru i oporavak datotečnog sustava postoje kako bi minimalizirali gubitke podataka i omogućili pristup do oštećenog sustava. Glavni alat za ovu namjenu je **fsck**, ali je **fsck** poput **mkfs** tek sučelje prema drugim koji obavljaju provjeru i oporavak poput **e2fsck** ili **xfs_check** (koji traži probleme) i **xfs_repair** (koji uklanja greške). Također je moguć i izravan poziv alata za određeni datotečni sustav pomoću **fsck** <tip>, primjerice **fsck.ext2** ili **fsck.zfs**.

Sintaksa naredbe `fsck` je:

```
fsck [-sACVRTNP] [-t tip] [--] [fsck-mogućnosti] <datotečni_sustav>
```

Od brojnih mogućnosti koje su drugi parametar naredbe, nisu sve namijenjene korištenju pri pozivu iz komandne linije. Dio mogućnosti je zamišljen da se koristi pri pokretanju sustava. Mogućnosti su navedene i objašnjene u sljedećoj tablici:

| Mogućnost | Objašnjenje |
|---|---|
| **-A** | Provjera svih datotečnih sustava navedenih u /etc/fstab. |
| **-C** | Prikazivanje trake napretka, nije podržano na svim datotečnim sustavima. |
| **-V** | Pokretanje detaljnog izvještavanja o aktivnostima. |
| **-N** | Testno pokretanje – sustav javlja koje bi akcije izveo bez da to doista izvede. |
| **-t<tip>** | Standardno fsck automatski utvrđuje koji se tip datotečnog sustava, ali ovom mogućnosti se definira tip. Kada se koristi s –A mogućnosti, rezultat je provjera samo u /etc/fstab navedenih datotečnih sustava zadanog tipa. |
| **fsck-mogućnosti** | Navode se mogućnosti specifične za datotečni sustav. |
| **<datotečni_sustav>** | Definicija nad kojim datotečnim sustavom ili sustavima se vrše provjere. |

Tablica 18 - Mogućnosti fsck naredbe

Naredbu `fsck` treba izvoditi samo nad datotečnim sustavima koji trenutno nisu montirani ili su montirani samo s ovlastima za čitanje. Zapisivanje na disk tijekom provjera može rezultirati oštećenjima datotečnog sustava.

Primjer izvođenja naredbe `fsck`:

```
# fsck /dev/sdb2
fsck from util-linux 2.25.2
e2fsck 1.42.12 (29-Aug-2014)
/dev/sdb2: clean, 11/48960 files, 7731/195584 blocks
```

Linux pokreće `fsck` automatski pri pokretanju sustava i vrši `fsck` provjeru datotečnih sustava prema zapisima iz /**etc/fstab** datoteke. Normalno je ponašanje da se provjerava datotečni sustav svakih 20 montiranja ili svakih 180 dana (ovisno o tome što bude ranije).

### 2.1.5. Automatsko montiranje datotečnih sustava

Ako je potrebno da datotečni sustav bude dostupan i nakon ponovnog pokretanja sustava, mora se definirati montiranje tog datotečnog sustava u datoteci /**etc/fstab**. /**etc/fstab** datoteka upravlja kako Linux pristupa particijama diskova i vanjskim diskovnim uređajima. Linux podržava jedinstvenu datotečnu strukturu u kojoj se svi diskovni uređaji montiraju na neki direktorij i za

sustav nije važno ako je određeni poddirektorij samo dio particije montirane na neki direktorij iznad njega ili je to odvojeni uređaj. Tako direktorij **/home** može biti samo direktorij na uređaju koji je montiran na / ili zasebni uređaj montiran direktno na **/home**.

U nastavku će se detaljnije opisati **/etc/fstab** datoteka. Struktura **/etc/fstab** datoteke je niz zapisa koji definiraju montiranja. Svaka linija datoteke je definicija jednog montiranja. Svaka ta definicija sastoji se od 6 zapisa odvojenih razmacima, a značenje tih zapisa je prikazano u sljedećoj tablici:

| Broj polja | Ime parametra | Objašnjenje |
|---|---|---|
| **1** | Uređaj (engl. *device*) | Nazivi datotečnih uređaja koji označavaju hard diskove, disketne uređaje i slično. Danas je standardno korištenje znački ili univerzalnih jedinstvenih identifikatira (engl. *Universally Unique Identifier, UUID*)<br>primjerice **LABEL=/home** ili **UID=877859 bc-23c2-433c-9a3d-6780de40ea31**. |
| **2** | Mjesto montiranja | Prazan direktorij na nekom drugom uređaju. Izuzetak od ovog pravila su „/“ i swap. |
| **3** | Tip datotečnog sustava | Moguće je koristiti zapis **auto**, s pomoću kojeg pri montiranju sustav sam prepoznaje tip. |
| **4** | Mogućnosti montiranja | Mogućnosti montiranja su pravo za čitanje (ro) ili pravo na pisanje i čitanje (rw) i slično. Za dostupne mogućnosti potrebno je proučiti dokumentaciju odgovarajućeg datotečnog sustava. |
| **5** | **dump** polje | Ponašanje alata dump za ovaj datotečni sustav. dump je nekada bio standardni alat za izradu sigurnosnih kopija podataka. Vrijednost 1 uzrokuje da se vrši izrada sigurnosne kopije, a vrijednost 0 uzrokuje da se ne izrađuje. Danas je standardno staviti 0 jer se alat i taj način izrade sigurnosne kopije ne koristi. |
| **6** | **fsck** polje | Ponašanje **fsck** alata za provjeru integriteta datotečnog sustava. Vrijednost 0 definira da **fsck** ne provjerava datotečni sustav, vrijednosti drugačije od 0 definiraju da se datotečni sustav treba provjeravati. Provjera se vrši slijedno od najnižeg prema najvišem broju te bi „/“ trebao imati vrijednost 1, a ostali datotečni sustavi druge više vrijednosti. |

Tablica 19 - Polja /etc/fstab datoteke

Primjer **/etc/fstab datoteke**:

```
# cat /etc/fstab
UUID=877859bc-23c2-433c-9a3d-6780de40ea31 /               ext4    errors=remount-ro 0       1
UUID=1e674c8c-5534-44a0-9f51-91fb20a75dbd /boot           ext4    defaults        0       2
UUID=420a20f4-e155-4799-b226-5e605fbfad21 none            swap    sw              0       0
/dev/sr0        /media/cdrom0   udf,iso9660 user,noauto     0       0
```

Istaknimo zadnju liniju ove datoteke koja definira montiranje optičkih uređaja. U liniji je vidljivo da se može odabrati više tipova datotečnog sustava. Sustav koristi odgovarajući tip ovisno o mediju koji se nalazi u optičkom uređaju.

Ranije su u poglavlju 2.1.1. spomenuti SMB/CIFS i NFS datotečni sustavi koji omogućavaju korisnicima sa jednog sustava da pristupe datotekama na udaljenom računalu. Uz ovu funkcionalnost postoji problem dostupnosti: ako je udaljeni datotečni sustav postavljen da je neprestano montiran postoji povećana vjerojatnost da će datotečni sustav postati nedostupan. Kada je nedostupan datotečni sustav koji bi trebao biti montiran, brojni alati poput **df** počnu se neobično ponašati – čekaju da sustav postane dostupan. Održavanje takvih datotečnih sustava neprestano montiranim troši resurse, uključujući i mrežne resurse, a to može biti problem na serverima s velikim brojem klijenata.

Za rješenje ovakvih problema postoji **autofs** – alat za automatsko montiranje. Sistemski alat **autofs** nije standardni dio instalacije svih distribucija, a naknadno ga je moguće instalirati instalacijom istoimenog paketa.

Središnja konfiguracijska datoteka **autofs** je **/etc/auto.master**. Standardno se u toj datoteci definiraju montiranja za određene direktorije i konfiguracijske datoteke u kojima se nalaze definicije montiranja poddirektorija. Primjerice, ako se želi definirati montiranje direktorija /**home** i /**shared/auto**, tada će se u /**etc/auto.master** unijeti sljedeći zapisi:

```
/home /etc/auto.homovi --timeout=60
/shared/auto /etc/auto.dijeljeni --timeout=60
```

Dakle, montiranje poddirektorija **/home** definira sadržaj **/etc/auto.homovi** datoteke, a montiranje poddirektorija **/shared/auto** definira **/etc/auto.dijeljeni** datoteku. Zapisi u **/etc/auto.homovi** i **/etc/auto.dijeljeni** datotekama definiraju montiranja poddirektorija od direktorija navedenih u **/etc/auto.master** na način da se definiraju relativne putanje. Parametar **--timeout** definira koliko sekundi od zadnje aktivnosti na datotečnom sustavu isti ostaje montiran.

Ako želimo montirati direktorije /**shared/auto/novi/trivijalni** i /**shared/auto/stari** tada bi zapisi u /**etc/auto.dijeljeni** mogli izgledati ovako:

```
novi/trivijalni www.l201.hr:/home/novi/trivic
stari -fstype=nfs4 www.l201.hr:/data/dokumenti/sigs
```

Na taj se način definiraju dva **nfs** montiranja, s napomenom da je u drugom slučaju eksplicitno definirano da je u pitanju nfs4. Prva linija kôda definira montiranje udaljenog direktorija

/**home/novi/trivic** s računala www.l201.hr na direktorij /**shared/auto/novi/trivijalni**, a druga linija definira montiranje direktorija /**data/dokumenti/sigs** s istoga računala na direktorij /**shared/auto/stari**.

**autofs** izbjegava nepotrebno trošenje resursa, budući da neće montirati neaktivne direktorije. Prema definicijama u konfiguracijskim datotekama, **autofs** kreira listu direktorija za koje je zadužen. Kada se registrira pokušaj pristupanja tim direktorijima ili njihovim poddirektorijima izvršava se montiranje (samo potrebnih direktorija). Nakon što istekne vrijeme definirano **parametrom „timeout“** (u gornjem primjeru 60 sekundi) od zadnje akcije na montiranom datotečnom sustavu, taj će datotečni sustav postati **demontiran** (engl. *unmounted*).

## 2.2. Vježba: Upravljanje datotečnim sustavima

1. Prije početka rada odaberite sliku stanja virtualnoga računala (engl. Snapshot) za početak ove vježbe.
2. Prijavite se na računalo kao korisnik l201. U GUI-ju pokrenite **Terminal (Activities → Terminal)**. Izvršite su - naredbu da postanete administrator (lozinka: L201).
3. Kreirajte 4 nove particije veličine 2 GB na **/dev/sdb**, koristeći **parted**. Particije neka budu tipa **ext2, ext3, ext4 i xfs** slijedno. Prvo morate postaviti oznaku (engl. label) alatom **mklabel** u vrijednost **msdos**. Prije izlaska iz alata **parted** provjerite stanje s **print**.
4. Stvorite na novim particijama slijedno datotečne sustave ext2, ext3, ext4 i xfs.
5. Kreirajte direktorije **/mnt/disk1, /mnt/disk2, /mnt/disk3 i /mnt/disk4**.
6. Montirajte datotečne sustave na te direktorije slijedno.
7. Provjerite kapacitet montiranih datotečnih sustava. Što uočavate?

Zašto je to tako?

8. Odmontirajte sva četiri datotečna sustava.
9. Pomoću **parted** obrišite drugi datotečni sustav.
10. Pomoću parted alata povećajte prvu particiju na veličinu od 3 GB.
11. Možete li ostali prostor dodati posljednje kreiranoj particiji? Ako da, napravite tako.

12. Montirajte proširenu particiju na **/mnt/disk1**. Koliki je kapacitiet i zašto?

13. Pomoću **resize2fs** povećajte particiju na sav dostupan prostor, ponovno provjerite kapacitet.

14. Odmontirajte sve datotečne sustave i obrišite sve particije na **/dev/sdb** disku.

## 2.3. Vježba: Automatsko mountiranje

1. Prije početka rada odaberite sliku stanja virtualnoga računala (engl. Snapshot) za početak ove vježbe.
2. Prijavite se na računalo kao korisnik l201. U GUI-ju pokrenite **Terminal (Activities → Terminal)**. Izvršite su - naredbu da postanete administrator (lozinka: L201).
3. Stvorite particiju od **1000 MB** na **/dev/sdb** – tip **ext4**. Stvorite datotečni sustav istoga tipa na toj particiji. Prvo morate postaviti oznaku (engl. label) s **mklabel** u vrijednost **msdos**.
4. Dodajte sljedeću liniju u **/etc/auto.master**:

   ```
   /var/autofs/novi/etc/auto.novi --timeout=2,sync,nodev,nosuid
   ```

   (Navedeno ide **sve** u jednu liniju.)
5. Stvorite **/etc/auto.novi** datoteku sadržaja:

   ```
   prvi -fstype=ext4     :/dev/sdb1
   ```
6. Provjerite stanje montiranih datotečnih sustava.
7. Ponovno pokrenite autofs.
8. Ponovno provjerite stanje montiranih datotečnih sustava.
9. Izvršite sljedeću naredbu:

   ```
   # ls /var/autofs/novi/prvi && df
   ```
10. Koja je razlika u odnosu na ranije izvršavanje? Zašto?

---

Ponovite naredbu df –h. Koji je sada izlaz i zašto?

---

# 3. Spremišni sustavi

**Trajanje poglavlja:**
**300 min**

Po završetku ove cjeline moći ćete:

- razlikovati tipove RAID-ova
- pripremiti sustav za kreiranje softverskog RAID-a
- stvoriti softverski RAID
- stvoriti i koristiti LVM
- prilagoditi LVM svojim potrebama
- nadgledati stanje LVM
- izraditi optički medij u Linuxu
- izraditi optički medij za pokretanje sustava u Linuxu
- razumjeti sastav zapisa na optičkom mediju
- razumjeti što je **udev**
- prepoznati standardne virutalne uređaje u direktoriju **/dev**
- prilagoditi **udev**

U ovoj se cjelini obrađuju spremišni sustavi operacijskog sustava *Linux*. U cjelini su opisani RAID-ovi i softverski spremišni sustav. Prikazan je postupak stvaranja i prilagodbe RAID-a i softverskog spremišnog prostora. Cjelina obrađuje i izradu optičkih medija na *Linux*u, a završava sa objašnjenjem rada *udev* upravljača uređajima u *Linux*u.

## 3.1. RAID

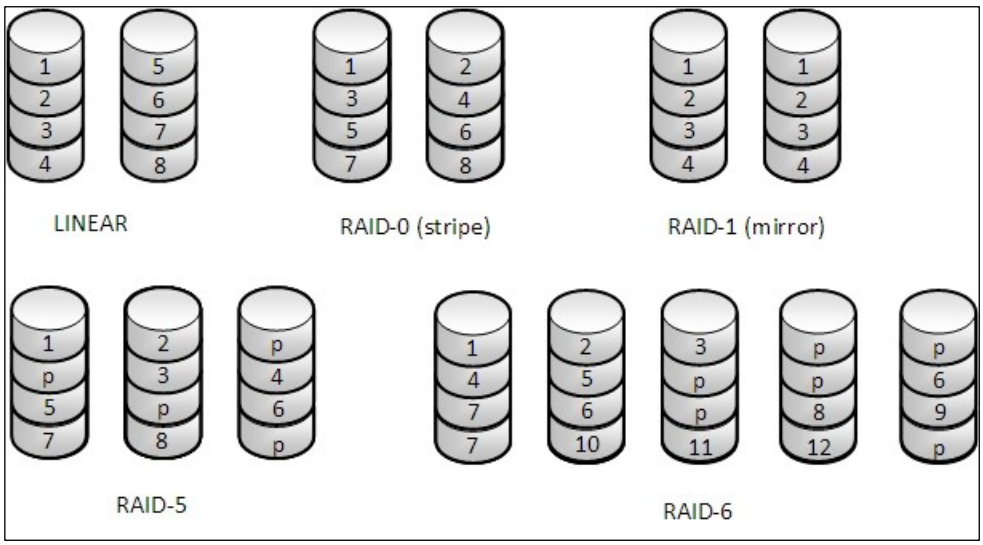
### 3.1.1. Vrste RAID-a

RAID (engl. *Redundant Array of Independent Disks*) je redundantan niz neovisnih diskova, što je ujedno vrlo precizna definicija uloge RAID-a. Izvorno se taj akronim tumačio i kao redundantan niz jeftinih diskova (engl. *Redundant Array of Inexpensive Disks*). Situacija gdje se više jeftinih (ujedno male brzine i malog kapaciteta) diskova koristi da bi se postigla bolja svojstva (brzina ili kapacitet) je česta u praksi, a rješenje je povezivanje takvih diskova u RAID.

*Važno je napomenuti da su diskovi (ne uključujući u ovu izjavu nove SSD diskove) najsporiji dio računala i da drugi dijelovi sustava (matična, procesor …) često čekaju na izvršavanje čitanja/zapisivanja.*

Danas se RAID-ovi koriste u brojnim naprednim sustavima na diskovima velike brzine i kapaciteta (ali i ne male cijene). Ovisno o sustavu gdje se koriste i tipu RAID-a koriste se za povećanje performansi, pouzdanosti ili koombinacije tog dvojeg.

Na donjoj slici prikazani su standardni tipovi RAID-a. Brojem označeni cilindri predstavljaju pojedine diskove:




Slika 1 - Standardni tipovi RAID-a

Na slici su prikazane minimalne konfiguracije za svaki tip RAID-a. Brojevi (**1-8**) predstavljaju podatke, a slovo **p** označava paritetne bitove. Slijede kratki opisi navedenih 5 tipova RAID-a te usporedba operacije čitanja i zapisivanja sa diskovima nad kojima nije konfiguriran RAID:

1. Linearni – Slijedno povezivanje više diskova u jedan veći. Nema promjena u brzinama čitanja i zapisivanja, ne postoji redundancija.
2. RAID-0 – Podaci se paralelno zapisuju na diskove. Ostvaruje se ubrzanje čitanja i zapisivanja zato što se pojedine datoteke čitaju/zapisuju ~50% na svakom disku u polju, ne postoji redundancija.
3. RAID-1 - Jednostavno repliciranje podataka. Čitanje je ubrzano zato što je moguće paralelno izvođenje, ali je zapisivanje usporeno, ostvarena je redundancija.
4. RAID-5 – Podaci su raspodijeljeni na sve diskove. Koriste se paritetni bitovi za oporavak, a paritetni bitovi nisu svi na jednom disku. Zbog ovakvog rasporeda uklanja se problem sporijeg zapisivanja vidljiv u RAID-4 polju koje se od RAID-5 razlikuje samo u tome da su svi paritetni bitovi na jednom disku, a ostaje brže čitanje i redundancija podataka.
5. RAID-6 – Proširenje RAID-5 gdje se koriste dva diska za ostvarenje pariteta. Ova konfiguracija je korisna kada se želi postići visoka razina pouzdanosti. Zbog potrebe izračuna dva pariteta vrijeme zapisivanja je značajno povećano. RAID-6 koristi se u upotrebi, ali je to polje kompleksno i zahtijeva veliki broj diskova te se rjeđe koristi u praksi (osim u SAN (engl. *Storage Area Network*) i NAS (engl. *Network-attached storage*) sustavima).

**Napomena**

Važna stvar koja čini hardversko polje pouzdanijim i robusnijim je što se metapodaci ne smještaju na istoj lokaciji kao i sami podaci. Ovo naravno nije slučaj u softverskom RAID polju, ali moguće je konfiguracijom definirati da se metapodaci pohranjuju na više lokacija.

Kako je vidljivo iz numeričkih oznaka postoje uz spomenute još i RAID-2 i RAID-3. RAID-2 je u potpunosti napušten u praksi, a razlika RAID-3 i RAID-4 minimalna, međutim ta razlika čini performanse i mogućnosti RAID-4 polja daleko superiornijim.

Dodatno se koriste i brojne kombinacije poput RAID-10, RAID-50 i RAID-60 koji su najzastupljeniji u praksi.

Postoje dva načina ostvarivanja RAID-a – softverski i hardverski. U slučaju hardverskog RAID-a koristi se specijalizirana podrška za ostvarenje RAID-a, a mogući tipovi RAID-a ovise o tome što podržava hardver i RAID upravljač (engl. *RAID controller*). Kod softverskog RAID-a može se instalirati dodatna podrška i izgraditi RAID bilo kojeg tipa za koji je ranije instalirana podrška. Što se svojstava tiče, hardverski RAID-ovi su brži i pouzdaniji, a softversko ostvarenje je besplatno i jedino moguće kada ne postoji specijalizirani hardver.

### 3.1.2. Linuxov softverski RAID

*Linux*ov softverski RAID je implementacija RAID-a bez korištenja RAID upravljača. U ovoj implementaciji se koristi specijalizirani softver za upravljanje koji će omogućiti da se ostvari funkcionalnost RAID-a nad proizvoljnim uređajima odnosno particijama.

Linuxov softverski RAID uobičajeno se ostvaruje nad particijama. Particije spajaju RAID upravljački programi jezgre u nove uređaje imenovane **/dev/md#** gdje je „**#**“ redni broj polja počevši od 0. Konfiguracija pomoću particija omogućava kreiranje RAID-a s diskovima različite veličine kreiranjem particija odgovarajuće veličine. Idealna veličina bi bila ekvivalent najvećeg zajedničkog djelitelja.

Važni nedostatak *Linux*ovog softverskog RAID-a je nedostatak podrške u nekim programima za učitavanje sustava. GRUB Legacy primjerice ne podržava RAID osim RAID-1 gdje je jednostavno moguće montirati jednu od particija samo za čitanje (engl. *read only*). Zbog ovoga je dobra ideja ostaviti nešto prostora na diskovima za običnu **/boot** particiju ili za izradu RAID-1 polja za potrebe te particije.

Rad softverskog RAID-a oslanja se na podršku u jezgri. Većina distribucija standardno pruža tu podršku, ali ako sami programski prevodimo jezgru potrebno je paziti da se pri konfiguraciji aktiviraju sva kasnije potrebna podrška za rad sa RAID-om. Podrška za RAID-ove se pri konfiguraciji nalazi u *Device Drivers → Multiple Devices Driver Support (RAID and LVM) → RAID Support area*.

Prvi korak u izradi softverskog RAID-a je particioniranje diskova. Važno je da se pri particioniranju diskova dodijeli odgovarajuća oznaka (kôd tipa). Oznaka je kod MBR diskova 0xFD, GPT (engl. *GUID Partition Table*) diskova 0xFD00, a kada se koriste alati zasnovani na libparted tada se mora postaviti RAID zastavica (engl. *RAID flag*).

U donjem primjeru dodajemo za particiju broj 5 zastavicu za kasnije korištenje u RAID-u.

```
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 2147MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1      1049kB  300MB   299MB   primary   ext4
 2      300MB   500MB   200MB   primary   ext2
 3      500MB   2147MB  1647MB  extended  lba
 5      500MB   600MB   99.8MB  logical   ext2     lba

(parted) set 5 raid on
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 2147MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1      1049kB  300MB   299MB   primary   ext4
 2      300MB   500MB   200MB   primary   ext2
 3      500MB   2147MB  1647MB  extended  lba
 5      500MB   600MB   99.8MB  logical   ext2         raid, lba
```

Naredba za postavljanje zastavice je `set`. U ovom slučaju je to `set 5 raid on`, dakle potrebno je postaviti zastavicu raid nad particijom 5.

Dva su standardna pristupa izradi particija za potrebe softverskog RAID-a. Prvi je pristup da se na svakom disku koji ulazi u RAID napravi particija preko cijelog diska, a drugi pristup je da se izradi više particija. Prvi je pristup jednostavniji, ali praktičan samo u situaciji kada su diskovi iste veličine. Drugi pristup je daleko fleksibilniji zato što diskovi ne moraju biti iste veličine i moguće je kreirati više različitih tipova RAID-a za različita montiranja u ovisnosti o potrebama. Primjerice, particija **/var** mogla bi biti RAID-0, a /**data** particija RAID-1 - ako se s nje često čita i rijetko zapisuje, a podaci su važni i ne smiju biti izgubljeni.

Kada se kreira RAID nad uređajima različite veličine nastaje gubitak. Naime, veličina RAID-a od n uređaja različitih veličina definirana je veličinom najmanjeg od njih. Dodatni prostor koji postoji na svim većim diskovima neće biti iskorišten. Primjerice, ako se radi o 3 uređaja kapaciteta 1.2, 1.7 i 1GB te se od njih napravi RAID-5, nastat će RAID kapaciteta 2GB (1GB + 1GB za podatke i 1GB za paritet). Ovaj gubitak je moguće izbjeći dijeljenjem uređaja na manje particije i kreiranjem RAID-a nad tim particijama.

### 3.1.3. Upravljanje s Linuxovim softverskim RAID-om

U trenutku kada su pripremljene particije za formiranje RAID-a, nad njima se pomoću naredbe `mdadm` upravlja RAID-om. Naredba `mdadm` je zamjena za zastarjeli **raidtools** paket. `mdadm` je program za kreiranje, upravljanje i nadzor MD uređaja. Naredba `mdadm` ima složenu sintaksu:

```
mdadm [način_rada] raid-uređaj [mogućnosti] uređaj
```

Standardni načini rada i mogućnosti prikazani su i objašnjeni u donjim tablicama:

| Način rada | Kratki oblik | Objašnjenje |
|---|---|---|
| **--assemble** | -A | Naredba za sastavljanje RAID-a; ne odnosi se na inicijalno stvaranje već za popravak ili ponovno povezivanje postojećeg polja. |
| **--build** | -B | Kreiranje polja bez metapodataka potrebnih za rad RAID-a. Samo za posebne potrebe za rad stručnjaka. |
| **--create** | -C | Stvaranje novog RAID-a. |
| **--follow ili --monitor** | -F | Pokretanje praćenja stanja nekog RAID-a. |
| **--grow** | -G | Promjena veličine polja. |
| **--incremental** | -I | Dodavanje jednog uređaja u polje. |
| **--auto-detect** | Ne postoji | Slanje zahtjeva jezgri da pronađe i automatski aktivira sve RAID uređaje koje može. |

Tablica 20 - Standardni načini rada mdadm naredbe

| Mogućnost | Kratki oblik | Mod rada | Objašnjenje |
|---|---|---|---|
| **--help** | -h | svi | Osnovna pomoć za rad sa mdadm alatom. |
| **--version** | -v | svi | Javlja verziju mdadm. |
| **--verbose** | -v | svi | Detaljnije izvještavanje pri izvršavanju. |
| **--force** | -f | svi | Prisilno izvršavanje, pri čemu se ignoriraju neka upozorenja i pogreške. |
| **--config=<datoteka>** | -c <datoteka> | svi | Definira konfiguracijsku datoteku za korištenje. Bez ove mogućnosti koristi se zadane postavke koje definiraju /etc/mdadm.conf ili |

| | | | |
|---|---|---|---|
| | | | /etc/mdadm/mdadm.conf ako prva ne postoji. |
| **--scan** | -s | svi | Definira da se sve mogućnosti koje nisu zadane učitaju iz konfiguracijskih datoteka. |
| **--metadata=<razina>** | -e <razina> | svi | Definira stil metapodataka. Zadana je vrijednost 0.90 koja ograničava RAID na 28 komponenti i 2 TiB. Vrijednosti 1, 1.0, 1. i 1.2 povećavaju ta ograničenja. |
| **--raid-devices=<broj>** | -n <broj> | create\| build\| grow | Definira koliko je aktivnih uređaja u polju. |
| **--spare-devices=<broj>** | -x <broj> | create\| build\| grow | Definira broj rezervnih uređaja u polju. |
| **--chunk=<veličina>** | -c < veličina> | create\| build\| grow | Definira veličinu bloka u kibibytovima. |
| **--level=<tip>** | -l <tip> | create\| build | Definira tip RAID-a. Moguće vrijednosti su: linear, raid0, 0, stripe, raid1, 1, mirrot, raid4, 4, raid5, 5, raid6, 6, raid10, 10, multipath, mp, faulty i container. |
| **--name=<ime>** | -N <ime> | create\| build\| grow\| assemble | Definira ime za RAID. Moguće samo kada je metadata stil 1. |
| **--add** | -a | manage | Dodavanje novog uređaja u polje. |
| **--re-add** | Ne postoji | manage | Vraća ranije uklonjeni uređaj u polje. |
| **--remove** | -r | manage | Uklanja uređaj iz polja. |
| **--fail ili --set-faulty** | -f | manage | Označava uređaj kao neispravan. |
| **--stop** | -S | misc | Zaustavlja cijelo polje. |

Tablica 21- Standardne mogućnosti mdadm naredbe

Pošto je `mdadm` opsežan alat s brojnim mogućnostima, u tablici 21 su navedene najčešće korištene. Za informacije o ostalim mogućnostima pročitajte pomoć **mdadm** alata.

Iako su sami RAID-ovi kao pojam kompleksni za razumijevanje, njihovo korištenje je vrlo jednostavno. Slijedi primjer kreiranja RAID-a.

Na disku **/dev/sdb** kreirane su 4 particije iste veličine: **/dev/sdb5, /dev/sdb6, /dev/sdb7** i **/dev/sdb8**. Particijama je postavljena raid zastavica.

```
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 2147MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:
Number  Start   End     Size    Type      File system  Flags
 1      1049kB  300MB   299MB   primary   ext4
 2      300MB   500MB   200MB   primary   ext2
 3      500MB   2147MB  1647MB  extended  lba
 5      500MB   600MB   99.8MB  logical   ext2         raid, lba
 6      600MB   700MB   100MB   logical   ext2         raid, lba
 7      700MB   800MB   100MB   logical   ext2         raid, lba
   8      800MB   900MB   100MB   logical   ext2         raid, lba
Slijedi kreiranje RAID-a tipa 5 korištenjem prve 3 particije i
dodjeljivanjem naziva /dev/md0:
# mdadm --create /dev/md0 --level=5 --raid-devices=3 /dev/sdb5 /dev/sdb6
/dev/sdb7
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
```

Datotečni sustav može se jednostavno kreirati, kao nad bilo kojom particijom - naredbom `mkfs`, a kreirani datotečni sustav može se montirati naredbom `mount`.

```
# mkfs -t ext4 /dev/md0
mke2fs 1.42.12 (29-Aug-2014)
Creating filesystem with 192512 1k blocks and 48192 inodes
Filesystem UUID: d241dca8-b60f-490a-8fc7-b1d425b4f19d
Superblock backups stored on blocks:
       8193, 24577, 40961, 57345, 73729

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
# mkdir /mnt/tmp
# mount /dev/md0 /mnt/tmp/
# df -h|grep tmp|grep mnt
Filesystem      Size  Used Avail Use% Mounted on
/dev/md0        179M  1.6M  164M   1% /mnt/tmp
```

U primjeru je vidljivo da je nastali datotečni sustav duplo veći od pojedine particije. Sada se može dodati zadnja particija u RAID. Prije toga potrebno je pogledati stanje RAID-a **/dev/md0**.
U **/proc/mdstat** datoteci nalaze se podaci o svim aktivnim RAID-ovima.

```
# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdb7[3] sdb6[1] sdb5[0]
    192512 blocks super 1.2 level 5, 512k chunk, algorithm 2 [3/3] [UUU]

unused devices: <none>
```

Za detaljniji prikaz može se koristiti naredba `mdadm -detail <raid_polje>`:

```
# mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Sat Jan 14 16:50:52 2017
     Raid Level : raid5
     Array Size : 192512 (188.03 MiB 197.13 MB)
  Used Dev Size : 96256 (94.02 MiB 98.57 MB)
   Raid Devices : 3
  Total Devices : 3
    Persistence : Superblock is persistent

    Update Time : Sat Jan 14 16:52:32 2017
          State : clean
 Active Devices : 3
Working Devices : 3
 Failed Devices : 0
  Spare Devices : 0

         Layout : left-symmetric
     Chunk Size : 512K

           Name : l201:0  (local to host l201)
           UUID : 92ee3c7f:6917fa2b:72578678:818d3422
         Events : 19

    Number   Major   Minor   RaidDevice State
       0       8       21        0      active sync   /dev/sdb5
       1       8       22        1      active sync   /dev/sdb6
       3       8       23        2      active
sync   /dev/sdb7
```

Vidljivo je u oba primjera da se sva tri uređaja koriste te su svi ispravni. Sada je potrebno dodati još jedna particija i ponovno provjeriti stanje:

```
# mdadm --add /dev/md0 /dev/sdb8
mdadm: added /dev/sdb8
# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdb8[4](S) sdb7[3] sdb6[1] sdb5[0]
      192512 blocks super 1.2 level 5, 512k chunk, algorithm 2 [3/3] [UUU]
```

```
unused devices:
# mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Sat Jan 14 16:50:52 2017
     Raid Level : raid5
     Array Size : 192512 (188.03 MiB 197.13 MB)
  Used Dev Size : 96256 (94.02 MiB 98.57 MB)
   Raid Devices : 3
  Total Devices : 4
    Persistence : Superblock is persistent

    Update Time : Sat Jan 14 17:23:19 2017
          State : clean
 Active Devices : 3
Working Devices : 4
 Failed Devices : 0
  Spare Devices : 1

         Layout : left-symmetric
     Chunk Size : 512K

           Name : l201:0  (local to host l201)
           UUID : 92ee3c7f:6917fa2b:72578678:818d3422
         Events : 20

    Number   Major   Minor   RaidDevice State
       0       8       21        0      active sync   /dev/sdb5
       1       8       22        1      active sync   /dev/sdb6
       3       8       23        2      active sync   /dev/sdb7
             4       8       24        -      spare   /dev/sdb8
```

Vidljivo je da je posljednji uređaj ispravan, ali se ne koristi. Koristit će se ako se pokvari neki od aktivnih uređaja. Kako bi se testirala ta funkcionalnost, potrebno je proglasiti **/dev/sdb6** neispravnim i pogledati što se dogodilo.

```
# mdadm --fail /dev/md0 /dev/sdb6
mdadm: set /dev/sdb6 faulty in /dev/md0
# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdb8[4] sdb7[3] sdb6[1](F) sdb5[0]
      192512 blocks super 1.2 level 5, 512k chunk, algorithm 2 [3/3] [UUU]

unused devices:
# mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Sat Jan 14 16:50:52 2017
     Raid Level : raid5
     Array Size : 192512 (188.03 MiB 197.13 MB)
  Used Dev Size : 96256 (94.02 MiB 98.57 MB)
   Raid Devices : 3
```

```
  Total Devices : 4
    Persistence : Superblock is persistent

    Update Time : Sat Jan 14 17:27:03 2017
          State : clean
 Active Devices : 3
Working Devices : 3
 Failed Devices : 1
  Spare Devices : 0

         Layout : left-symmetric
     Chunk Size : 512K

           Name : l201:0  (local to host l201)
           UUID : 92ee3c7f:6917fa2b:72578678:818d3422
         Events : 39

    Number   Major   Minor   RaidDevice State
       0       8       21        0      active sync   /dev/sdb5
       4       8       24        1      active sync   /dev/sdb8
       3       8       23        2      active sync   /dev/sdb7
       1       8       22        -      faulty   /dev/sdb6
# df -h |grep tmp|grep mnt
Filesystem      Size  Used Avail Use% Mounted on
/dev/md0        179M  1.6M  164M   1% /mnt/tmp
# touch /mnt/tmp/test
# ls /mnt/tmp/
lost+found  test
```

Na kraju gornjeg primjera testirano je stanje sustava te je utvrđeno da je montiranje još uvijek aktivno i da je moguće i zapisivanje i čitanje na datotečnom sustavu. Neispravan uređaj je moguće ukloniti naredbom `mdadm -remove`. U praksi to izgleda ovako:

```
# mdadm --remove /dev/md0 /dev/sdb6
mdadm: hot removed /dev/sdb6 from /dev/md0
# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdb8[4] sdb7[3] sdb5[0]
      192512 blocks super 1.2 level 5, 512k chunk, algorithm 2 [3/3] [UUU]

unused devices:
# mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Sat Jan 14 16:50:52 2017
     Raid Level : raid5
     Array Size : 192512 (188.03 MiB 197.13 MB)
  Used Dev Size : 96256 (94.02 MiB 98.57 MB)
   Raid Devices : 3
  Total Devices : 3
    Persistence : Superblock is persistent

    Update Time : Sat Jan 14 17:31:30 2017
```

```
          State : clean
 Active Devices : 3
Working Devices : 3
 Failed Devices : 0
  Spare Devices : 0

         Layout : left-symmetric
     Chunk Size : 512K

           Name : l201:0  (local to host l201)
           UUID : 92ee3c7f:6917fa2b:72578678:818d3422
         Events : 40
    Number   Major   Minor   RaidDevice State
       0       8       21        0      active sync   /dev/sdb5
       4       8       24        1      active sync   /dev/sdb8
       3       8       23        2      active sync   /dev/sdb7
```

## 3.2 Softverski spremišni sustav

### 3.2.1 LVM

**LVM** je akronim za *Logical Volume Management*, a ono što LVM radi jest omogućavanje fleksibilnosti i jednostavnosti manipulacije nad spremištima podataka. LVM je nastao da omogući fleksibilnost koja ne postoji kada se koriste particije. Za ostvarivanje fleksibilnosti potrebna je dodatna složenost u obliku tri razine podatkovnih struktura.

Primjer fleksibilnosti koju nemaju particije vidljiv je na primjeru u slici 2:


Slika 2 - Primjer prednosti LVM-a

Na slici su particije koje su slobodne za korištenje označene bijelom bojom. U danom primjeru je na disku slobodno preko 20GB prostora. Ipak, stvoriti particiju od 20GB nije moguće. Potrebno je naglasiti činjenicu spomenutu u ranijim primjerima izrade particija: Svaka particija sastoji se od neprekinutog prostora na disku! Zbog toga je pri kreiranju particije dovoljno imenovati prvi i posljednji blok, budući da se time jednoznačno definira particija.

Ovaj nedostatak moguće je izbjeći korištenjem alata koji manipuliraju postojećim particijama. Primjerice, sa **gparted** bismo mogli razmjestiti sve postojeće particije tako da zauzimaju neprekinuti prostor slijedno od početka diska. Ipak, ovaj pristup ima dva velika nedostatka:

1. Postupak razmještaja svih podataka traje dugo i
2. Ako tijekom tog postupka nastane neka nepredviđena greška mogu biti izgubljeni svi podaci na particiji koja je u procesu razmještaja.

LVM sa slobodnim prostorom na uređajima upravlja slično kao sa slobodnim prostorom na datotečnom sustavu. Kada se na datotečnom sustavu kreira datoteka ona ne zauzima prostor slijedno na disku, već ga zauzima bez obzira na lokaciju, veličinu i raspored prostora. Da bi mogao tako raditi sa diskovima LVM koristi podatkovne strukture na tri razine – PV (engl. *Physical Volume*), VG (engl. *Volume Group*) i LV (engl. *Logical Volume*).

**Fizičko spremište** (PV) je particija namijenjena za korištenje kao dio LVM. Budući da neki programi za učitavanje sustava ne mogu raditi sa LVM particijama, dobra je ideja ne kreirati fizička spremišta nad cijelim spremišnim prostorom.

**Skupina spremišta** (VG) je skup jednog ili više PV. Ovakvo povezivanje omogućava kreiranje prostora za alokaciju koji može biti veći od bilo kojeg pojedinačnog uređaja dostupnog sustavu. Primjerice sa 2 diska kapaciteta 1TB mogu se kreirati 2 fizička spremišta kapaciteta 1TB i povezati u skupinu spremišta kapaciteta 2 TB.

**Logičko spremište** (LV) je najviša razina podatkovne apstrakcije u LVM. U logičko spremište se dodjeljuje dio ili cijela skupina spremišta. Pri korištenju logičkog spremišta korisniku nije važno niti je moguće specificirati koji se dio skupine spremišta koristi. Logičko spremište ima pristup određenom spremišnom prostoru i po potrebi iskoristi prostor iz tog „bazena“ resursa. Primjer: Na raspolaganju je skupina spremišta kapaciteta 4GB sastavljena od 4 fizička spremišta kapaciteta 1GB i dodijeljena jednom logičkom spremištu (dakle, Logičko spremište će biti sastavljeno od skupine spremišta kapaciteta 4TB, a na nižoj razini od 4 fizičkih spremišta, svakog veličine 1TB). Za primjer se može kreirati datotečni sustav od 2GB na tom LV. Pri kreiranju datotečnog sustava pribavi se 2TB podataka koji mogu biti iz bilo kojeg para fizičkih spremišta ili sa sva 4.

Ovakav način gdje logičko spremište ima pristup određenim resursima i samo treba definirati koliko resursa treba ima još jednu prednost – veliku fleksibilnost. Logička spremišta i grupe spremišta se mogu proširivati po potrebi. LVM je posebno koristan ako se koristi za datotečne sustave koji imaju veliku razinu nepredvidljivosti. Pomoću LVM moguće je lako i brzo kreirati nove datotečne sustave, povećati postojeće, dodavati diskove sustavu i slično.

Najveći nedostatak LVM je dodatna složenost. Ipak pri korištenju LVM čest je slučaj da se stvara manji broj particija nego bez LVM. Potrebno je koristiti više LVM alata kako bi te particije bile pretvorene u fizičko spremište, zatim povezane u skupine spremišta, što bi u konačnici dovelo do kreiranja logičkih spremišta koja se mogu koristiti. Dodatni alati za rad sa LVM konfiguracijom moraju postojati u initramfs datoteci, kao i u jezgri, kako bi konfiguracija bila aktivna nakon ponovnog pokretanja sustava.

Još jedan nedostatak LVM je da u slučaju kvarova LVM konfiguracija može pogoršati doseg incidenta. Naime, kada je logičko spremište rasprostranjeno preko više fizičkih uređaja, kvar samo jednog od tih uređaja znači i gubitak cijelog logičkog spremišta. Podacima u LVM upravljanom sustavu moguće je pristupiti samo ako su LVM metapodaci dostupni. Bez konfiguracijskih podataka svi zapisi na svim fizičkim spremištima su beskorisni.

Praktičan način za zaštitu podataka u slučaju ovakvog ispada je kreiranje fizičkih spremišta na nekoj od RAID konfiguracija sa redundancijom. Dodatno, danas se virtualizacija koristi više od samostalnih servera, a ona uobičajeno koristi posebni diskovni sustav sa ugrađenom redundancijom.

Virtualizacija također pruža visoke mogućnosti upravljanja virtualnim diskovima kao i LVM pa je u toj okolini često LVM redundantan.

### 3.2.2 Upravljanje LVM-om

Rad LVM može se raščlaniti na stvaranje i kasnije upravljanje. Ta su dva koraka razdvojena na tri razine – stvaranje i upravljanje sa fizičkim spremištem; stvaranje i upravljanje sa skupinama spremišta; stvaranje i upravljanje sa logičkim spremištima.

Većina alata s kojima smo se susretali funkcioniraju na način da se jedna naredba koristi za niz operacija. Dodatno se za kreiranje LVM konfiguracija uz **lvm** naredbu sa velikim brojem mogućnosti može koristiti i niz naredbi koje počinju s dva ključna slova za imenovanje podatkovne strukture nad kojom se LVM operacija izvršava. Naredbe koje počinju sa ta dva ključna znaka namijenjene su za korištenje samo nad podatkovnom strukturom za koju su akronim ta dva slova.

**Stvaranje i upravljanje sa fizičkim spremištima**

Prvi korak pri korištenju LVM je kreiranje fizičkog spremišta. Prije nego što se kreira fizičko spremište potrebno je particije koje se koriste kao fizička spremišta označiti točnim kôdom za korištenje u LVM. MBR kôd za LVM particije je 0x8E, a za GPT diskove =x8E00. Alati zasnovani na libparted podržavaju korištenje zastavice lmv. Tako će u `parted` naredba za postavljanje značke biti **set <broj> lvm on**.

U trenutku kada postoje uređaji za korištenje (pravilno označeni) moguće je krenuti prilagođavati njihov sadržaj. Za kreiranje i upravljanje sa fizičkim spremištem koristi se niz naredbi koje počinju sa pv. Moguće je izvršavati iste zadatke izvođenjem naredbe `lvm` s odgovarajućim mogućnostima. Pregled najvažnijeg dijela naredbi za upravljanje sa fizičkim spremištima prikazan je u donjoj tablici.

| Naredba | Objašnjenje |
|---|---|
| **pvchange** | Izmjena dozvola alokacije nad fizičkim spremištem. |
| **pvck** | Traži greške na fizičkom spremištu, slično kao naredba fsck za datotečne sustave. |
| **pvcreate** | Inicijalizira uređaj za korištenje od strane LVM, slično kao naredba mkfs za datotečne sustave. |
| **pvdisplay** | Naredba za prikaz detalja o danom fizičkom spremištu. Detalji uključuju kojoj skupini spremišta pripada, koliko je kapacitet fizičkog spremišta i koliko se fizičkog spremišta koristi kao dio nekog logičkog spremišta. |
| **pvmove** | Prebacuju se podaci sa jednog fizičkog spremišta na drugo. Koristi se kada postoji razlog za izbjegavanje ili prestanak korištenje uređaja na kojem je fizičko spremište. |
| **pvremove** | Uklanja podatkovne strukture fizičkog spremišta s uređaja. Suprotna naredba od pvcreate, koristi se tek kada su svi podaci uklonjeni sa fizičkog spremišta i |

| | |
|---|---|
| | kada je fizičko spremište „isključeno“ iz skupine spremišta. |
| **pvresize** | Mijenja se veličina fizičkog spremišta, ova se naredba koristi nakon što je particija na kojoj se fizičko spremište nalazi povećana ili prije smanjivanja particija s fdisk ili parted. |
| **pvs** | Naredba za prikaz detalja o fizičkom spremištu. Daje sažetiji prikaz od pvdisplay. |
| **pvscan** | Pokretanje skeniranja disk particija u potrazi za LVM strukturama podataka. |

Tablica 22 - Naredbe za upravljanje fizičkim spremištem

Pri kreiranju LVM konfiguracija najvažnija naredba od navedenih je `pvcreate`. Iako i ostale navedene naredbe sadrže niz mogućnosti, većina je visoko tehnička i koriste se u posebnim situacijama. `pvcreate` najčešće se koristi sa samo jednim parametrom – imenom diskovnog uređaja nad kojim se kreira PV. U primjeru koji slijedi prvo će se kreirati particija s lvm zastavicom, a zatim će se na njoj kreirati fizičko spremište naredba `pvcreate`.

```
(parted) mkpart
Partition type?  primary/extended? p
File system type?  [ext2]?
Start? 0
End? 1000
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 8590MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:
Number  Start  End     Size    Type     File system  Flags
 1      512B   1000MB  1000MB  primary  ext2         lba
(parted) set 1 lvm on
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 8590MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start  End     Size    Type     File system  Flags
 1      512B   1000MB  1000MB  primary  ext2         lvm, lba

(parted) quit
Information: You may need to update /etc/fstab.

# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created
```

Ova se naredba mora ponoviti za sve particije/uređaje koji će se koristiti u LVM konfiguraciji. U ovom primjeru će se kreirati fizičko spremište nad četiri particije. Particije su različitih veličina:

```
Number  Start   End     Size    Type      File system  Flags
 1      512B    1000MB  1000MB  primary lvm
 2      1000MB  8590MB  7590MB  extended lba
 5      1000MB  1500MB  500MB   logical   ext2 lvm, lba
 6      1500MB  4000MB  2500MB  logical   ext2 lvm, lba
7      4000MB  8590MB  4590MB  logical   ext2 lvm, lba
```

Stanje tih fizičkih spremišta može se provjeriti naredbom `pvdisplay` ili naredbom `pvs`:

```
# pvdisplay
  "/dev/sdb1" is a new physical volume of "953.67 MiB"
  --- NEW Physical volume ---
  PV Name               /dev/sdb1
  VG Name
  PV Size               953.67 MiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               13iJvB-isIR-0Mcr-YNLd-Nqyv-kVCG-MGpvZK

  "/dev/sdb5" is a new physical volume of "476.84 MiB"
  --- NEW Physical volume ---
  PV Name               /dev/sdb5
  VG Name
  PV Size               476.84 MiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               OFyv2r-XSlT-TjMY-dZuo-zqOq-dOTz-KKjruD

  "/dev/sdb6" is a new physical volume of "2.33 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/sdb6
  VG Name
  PV Size               2.33 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               copsJP-jMMw-5QEt-KlVa-GtqG-3i54-v0I2mB

  "/dev/sdb7" is a new physical volume of "4.27 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/sdb7
```

```
  VG Name
  PV Size               4.27 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               16AoGr-Q3O3-9g4r-e8Br-n1LK-UbxY-PZg3fp
# pvs
  PV         VG   Fmt  Attr PSize   PFree
  /dev/sdb1       lvm2 ---  953.67m 953.67m
  /dev/sdb5       lvm2 ---  476.84m 476.84m
  /dev/sdb6       lvm2 ---    2.33g   2.33g
     /dev/sdb7       lvm2 ---    4.27g   4.27g
```

Vidljivo je da `pvs` daje kratak pregled, a `pvdisplay` daje potpuni detaljni pregled svih fizičkih spremišta. Bez parametara obje naredbe javljaju detalje o svim PV dostupnim na sustavu. Moguće je i provjeriti stanje fizičkog spremišta određenog uređaja, tada će naredbe vraćati detalje samo o tom uređaju. Primjerice:

```
# pvs /dev/sdb5
  PV         VG   Fmt  Attr PSize   PFree
     /dev/sdb5       lvm2 ---  476.84m 476.84m
```

Uz dosad navedene, najkorištenije su naredbe `pvmove` i `pvremove`. Budući da su te naredbe korisne samo kada je fizičko spremište već dodijeljeno nekoj skupini spremišta, one će biti prikazane primjerom kasnije te će biti pojašnjeno kada se one koriste.

**Stvaranje i upravljanje sa skupinama spremišta**

Kao i kod fizičkih spremišta, i za upravljanje sa skupinama spremišta ne koristi se jedna naredba s nizom mogućnosti nego niz naredbi koje počinju s vg. Najznačajnije naredbe prikazane su u donjoj tablici:

| Naredba | Objašnjenje |
|---|---|
| **vgcfgbackup** | Kreiranje sigurnosne kopije metapodataka određene skupine spremišta. Zadano se pohranjuje u /etc/lvm/backup. |
| **vgcfgrestore** | Pribavljanje sigurnosne kopije metapodataka za skupinu spremišta, ako nije zadano drugačije pribavlja iz /etc/lvm/backup. |
| **vgchange** | Naredba za izmjenu atributa skupine spremišta, može aktivirati i deaktivirati skupinu spremišta. |
| **vgck** | Provjera metapodataka za skupinu spremišta. Slično fsck za datotečni sustava. |
| **vgconvert** | Pretvaranje metapodataka između stare i nove verzije LVM. Aktualna verzija je LVM2. |
| **vgcreate** | Stvaranje skupine spremišta. Mora biti zadano barem jedno fizičko spremište. |

| | |
|---|---|
| **vgdisplay** | Detaljan prikaz podataka o skupini spremišta. |
| **vgexport** | Omogućavanje prijenosa skupine spremišta na drugo računalo. Ova naredba prikriva skupinu spremišta od lokalnog računala. |
| **vgextend** | Dodavanje fizičkog spremišta u skupinu spremišta. |
| **vgimport** | Naredba suprotna od vgexport, čini skupinu spremišta (ponovno) vidljivom lokalnom računalu. |
| **vgimportclone** | Vrši učitavanje skupine spremišta uz preimenovanje konfliktno imenovanih logičkih spremišta. Koristi se kada se želi pristupiti sigurnosnoj kopiji skupine spremišta, a dok je izvorna kopija aktivna. |
| **vgmerge** | Spaja dvije skupine spremišta, jedna od njih mora biti neaktivna. |
| **vgmknodes** | Kreiranje virtualnih uređaja u direktoriju /dev/ za postojeća logička spremišta. Obično se poziva automatski, a ne ručno. |
| **vgreduce** | Uklanjanje jednog ili više neaktivnih fizičkih spremišta iz skupine spremišta. |
| **vgremove** | Uklanjanje cijele skupine spremišta. Naredba suprotna od vgcreate. |
| **vgrename** | Preimenovanje skupine spremišta. |
| **vgs** | Sažet prikaz podataka o skupini spremišta. |
| **vgscan** | Pretražuje sustav tražeći skupine spremišta. Izvodi se u nekim posebnim okolnostima koje mogu uzrokovati da skupina spremišta nestane ili postane privremeno nedostupna poput zamjene diskova. |
| **vgsplit** | Naredba za dijeljenje jedne skupine spremišta u dvije. |

Tablica 23 - Naredbe za upravljanje skupinama spremišta

Kao što je vidljivo iz tablice, postoje brojne operacije koje se mogu izvoditi nad skupinama spremišta. Najčešće korištene
su **vgchange**, **vgcreate**, **vgdisplay**, **vgextend**, **vgreduce, vgremove** i **vgs**.

Pri kreiranju skupine spremišta standardno se koristi naredba `vgcreate` s dva parametra nazivom skupine spremišta i listom od jednog ili više fizičkih spremišta koja će pripadati toj skupini spremišta. U donjem primjeru koristi se 3 skupine spremišta.

```
# vgcreate l201_1 /dev/sdb1 /dev/sdb5 /dev/sdb6
  Volume group "l201_1" successfully created
# vgs l201_1
  VG       #PV #LV #SN Attr   VSize VFree
  l201_1   3   0   0 wz--n- 3.71g 3.71g
# vgdisplay l201_1
```

```
  --- Volume group ---
  VG Name               l201_1
  System ID
  Format                lvm2
  Metadata Areas        3
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                3
  Act PV                3
  VG Size               3.71 GiB
  PE Size               4.00 MiB
  Total PE              951
  Alloc PE / Size       0 / 0
  Free  PE / Size       951 / 3.71 GiB
     VG UUID               2sM8TW-bcoW-VqeB-XVwL-ymiP-6s3S-wyhVrE
```

Kako je vidljivo u primjeru naredbe za prikaz detalja skupine spremišta - ne prikazuje se koja fizička spremišta su u skupini spremišta. Najjednostavniji način za dobivanje te informacije je pomoću naredbe `pvs`.

```
# pvs
  PV         VG       Fmt  Attr PSize   PFree
   /dev/sdb1  l201_1 lvm2 a--  952.00m 952.00m
   /dev/sdb5  l201_1 lvm2 a--  472.00m 472.00m
   /dev/sdb6  l201_1 lvm2 a--    2.32g   2.32g
   /dev/sdb7           lvm2 ---    4.27g   4.27g
```

Kako je vidljivo iz primjera sva su fizička spremišta dodijeljena u skupinu spremišta imena **l201_1** osim **/dev/sdb7**. Slijedi primjer dodavanja tog novog fizičkog spremišta i uklanjanje /dev/sdb1.

```
# vgextend l201_1 /dev/sdb7
  Volume group "l201_1" successfully extended
# pvs
  PV         VG       Fmt  Attr PSize   PFree
  /dev/sdb1  l201_1 lvm2 a--  952.00m 952.00m
  /dev/sdb5  l201_1 lvm2 a--  472.00m 472.00m
  /dev/sdb6  l201_1 lvm2 a--    2.32g   2.32g
  /dev/sdb7  l201_1 lvm2 a--    4.27g   4.27g
# vgreduce l201_1 /dev/sdb1
  Removed "/dev/sdb1" from volume group "l201_1"
# pvs
  PV         VG       Fmt  Attr PSize   PFree
  /dev/sdb1           lvm2 ---  953.67m 953.67m
```

```
  /dev/sdb5  l201_1 lvm2 a--  472.00m 472.00m
  /dev/sdb6  l201_1 lvm2 a--    2.32g   2.32g
     /dev/sdb7  l201_1 lvm2 a--    4.27g   4.27g
```

Jedini korak koji još preostaje prije osposobljavanja LVM konfiguracije je kreiranje logičkog spremišta.

**Stvaranje i upravljanje s logičkim spremištem**

Kao i za ranije podatkovne strukture tako i za upravljanje s logičkim spremištem postoji niz naredbi čiji nazivi počinje sa lv. U tablici su navedene i objašnjene te naredbe.

| Naredba | Objašnjenje |
|---|---|
| **lvchange** | Naredba mijenja svojstva logičkog spremišta. Moguća svojstva mogu biti neprekinutost ili mogućnost zapisivanja i slično. |
| **lvconvert** | Mijenjanje stanja logičkog spremišta. Moguća stanja su *linear*, *mirror* i *snapshot*. |
| **lvcreate** | Stvaranje logičkog spremišta. |
| **lvdisplay** | Detaljan prikaz podataka o logičkom spremištu. |
| **lvextend** | Povećavanje veličine logičkog spremišta. Ovime se ne povećava automatski i veličina datotečnog sustava na logičkom spremištu. |
| **lvreduce** | Smanjenje veličine logičkog spremišta. Također se ne smanjuje automatski i veličina datotečnog sustava na logičkom spremištu. |
| **lvremove** | Brisanje logičkog spremišta. |
| **lvrename** | Preimenovanje logičkog spremišta. |
| **lvresize** | Promjena veličine logičkog spremišta, kombinirana funkcionalnost **lvextend** i **lvreduce** te isto tako ne utječe automatski na datotečni sustav. |
| **lvs** | Sažet prikaz podataka o logičkom spremištu. |
| **lvscan** | Lociranje logičkih spremišta na svim dostupnim diskovima. |

Tablica 24 - Naredbe za upravljanje logičkim spremištem

Naredba za kreiranje logičkog spremišta `lvcreate`, kao i ostale naredbe u gornjoj tablici, ima velik broj mogućnosti. Za mogućnosti svake pojedine naredbe poželjno je pogledati njene man stranice. Za `lvcreate` je važno da završava sa nazivom skupine spremišta na kojoj se stvara logičko spremište. Jedini parametar koji je još potreban je veličina. U donjem primjeru prikazano je kreiranje logičkog spremišta veličine 1000MB na skupini spremišta imena l201_1.

```
# lvcreate -L 1000M l201_1
 Logical volume "lvol0" created
```

Slijedi prikaz kako je ova naredba kreiranja jednog logičkog spremišta utjecala na stanje fizičkih spremišta, skupina spremišta i logičkih spremišta:

```
# lvdisplay
  --- Logical volume ---
  LV Path                /dev/l201_1/lvol0
  LV Name                lvol0
  VG Name                l201_1
  LV UUID                bscTLY-dfAz-a8ki-Fmjg-711g-AIbM-rOS38n
  LV Write Access        read/write
  LV Creation host, time l201, 2017-01-17 15:38:04 +0100
  LV Status              available
  # open                 0
  LV Size                1000.00 MiB
  Current LE             250
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           254:0

# vgdisplay
  --- Volume group ---
  VG Name               l201_1
  System ID
  Format                lvm2
  Metadata Areas        3
  Metadata Sequence No  8
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               0
  Max PV                0
  Cur PV                3
  Act PV                3
  VG Size               7.06 GiB
  PE Size               4.00 MiB
  Total PE              1807
  Alloc PE / Size       250 / 1000.00 MiB
  Free  PE / Size       1557 / 6.08 GiB
  VG UUID               2sM8TW-bcoW-VqeB-XVwL-ymiP-6s3S-wyhVrE

# pvs
  PV         VG       Fmt  Attr PSize   PFree
  /dev/sdb1           lvm2 ---  953.67m 953.67m
  /dev/sdb5  l201_1 lvm2 a--  472.00m 472.00m
  /dev/sdb6  l201_1 lvm2 a--    2.32g   1.35g
     /dev/sdb7  l201_1 lvm2 a--    4.27g   4.27g
```

U trenutku kada je logičko spremište kreirano spremno je za korištenje preko jednog od minimalno dva virtualna uređaja koji se automatski stvaraju. Prvi virtualni uređaj je u direktoriju **/dev/mapper/**

naziva **<ime_vg><ime_lv>**, a drugi je u direktoriju **/dev/**, u poddirektoriju kojem je naziv jednak nazivu skupine spremišta. Ime datoteke je ime logičkog spremišta. Dakle, u ovom primjeru virtualni uređaji su **/dev/mapper/ l201_1-lvol0** i **/dev/ l201_1/lvol0**. Standardno se u **/dev/mapper/** nalazi prava poveznica na virtualni uređaj, a drugi je virtualni uređaj tek simbolička poveznica na njega. Na nekim distribucijama se kreira pravi virtualni uređaj negdje drugdje u poddirektorijima direktorija /**dev**/, a sve ostalo su simboličke poveznice na taj uređaj.

U trenutku kada je logičko spremište kreirano moguće je s njim raditi kao sa standardnom particijom. Razlika je da se logičko spremište može po potrebi povećati ili smanjiti. Također, moguće je povećati i logičko spremište i na njemu stvoreni datotečni sustav dok je taj datotečni sustav montiran. Ukoliko želimo smanjiti logičko spremište, tada je prvo potrebno smanjiti datotečni sustav. Tu je važno napomenuti da se to ne može raditi dok je datotečni sustav montiran i da je moguće da dođe do gubitka podataka pri smanjivanju datotečnog sustava. Današnji alati ipak dopuštaju da se u nekim scenarijima napravi ponovno montiranje živog sustava, ali ova je funkcionalnost samo za napredne korisnike.
Na primjeru je prikazano ručno montiranje ranije montiranog datotečnog sustava:

```
# lvrename l201_1 lvol0 testni_lv
  Renamed "lvol0" to "testni_lv" in volume group "l201_1"
# df -h|grep test
/dev/mapper/l201_1-lvol0  493M  768K  469M   1% /mnt/test
# umount /mnt/test
# mount /dev/mapper/l201_1-lvol0 /mnt/test
mount: special device /dev/mapper/l201_1-lvol0 does not exist
# ls /dev/mapper/
control  l201_1-testnilv
```

Slijedi prikaz svih mogućnosti na jednom primjeru. Na početku primjera prikazan je datotečni sustav koji je montiran na /mnt/temp i veličine je 1G. Najprije će se logičko spremište i datotečni sustav povećati na 1800M. Za povećanje veličine logičkog spremišta koristi se naredba `lvresize` koja će dodavanjem prefiksa **+** ispred veličine logičkog spremišta izvršiti povećanje za zadanu vrijednost. Ako je prefiks **–**, veličina logičkog spremišta se smanjuje, a ako nema prefiksa, postavlja se veličina na zadanu vrijednost.

```
# df -h|grep test
Filesystem                 Size  Used Avail Use% Mounted on
/dev/mapper/l201_1-lvol0 1009M  1.3M  956M   1% /mnt/test
# lvresize -L 1800M /dev/l201_1/lvol0
  Size of logical volume l201_1/lvol0 changed from 1.00 GiB (256 extents) to
1.76 GiB (450 extents).
  Logical volume lvol0 successfully resized
# resize2fs /dev/mapper/l201_1-lvol0
resize2fs 1.42.12 (29-Aug-2014)
Filesystem at /dev/mapper/l201_1-lvol0 is mounted on /mnt/test; on-line resizing
required
old_desc_blocks = 1, new_desc_blocks = 1
The filesystem on /dev/mapper/l201_1-lvol0 is now 460800 (4k) blocks long.

# df -h|grep test
Filesystem                 Size  Used Avail Use% Mounted on
/dev/mapper/l201_1-lvol0  1.8G  1.5M  1.7G   1% /mnt/test
```

Kao što je vidljivo, veličina dostupnog diskovnog prostora se promijenila i sada je ukupno 1800M. Slijedi prikaz smanjivanja veličine, na 500M.

```
# resize2fs /dev/mapper/l201_1-lvol0 500M
resize2fs 1.42.12 (29-Aug-2014)
Filesystem at /dev/mapper/l201_1-lvol0 is mounted on /mnt/test; on-line resizing
required
resize2fs: On-line shrinking not supported
# umount /mnt/test
# resize2fs /dev/mapper/l201_1-lvol0 500M
resize2fs 1.42.12 (29-Aug-2014)
Please run 'e2fsck -f /dev/mapper/l201_1-lvol0' first.

# e2fsck -f /dev/mapper/l201_1-lvol0
e2fsck 1.42.12 (29-Aug-2014)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/mapper/l201_1-lvol0: 11/120000 files (0.0% non-contiguous), 7920/460800
blocks
# resize2fs /dev/mapper/l201_1-lvol0 500M
resize2fs 1.42.12 (29-Aug-2014)
Resizing the filesystem on /dev/mapper/l201_1-lvol0 to 128000 (4k) blocks.
The filesystem on /dev/mapper/l201_1-lvol0 is now 128000 (4k) blocks long.

# lvresize -L 500M /dev/l201_1/lvol0
  WARNING: Reducing active logical volume to 500.00 MiB
  THIS MAY DESTROY YOUR DATA (filesystem etc.)
Do you really want to reduce lvol0? [y/n]: y
  Size of logical volume l201_1/lvol0 changed from 1.76 GiB (450 extents) to
500.00 MiB (125 extents).
  Logical volume lvol0 successfully resized
# mount /dev/mapper/l201_1-lvol0 /mnt/test
# df -h|grep test
Filesystem                 Size  Used Avail Use% Mounted on
/dev/mapper/l201_1-lvol0  493M  768K  469M   1% /mnt/test
```

Kao što je vidljivo, nije moguće smanjiti datotečni sustav koji je montiran, odnosno radi dodatne sigurnosti `resize2fs` najprije traži provjeru datotečnog sustava. Vidljivo je upozorenje: kada se smanjuje i logičko spremište i datotečni sustav može doći do gubitka podataka!

Na kraju je potrebno napomenuti da je izmjena nad logičkim spremištem nešto što se treba raditi samo kada je to potrebno, zbog toga što čak i promjena naziva može imati nepredviđene posljedice. Npr, ako se promijeni ime **lvol0** u **testni_lv**, mijenjaju se i nazivi virtualnih uređaja u /dev/. Svi montirani datotečni sustavi nastavit će raditi, ali pri idućem pokretanju sustava zapisi u /etc/fstab neće uspješno montirati datotečni sustav.

## 3.3 Izrada CD/DVD medija u Linuxu

### 3.3.1 Izrada medija

Optički mediji dugo su bili popularan način za prijenos relativno velikih datoteka odnosno količina podataka. CD-R i CD-RW mediji imaju kapacitet od 700MB, DVD mediji za zapisivanje imaju kapacitet 4.7-8.5GB, a Blu-ray optički mediji 25-50GB. Snaga ovih medija je njihova pouzdanost i povoljna cijena.

Da bi bilo moguće snimati na optički medij potrebno je stvoriti potpuni datotečni sustav i potom ga snimiti na medij. Zapisivanje na optički medij zahtijeva korištenje dva alata, jedan služi izradi datotečnog sustava za zapisivanje na optički medij, a drugi samom zapisivanju na optički medij.

Alat za kreiranje datotečnog sustava za zapisivanje na optički medij na Debian distribuciji je **genisoimage**. Sve mogućnosti ovog alata i srodnih alata na drugim distribucijama potječu od njegova prethodnika **mkisofs**.

**Napomena**

**mkisofs** je stari naziv genisoimage, ali i dalje postoji simbolička poveznica na **genisoimage** naziva **mkisofs** pa se može koristiti i ta naredba.

Alati za zapisivanje na optički medij dostupni na Debian distribuciji su **wodim** i **cdrskin**. Oba alata imaju iste mogućnost koje su izvorno mogućnosti njihova prethodnika **cdrecorder**. Moguće je i koristiti alat **xorriso** koji koristi mogućnosti **–as mkisofs** i **–as cdrecord** da bi obavljao kreiranje datotečnog sustava ili zapisivanje na medij respektivno. Obje funkcije može obavljati i growisofs alat, ali samo na DVD i Blu-ray medijima.

Izuzetak pravila o kreiranju datotečnog sustava prije zapisivanja na optički medij je pisanje i čitanje UDF optičkih medija, o čemu će se govoriti kasnije.

Zapisivanje na optički medij sastoji se od 3 koraka:

- prikupljanje datoteka koje se zapisuju na jednoj lokaciji - ta lokacija je proizvoljni direktorij.
- kreiranje datotečnog sustava – program **genisoimage** treba direktorij iz prethodnog koraka kako bi kreirao **ISO-9660 datotečni sustav** u obliku datoteke.
- datoteka koju je kreirao **genisoimage** snima se na optički medij korištenjem alata za zapisivanje („prženje“) poput alata **wodim**. To je zapravo kopiranje datoteke stvorene u prethodnom koraku, a na kojoj se nalazi odgovarajući datotečni sustav.

Alat **growisoimage**, koji se standardno nalazi u **dvd+rw-tools** paketu, ujedinjuje funkcionalnosti od alata **genisofs** i **cdrecorder**, ali ne radi s manjim optičkim medijima, odnosno sa CD-R i CD-RW medijima. S druge strane, neke (starije) verzije wodim alata ne rade s DVD i Blu-ray optičkim medijima.

**Primjer zapisivanja na optički medij**

Za primjer će biti prikazano stvaranje kopije sadržaja Desktop direktorija na optičkom mediju. Najprije je potrebno kopirati željeni sadržaj u direktorij koji će se koristiti za stvaranje datoteke za zapis na disk.

```
# mkdir -p /tmp/snimanje/desktop
# cp -r ~/Desktop/ /tmp/snimanje/desktop/
```

Nakon što su prikupljene sve datoteke na jednom mjestu, može se s jednom naredbom `growisofs` pokrenuti zapisivanje na optički medij:

```
growisofs -speed=4-Z/dev/dvdrw -J -r -V"DVD snimljeni"
/tmp/snimanje/desktop/
```

Mogućnosti koje su korištene su objašnjene u donjoj tablici.

| Mogućnost | Objašnjenje |
|---|---|
| **-speed** | Definira brzinu zapisivanja. |
| **-J** | Aktivira **Joliet** format. |
| **-r** | Aktivira **Rock Ridge** format. |
| **-V** | Definira ime medija. |
| **/tmp/przenje/desktop/** | Definira gdje se nalaze sve datoteke koje će se kopirati na optički medij. |

Tablica 25 - Korištene mogućnosti naredbe growisofs

**Primjer kreiranja optičkog medija čitljivog na više platforma**

Prethodni primjer će kreirati optički medij koji je čitljiv na svim *Linux* distribucijama, ali da bi se omogućilo da se optičkom mediju može pristupiti s velikog broja raznovrsnih operacijskih sustava i platformi, poželjno je instalirati ga s velikim brojem datotečnih sustava i njihovim proširenjima (engl. *extensions*). Podaci u tom slučaju neće biti umnoženi. Svaka datoteka iz izvorišnog direktorija postojat će samo jednom na optičkom disku, ali će postojati više datotečnih sustava koji pokazuju na tu datoteku. Na ovaj je način dodatna potrošnja diska minimalna.

Svojstva koja treba sadržavati optički medij za više platformi su:

- Mogućnost **–f** od **genisoimage** ili **growisofs** koja uzrokuje praćenje simboličkih poveznica. Na taj se način ne kopiraju doslovno simboličke poveznice, već se umjesto njih na optički disk smješta datoteka na koju te simboličke poveznice pokazuju. Može značajno povećati veličinu zapisa.
- Dugački ISO-9660 nazivi datoteka – za ISO-9660 **genisoimage** standardno koristi kratke nazive datoteka, ali može se **opcijom –l** zadati da koristi duge (do 31 znaka) nazive.
- **Joliet** format aktivira se **opcijom –J**

- **Rock Ridge** format aktivira se **opcijom –R**
- **UDF** format aktivira se **opcijom –udf**
- **HFS** format aktivira se **opcijom –hfs**

„Manje je bolje“ pristup nije nužno najbolji za optičke medije pa je preporučljivo kod zapisivanja sigurnosne kopije koristiti podršku za što širim spektrom formata. Tako će postojati mogućnost da će jedan od korištenih formata još uvijek biti podržan za 10 (i više) godina.

**Čitanje i zapisivanje UDF optičkih medija**

Ranije navedeno pravilo da se na optičkom mediju mora nalaziti potpuni datotečni sustav ne odnosi se na UDF optičke diskove. Najprije je potrebno napomenuti da nije dovoljno samo postaviti **–udf** opciju u naredbi `genisoimage`. U tom će se slučaju UDF stvoriti samo kao dodatak ISO-9660 datotečnog sustava.

Kako bi se omogućio potpuni pristup za čitanje i zapisivanje na optički medij, poput USB prijenosnih memorija, potrebno je koristiti jezgru s podrškom za UDF i podrškom za zapisivanje paketa (engl. *packet writing*) te paket **udftools**. S tim softverom i navedenom podrškom jezgre moguće je montirati DVD+RW medije kao da su u pitanju standardni diskovni uređaji. Za razliku od DVD+RW medija, CD-RW mediji ne mogu se tako montirati ni s navedenim softverom i postavkama.

U trenutku kada je sva navedena podrška prisutna jednostavno se izvrši naredba `mkudffs` nad željenim uređajem, primjerice:

```
# mkudffs /dev/dvdrw
```

I nakon te naredbe moguće je montirati optički disk i raditi sa sadržajem na njemu kao da je običan disk kapaciteta 4.7 ili 8.5 GB. Zapisivanje na optički disk ipak je puno sporije nego što je na druge medije, pa treba očekivati spor rad pri kopiranju velikih količina podataka.

Poput **ISO-9660** s **Rock Ridge podrškom** i UDF standard podržava *Linux* standard vlasništva i dozvola nad datotekama. Stoga je važno postaviti odgovarajuće dozvole nad datotekama , u slučaju da će se optički medij koristiti na drugim *Linux* računalima, odnosno da će mu pristupati drugi korisnici.

### 3.3.2 Izrada medija za pokretanje operacijskog sustava

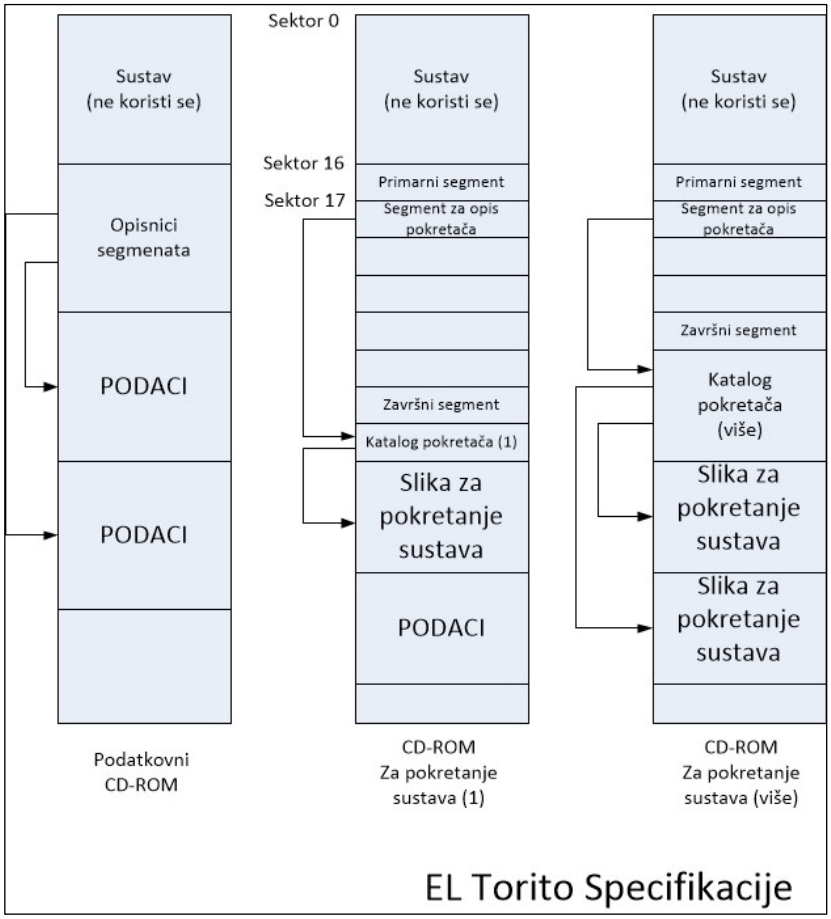
Kako bi **BIOS** (engl. *basic input/output system*) mogao pokrenuti sustav koristeći **CDROM 1995**, Phoenix Technologies i IBM su razvili proširenje ISO-9660 specifikacije naziva **El Torito**. Ova specifikacija omogućava da BIOS pristupa medijima koristeći emulator hard diska ili disketne jedinice.

Tri tipa CD-ROM konfiguracija prikazana su na donjoj slici: podatkovni CD-ROM, CD-ROM za pokretanje sustava i CD-ROM za pokretanje sustava, koji sadrži više slika za pokretanje sustava.

Osnovna ISO 9660 specifikacija podržava samo pohranu datoteka i specificira konfiguraciju prikazanu u prvom stupcu na slici. Specifikacija **El Torito** napisana je tako da omogućava

pohranjivanje slike za pokretanje sustava na način da se na optičkom mediju ne kosi sa standardom za pohranu podataka. Tako optički medij može sadržavati podatke kojima se pristupa standardno, iako sadrži i sliku za pokretanje sustava. El Torito uvodi segment za opis pokretača sustava (engl. *Boot Record Volume*) u koji se upisuju sve informacije koje sustav treba za lociranje slika za pokretanje sustava. Ovaj je segment čitan od strane BIOS sustava koji podržavaju El Torito. Slike za pokretanje sustava su samo tada dostupne - kada se optički medij montira na uobičajen način, tada se podaci slike za pokretanje sustava ne vide.

Kao što je prikazano na slici 3, BIOS prvo čita **sektor 17**. U tom sektoru nalazi se pokazivač na katalog pokretača sustava. U katalogu se nalaze zapisi za jedan ili više slika za pokretanje sustava. Svaki zapis sadrži pokazivač na prvi sektor i veličinu slike za pokretanje sustava. Nekada se standardno ta slika kopirala na emulirani disk ili emuliranu disketnu jedinicu. Stari programi za učitavanje sustava mogli su bolje manipulirati s tim emuliranim uređajima. Danas se standardno koristi pristup bez emulacije (engl. *no-emulate mod*). U tom se pristupu slika za pokretanje učita direktno u memoriju na odgovarajuću adresu (0x7c00), a ostali sadržaj diska se zatim može dalje koristiti po potrebi.




Slika 3 - El Torito specifikacije

ISOLINUX pokretač sustava koristi **El Torito** proširenje ISO 9660 standarda u pristupu bez emulacije. Korištenjem ovog pristupa izbjegavaju se dva problema, a to su problem nedostatka prostora koji nastaje pri emulaciji disketne jedinice i problem nekompatibilnosti koji se pojavljuje pri emulaciji hard diska. **isolinux** paket instalira isolinux.bin pokretač sustava. Ovisno o distribuciji

smješta ga u **/usr/lib/ISOLINUX/** ili **/usr/share/ISOLINUX/**. Za kreiranje medija za pokretanje sustava potrebna je i datoteka jezgre i inicijalnog RAM datotečnog sustava.

Kreiranje CD–a za pokretanje sustava:

1. Stvoriti direktorij za rad:

```
# mkdir -p /work/boot-cd
```

2. Kopirati potrebne datoteke:

```
# cp /boot/initrd.img-3.13-1-486  /work/boot-cd/
# cp /usr/lib/ISOLINUX/isolinux.bin /work/boot-cd/
# cp /boot/initrd.img-3.13-1-486  /work/boot-cd/
# cp /boot/vmlinuz-3.13-1-486  /work/boot-cd/
```

3. Stvoriti i urediti konfiguracijsku datoteku **/work/boot-cd/isolinux.cfg** sljedećeg sadržaja:

```
DEFAULT linux
LABEL  linux
KERNEL vmlinuz
APPEND  initrd=initrd root=/dev/sda1
```

4. Stvoriti disk za pokretanje sustava:

```
root@l201:/work/boot-cd# genisoimage -r -V "l201_boot" -cache-inodes --
joliet -L -b isolinux.bin -c boot.cat -no-emul-boot -boot-load-size 4 -
boot-info-table -o l201_boot.iso .
```

5. Stvorena datoteka snima se na CD standardno kao i svaka druga datoteka koja već sadržava datotečni sustav. Mogu se koristiti bilo koji od navedenih alata. Od naredbi navedenih u donjem primjeru koristi se samo jedna:

```
# wodim -v dev=/dev/sr0 -dao ./l201_boot.iso
# cdrskin -v dev=/dev/sr0 -dao ./l201_boot.iso
# xorriso -as cdrecord -v dev=/dev/sr0 -dao ./l201_boot.iso
```

U završnom koraku primjera može se vidjeti da se mogućnosti naredbi ne razlikuju ni u obliku ni značenju, budući da su naslijeđene iz iste naredbe.

## 3.4 Upravljanje uređajima u Linuxu

### 3.4.1. Upravljanje uređajima

Uz prave datotečne sustave koji pružaju pristup fizičkim uređajima koriste se i virtualni datotečni sustavi. Najznačajniji takav sustav je **udev**, koji upravlja datotekama u direktoriju **/dev**. Datoteke u direktoriju **/dev** pružaju pristup hardveru na računalu, od tipkovnice do hard diskova.

**Napomena**

Zapisi u **/dev** direktoriju ne kreiraju se za sve uređaje. Najznačajnija iznimka su mrežni uređaji. udev upravlja s tim uređajima, na način da se pomoću tog sustava mogu preimenovati postojeći uređaji i definirati imena uređaja na konzistentan način, odnosno tako da je ime konzistentno pri ponovnom pokretanju sustava.

Pri pokretanju sustava jezgra skenira sve dostupne kanale da utvrdi koji je sve hardver dostupan. Zatim **udev** stvara sve zapise u **/dev** virtualnom datotečnom sustavu, koji su poveznica na većinu uređaja. Ovaj pristup omogućava jednostavno i pregledno pristupanje uređajima, čak i nakon izmjena hardvera te zadržava direktorij **/dev** relativno čistim. Prije korištenja udev sustava *Linux* sustavi kreirali su pred-definirane zapise u **/dev** direktoriju, za sve standardne uređaje bez obzira jesu li bili prisutni.

Postoje standardi imenovanja uređaja u direktoriju **/dev**. Tablica u nastavku donosi dio tih standarda. U tablici se koriste 3 oznake koje treba razjasniti:

- <s> - označava jedno slovo (obično dodijeljena slova kreću od slova a na dalje, po abecedi)
- <B> - označava jednu znamenku (obično dodijeljeni brojevi kreću od 1)
- <Rij> - označava nazive datoteka ili dodatnih poddirektorija

Treba napomenuti da ova tablica nije potpuna, te se neki od unosa razlikuju do neke mjere kod različitih distribucija.

| Predložak putanje | Objašnjenje |
|---|---|
| **/dev/sd<s>** | Cijeli disk kojem se pristupa putem **SCSI** (engl. *Small Computer System Interface*) sabirnice. Danas i neki drugi diskovi koriste ovo označavanje. |
| **/dev/sd<s><B>** | Particija na redak ranije navedenom uređaju na SCSI sabirnici. |
| **/dev/hd<s>** | Cijeli disk ili optički disk kojem se pristupa preko IDE (engl. *Integrated Device Electronics*) sabirnice. Danas se rijetko koristi. |
| **/dev/hd<s><B>** | Particija na redak ranije navedenom uređaju na IDE sabirnici. |
| **/dev/fd<B>** | Disketna jedinica. |
| **/dev/sr<B>** | Uređaj za pristup optičkim medijima. Simboličke poveznice /dev/cdrom, /dev/cdrw, /dev/dvd, i /dev/dvdrw također često postoje |

| | |
|---|---|
| **/dev/lp<B>** | Paralelni port. |
| **/dev/usb/lp<B>** | USB pisač. |
| **/dev/ttyS<B>** | RS-232 serijski port. |
| **/dev/tty<B>** | Tekstualna login konzola. |
| **/dev/pts/<B>** | Korisnička sjednica u tekstualnom modu. Najčešće se radi o sjednici udaljenog pristupa, tekstualnoj korisničkoj sjednici u grafičkom sučelju ili o nekom drugom pristupu za login. |
| **/dev/bus/usb/<Rij>** | U ovom se direktoriju nalaze USB uređaji. |
| **/dev/snd/<Rij>** | Hardver za zvuk. |
| **/dev/input/<Rij>** | Uređaji za komunikaciju korisnika s računalom. Primarno se radi o mišu u /dev/input/mice. |
| **/dev/zero** | Virtualni uređaj koji generira beskonačan niz nula kada se čita iz njega. |
| **/dev/null** | Virtualni ponor. Ovaj uređaj čita sve što mu se pošalje i ne čini ništa sa tim podacima. Podaci poslani na /dev/null jednostavno nestaju. |

Tablica 26 - Standardni uređaji u /dev direktoriju

Većina uređaja u **/dev** komuniciraju na razini znakova, što znači da primaju informacije znak po znak (bajt, engl. *byte*). Tipični uređaji koji tako funkcioniraju su miš, konzole i portovi za pisače. Neki uređaji komuniciraju na razini **blokova** (blok uređaji, engl. *block devices*), a to znači da je sa njih moguće čitanje i zapisivanje samo u blokovima od više bajtova. Tipični uređaji koji tako komuniciraju su diskovi koji standardno komuniciraju u blokovima od 512 bajtova, a neki i s većim blokovima.

Virtualne datoteke u direktoriju **/dev** imaju vlasništvo i pravila prava pristupa kao i sve ostale datoteke na *Linux* sustavu. Te ovlasti definiraju tko može pristupiti kojem uređaju i na koji način. Važna mogućnost **udev** sustava je njegova mogućnost da te ovlasti izmijeni i prilagodi potrebama sustava.

### Izgled udev pravila

U direktoriju **/etc/udev/rules.d** nalaze se datoteke koje definiraju **udev** pravila za uređaje. Imena datoteka su <broj>-<opis>.rules. Broj definira slijed kojim udev učitava pravila pri pokretanju sustava ili kada se određeni hardver priključi na računalo.

Naredba `udevadm` može se koristiti za prikupljanje određenih parametara uređaja. Te je parametre potrebno znati kako bi se moglo postaviti valjane definicije za određeni uređaj. Naredba je vrlo kompleksna kao i pravila vezana za pojedine uređaje. Ova složenost vidljiva je na sljedećem primjeru:

```
# udevadm info -a -p $(udevadm info -q path -n /dev/input/mouse1)

Udevadm info starts with the device specified by the devpath and then
```

```
walks up the chain of parent devices. It prints for every device
found, all possible attributes in the udev rules key format.
A rule to match, can be composed by the attributes of the device
and the attributes from one single parent device.
  looking at device '/devices/pci0000:00/0000:00:06.0/usb1/1-1/1-
1:1.0/0003:80EE:0021.0001/input/input3/mouse1':
    KERNEL=="mouse1"
    SUBSYSTEM=="input"
    DRIVER==""
  looking at parent device '/devices/pci0000:00/0000:00:06.0/usb1/1-1/1-
1:1.0/0003:80EE:0021.0001/input/input3':
    KERNELS=="input3"
    SUBSYSTEMS=="input"
    DRIVERS==""
    ATTRS{name}=="VirtualBox USB Tablet"
    ATTRS{phys}=="usb-0000:00:06.0-1/input0"
    ATTRS{uniq}==""
    ATTRS{properties}=="0"
  looking at parent device '/devices/pci0000:00/0000:00:06.0/usb1/1-1/1-
1:1.0/0003:80EE:0021.0001':
    KERNELS=="0003:80EE:0021.0001"
    SUBSYSTEMS=="hid"
    DRIVERS=="hid-generic"
  looking at parent device '/devices/pci0000:00/0000:00:06.0/usb1/1-1/1-1:1.0':
    KERNELS=="1-1:1.0"
    SUBSYSTEMS=="usb"
    DRIVERS=="usbhid"
    ATTRS{bInterfaceClass}=="03"
    ATTRS{bInterfaceSubClass}=="00"
    ATTRS{bInterfaceProtocol}=="00"
    ATTRS{bNumEndpoints}=="01"
    ATTRS{supports_autosuspend}=="1"
    ATTRS{bAlternateSetting}==" 0"
    ATTRS{bInterfaceNumber}=="00"

  looking at parent device '/devices/pci0000:00/0000:00:06.0/usb1/1-1':
    KERNELS=="1-1"
    SUBSYSTEMS=="usb"
    DRIVERS=="usb"
    ATTRS{bDeviceSubClass}=="00"
    ATTRS{bDeviceProtocol}=="00"
    ATTRS{devpath}=="1"
    ATTRS{idVendor}=="80ee"
    ATTRS{speed}=="12"
    ATTRS{bNumInterfaces}==" 1"
    ATTRS{bConfigurationValue}=="1"
    ATTRS{bMaxPacketSize0}=="8"
    ATTRS{busnum}=="1"
    ATTRS{devnum}=="2"
    ATTRS{configuration}==""
    ATTRS{bMaxPower}=="100mA"
    ATTRS{authorized}=="1"
    ATTRS{bmAttributes}=="80"
    ATTRS{bNumConfigurations}=="1"
```

```
    ATTRS{maxchild}=="0"
    ATTRS{bcdDevice}=="0100"
    ATTRS{avoid_reset_quirk}=="0"
    ATTRS{quirks}=="0x0"
    ATTRS{version}==" 1.10"
    ATTRS{urbnum}=="14"
    ATTRS{ltm_capable}=="no"
    ATTRS{manufacturer}=="VirtualBox"
    ATTRS{removable}=="unknown"
    ATTRS{idProduct}=="0021"
    ATTRS{bDeviceClass}=="00"
    ATTRS{product}=="USB Tablet"
  looking at parent device '/devices/pci0000:00':
    KERNELS=="pci0000:00"
    SUBSYSTEMS==""
    DRIVERS==""
#
```

Dio naredbe koji se prvi izvršava je dio omeđen zagradama: **„udevadm info -q path -n /dev/input/mouse1“** – ovdje se traži putanja do uređaja koji je definiran s **/dev/input/mouse1**. Radi se o putanji koja se koristi za upite ili pretraživanje atributa uređaja. U „vanjskom“ se dijelu naredbe ta prikupljena putanja koristi kako bi se prikazali svi atributi uređaja i svi atributi svih uređaja u lancu koji su roditeljski elementi odnosno preci danom uređaju.

**Kreiranje i izmjena udev pravila**

Kada želimo izmijeniti neko **udev** pravilo, prvo je potrebno pronaći svojstvo koje je jedinstveno za ciljani uređaj. Taj je parametar idealno **ATTRS{name}**, ali ako nije moguće koristiti taj parametar moguće je koristiti (gdje postoje) serijski broj uređaja, ime upravljačkog programa, identifikacijski kôd dodijeljen od proizvođača i slično. Pravila u **udev** datotekama konfiguracije sastoje se od niza parova **ključ/vrijednost** odvojenih zarezom. Između ključa i vrijednosti nalazi se operator koji može biti operator uvjetovanja ili operator pridruživanja. Operator uvjetovanja definira kada se određeno pridruživanje izvršava.

U donjoj su tablici navedeni operatori s klasifikacijom:

| Operator | Tip | Objašnjenje |
|---|---|---|
| == | Uvjetovanje | Provjerava jednakost. |
| != | Uvjetovanje | Provjerava nejednakost. |
| = | Pridruživanje | Dodjeljuje vrijednost ključu, zamjenjujući staru vrijednost novom. |
| += | Pridruživanje | Dodaje vrijednost u set postojećih vrijednosti ključa. |
| := | Pridruživanje | Dodjeljuje vrijednost ključu, onemogućuje buduće izmjene. |

Tablica 27 - Operatori za udev konfiguracijske datoteke

Postoje brojni **udev** ključevi. Kada se definira ključ u uvjetovanju može se koristiti specijalne znakove tako da ***** odgovara bilo kojem nizu znakova, **?** mijenja jedan (bilo koji) znak, a unutar uglatih zagrada može se dati više znakova od kojih se prihvaća jedan; na primjer [agh] prihvaća bilo koji od tri znaka “a“, “g“ ili “h“ . U donjoj tablici prikazani su samo oni ključevi koji će biti korišteni u kasnijem primjeru. Za potpuni popis mogućih ključeva potrebno je proučiti **udev** man stranice.

| Ključ | Tip | Objašnjenje |
|---|---|---|
| **SUBSYSTEM** | Uvjetovanje | Naziv podsustava u jezgri, poput **sound** ili **net**. |
| **ACTION** | Uvjetovanje | Naziv akcije koju je **udev** poduzeo, poput dodavanja uređaja (*add*). |
| **GOTO** | Uvjetovanje | Naredba za preusmjeravanje tijeka izvršavanja pravila na drugi dio datoteke. |
| **ATTR{niz}** | Uvjetovanje ili pridruživanje | Proizvoljni naziv koje se može postaviti za uređaj. |
| **SYMLINK** | Pridruživanje | Imenovanje simboličke poveznice koja se kreira (relativna putanja prema /dev direktoriju). |
| **MODE** | Pridruživanje | Dozvole nad datotekom uređaja. |
| **OWNER** | Pridruživanje | Vlasništvo nad datotekom uređaja. |
| **GROUP** | Pridruživanje | Pripadnost grupi datoteke uređaja. |
| **LABEL** | Upravljanje | Jedinstvena oznaka u datoteci na koju se može referencirati. |
| **KERNEL** | Uvjetovanje | Naziv podsustava u jezgri, poput **sd*** za SCSI diskove. |
| **NAME** | Pridruživanje | Naziv datoteke uređaja koji se kreira, relativno prema /dev. (Izuzetak su mrežni uređaji za koje naziv postoji ali se ne kreira datoteka uređaja u /dev). |

Tablica 28 - udev ključevi

Primjer pravila smjestit će se u **/etc/udev/rules.d/99-moja_pravila.rules** datoteku. Standardno se ne vrše izmjene nad postojećim udev pravilima, budući da se te datoteke mijenjaju pri nadogradnji paketa. Postavljamo sljedeći sadržaj:

```
#(1)
SUBSYSTEM!="usb_device", ACTION!="add", GOTO="minolta_rules_end"
#(2) Minolta|DiMAGE Scan Elite 5400
ATTR{idVendor}=="0686", ATTR{idProduct}=="400e", SYMLINK+="scan5400"
MODE="0660", OWNER="lisa", GROUP="scanner"
LABEL="minolta_rules_end"
# (3)PCI device 0x8086:0x1030 (e100)
SUBSYSTEM=="net", ACTION=="add", ATTR{address}=="00:03:47:b1:e3:d8",
KERNEL=="eth*", NAME="eth0"
# (4)PCI device 0x10ec:0x8168 (r8169)
SUBSYSTEM=="net", ACTION=="add", ATTR{address}=="00:e0:4d:a3:22:c5",
KERNEL=="eth*", NAME="eth1"
```

Prva linija sa **goto pravilom** definira da se druga naredba izvršava samo kada se dodaje **usb uređaj**. Druga linija dodjeljuje ovlasti i vlasništvo nad datotekom koja povezuje skener. Može se uočiti da je **SYMLINK ključ** proširen sa += budući da to nije kôd ključeva moguće prava pristupa (MODE), vlasništvo (OWNER) i pripadnost grupi (GROUP) nisu proširenu već su definirani.

Treća i četvrta linija su slične i definiraju nazive za određene mrežne uređaje. Ova pravila koriste MAC adresu kao ključ za postavljanje naziva i na taj način osiguravaju da će iste kartice imati isti naziv na sustavu. Osim ove primjene da se osigura konzistentno imenovanje uređaja na osnovu nepromjenjivih parametara, **udev** pravila mogu se koristiti za postavljanje smislenih i ljudski razumljivih naziva. Moguće je primjerice u prvoj liniji postaviti alias tako da je datoteka uređaja u **/dev** direktoriju **/dev/skener_soba_4**.

**Testiranje rada udev**

Testiranje udev pravila ne zahtijeva ponovno pokretanje sustava. Pravila koja se smjeste u direktorij **/etc/udev/rules.d/** postaju aktivna čim je datoteka snimljena. Biti će potrebno ugasiti i upaliti uređaj ili ako to nije moguće, hardverski ili softverski ponovno inicijalizirati vezu prema uređaju. Ako ni jedan od tih pristupa nije moguć, onda se primjena novih pravila ponekada može prisilno pokrenuti, ali je potrebno ukloniti i ponovno učitati odgovarajuće module jezgre. Ako ni jedan od tih pristupa ne pokazuje rezultate, onda može biti nužno ponovno pokretanje sustava kako bi testirali novu **udev** konfiguraciju.

Ako se pojavi problem u radu ili uređaj ne radi na željeni način moguće je pomoću alata **udevadm** s mogućnosti **monitor** prikupiti podatke o aktivnostima konfiguriranog uređaja. Na taj se način prikupljaju svi događaji vezani uz aktivnosti jezgre i **udev** sustava. Primjer izlaza naredbe:

```
root@l201:/work/boot-cd# udevadm monitor
monitor will print the received events for:
UDEV - the event which udev sends out after rule processing
KERNEL - the kernel uevent

KERNEL[33203.775626] add      /module/pcmcia_core (module)
KERNEL[33203.782694] add      /class/pcmcia_socket (class)
KERNEL[33203.785315] add      /module/pcmcia (module)
KERNEL[33203.788646] add      /bus/pcmcia (bus)
UDEV  [33203.791918] add      /module/pcmcia_core (module)
KERNEL[33203.797331] add      /module/snd_vx_lib (module)
UDEV  [33203.805040] add      /module/pcmcia (module)
UDEV  [33203.808354] add      /module/snd_vx_lib (module)
UDEV  [33203.809551] add      /class/pcmcia_socket (class)
KERNEL[33203.815378] add      /module/snd_vxpocket (module)
KERNEL[33203.815443] add      /bus/pcmcia/drivers/snd-vxpocket (drivers)
UDEV  [33203.817667] add      /module/snd_vxpocket (module)
UDEV  [33203.824716] add      /bus/pcmcia (bus)
UDEV  [33203.826615] add      /bus/pcmcia/drivers/snd-vxpocket (drivers)
```

```
KERNEL[33221.088889] change
/devices/pci0000:00/0000:00:01.1/ata4/host3/target3:0:0/3:0:0:0/block/sr
0 (block)
UDEV  [33221.218339] change
/devices/pci0000:00/0000:00:01.1/ata4/host3/target3:0:0/3:0:0:0/block/sr
0 (block)
```

Ovaj primjer opisuje dva događaja. Zadnje dvije linije opisuju montiranje optičkog medija, a sve ostale linije učitavanje modula **snd-vxpocket** u jezgru. Slični zapisi stvaraju se i pri dodavanju uređaja i aktiviranju **udev** pravila, koja je posljedica dodavanja uređaja.

## 3.5 Vježba: RAID

1. Prije početka rada odaberite sliku stanja virtualnoga računala (engl.Snapshot) za početak ove vježbe.
2. Prijavite se na računalo kao korisnik l201. U GUI-ju pokrenite **Terminal** (**Activities** → **Terminal**). Izvršite su - naredbu da postanete administrator (lozinka: L201).
3. Kreirajte 6 novih particija veličine 1 GB na **/dev/sdb**, koristeći **parted**. Prvo morate postaviti oznaku (engl. label) s **mklabel** u vrijednost **msdos**. Budući da koristimo više od 4 particije, prvo je potrebno napraviti jednu ***extended*** particiju veličine preko 7,6 GB, a u njoj stvoriti 6 logičkih particija.
4. Postavite za sve particije zastavicu RAID.
5. Kreirajte RAID polje tipa 5 koristeći prve tri particije koje ste kreirali.
6. Kreirajte **ext4** datotečni sustav na RAID polju kreiranom u prethodnom zadatku.
7. Izradite direktorij **/mnt/raid** i na njega montirajte datotečni sustav iz prethodnog zadatka.
8. Provjerite kapacitet montiranih datotečnih sustava. Što uočavate? Zašto je to tako?

   _____

   _____

9. Provjerite stanje RAID polja naredbom **mdadm --detail <RAID uređaj>**.

   _____

10. Dodajte sve ostale uređaje u RAID polje.

    _____

11. Ponovno provjerite kapacitet montiranih datotečnih sustava i stanje RAID polja. Što uočavate?

    _____

    Zašto je stanje takvo?

    _____

12. Povećajte veličinu datotečnoga sustava na maksimalni kapacitet. Ponovno provjerite veličine montiranih uređaja.

    _____

13. Kreirajte datoteku u direktoriju **/mnt/raid**.

    _____

14. Uklonite drugi disk iz RAID polja. Prvo ga morate proglasiti neispravnim, koristeći *--fail*, a onda ga je moguće ukloniti. Poslije svake naredbe provjerite stanje RAID polja:

15. Napravite novu particiju veličine 1500 M na **/dev/sdb**, postavite RAID zastavicu.

16. Dodajte novi disk u RAID polje. Čim uspijete, pokrenite naredbu `watch cat /proc/mdstat` ([ctrl]+[c] za izaći).

17. Što se dogodilo? Kako sada izgledaju kapaciteti datotečnih sustava i stanje RAID polja?

## 3.6 Vježba: Softverski spremišni sustav

1. Prije početka rada odaberite sliku stanja virtualnoga računala (engl. Snapshot) za početak ove vježbe.
2. Prijavite se na računalo kao korisnik l201. U GUI-ju pokrenite **Terminal** (**Activities** → **Terminal**). Izvršite su - naredbu da postanete administrator (lozinka: L201).
3. Na disku /dev/sdb/ stvorite tri particije veličina 100, 500, i 200 MB. Prvo morate postaviti oznaku (engl. label) s **mklabel** u vrijednost **msdos**.

4. Stvorite fizička spremišta nad tim particijama.

5. Provjerite stanje fizičkih spremišta.

6. Stvorite novu skupinu spremišta (dodajte u skupinu spremišta prvu i zadnju particiju).

7. Provjerite stanje fizičkih spremišta i skupina spremišta. Koja je razlika u izlazima dviju naredbi za pregled stanja skupina spremišta?

8. Na skupini spremišta stvorite logičko spremište veličine 250 MB, bi li isto bilo moguće s particijama, zašto?

9. Provjerite stanje naredbom za provjeru stanja logičkih spremišta.

10. Na logičkom spremištu stvorite novi datotečni sustav (ext4).

11. Montirajte sustav na direktorij **/mnt/lvm** (stvorite taj direktorij) i provjerite kapacitet.

12. Povećajte logičko spremište na maksimalnu trenutno dostupnu veličinu.

13. Povećajte i pripadajući datotečni sustav.

14. Odmontirajte datotečni sustav. Promijenite veličinu datotečnoga sustava na 200 M.

(izvršite provjeru datotečnoga sustava ako sustav to zatraži.)

15. Promjenite veličinu logičkoga spremišta na 100 M i montirajte datotečni sustav.

16. Koliki je kapacitet datotečnoga sustava, zašto?

# 4. Upravljanje i nadzor sustava

**Trajanje poglavlja:**
**375 min**

Po završetku ove cjeline moći ćete:

- razlikovati syslog, rsyslog i syslog-ng
- namjestiti slanje sistemskih zapisa na udaljeno računalo
- namjestiti konfiguraciju proizvoljnog servisa za upravljanje sistemskim zapisima
- izraditi osnovne Debian pakete
- brzo kreirati nekoliko korisnih tipova paketa
- razumjeti osnove mogućnosti paketa i osnove procesa instalacije i deinstalacije
- razumjeti ograničenja i mogućnosti skripta i programskih jezika
- koristiti postojeće Perl module
- instalirati nove Perl module
- ručno provjeriti stanje aktivnih procesa naredbama **ps**, **top** i **htop**
- namjestiti i testirati rad automatskog nadzora
- instalirati i namjestiti alat za automatski nadzor

U ovoj se cjelini obrađuju alati i servisi koji omogućavaju nadgledanje rada *Linux* sustava. U cjelini je obrađena i izrada programskih paketa, kao i osnova rada s Perlom. Cjelina završava s uvidom u automatsko nadziranje stanja procesa i servisa.

## 4.1. Upravljanje sistemskim zapisima

### 4.1.1. Servis syslog

Syslog je razvijen 1980. godine kao dio Sendmail projekta. Spremno je prihvaćen od drugih aplikacija i postao je standard za upravljanje sistemskim porukama na *Linux* sustavima. Syslog je izvorno funkcionirao kao *de facto standard* i postojale su brojne implementacije, ali često nepotpune. To stanje je ispravljeno kada je IETF dokumentirao taj standard u **RFC 3164**. Potpuna standardizacija provedena je u **RFC 5424**.

Središnja konfiguracijska datoteka je **/etc/syslog**. U datoteci se standardno nalazi niz pravila oblika:

```
tipX.razinaX; tipY.razinaY  <odredište_poruke>
```

Tip može imati jednu od ovih vrijednosti:

| Tip | Objašnjenje |
|---|---|
| **auth** | Autentikacija na sustavu. |
| **authpriv** | Privatna autentikacija. |
| **cron** | Poruke **cron** pozadinskog procesa. |
| **daemon** | Poruke ostalih pozadinskih procesa. |
| **kern** | Poruke jezgre. |
| **lpr** | Podsustav za linijske pisače. |
| **mail** | Poruke servisa elektroničke pošte. |
| **mark** | Interne poruke (samo za testiranje). |
| **news** | Podsustav network news. |
| **security** | Zastarjelo – treba koristiti **auth.** |
| **syslog** | Poruke o aktivnostima središnjeg upravitelja datotekama sistemskih zapisa sustava. |
| **user** | Aktivnosti korisničkih procesa. |
| **uucp** | Podsustav za kopiranja među *Unix* sustavima. |
| **local0 do local7** | Lokalne poruke za testiranje. |

Tablica 29 - Tipovi događaja u syslogu

Razine mogu biti (poredano od najniže):

| Tip | Objašnjenje |
|---|---|
| **debug** | Najniža razina, sve aktivnosti se bilježe u svrhu pronalaženja pogrešaka u konfiguraciji. |
| **info** | Informativni zapisi o aktivnosti - ne indiciraju promjene stanja ili pogreške. |
| **notice** | Informativni zapisi o manje standardnim aktivnostima poput pokretanja/zaustavljanja servisa. |
| **warning (ili warn)** | Upozorenja o mogućim problemima u sustavu. |
| **err (ili error)** | Pogreške (nekritične pogreške koje neće rezultirati zaustavljanjem servisa ili zaustavljanjem rada sustava). |
| **crit** | Kritične pogreške koje uzrokuju zaustavljanje rada (servisa ili računala). Pogreške od kojih se servis ne može oporaviti. |
| **alert** | Ozbiljne pogreške koje mogu rezultirati zaustavljanjem sustava. |
| **emerg (ili panic)** | Kritične pogreške u radu jezgre ili hardverskih komponenti. |

Tablica 30 - Razine događaja u syslogu

Uređeni par tip i razina naziva se **prioritet.**

Odredište poruke može biti datoteka, terminal (/dev/tty<B>, gdje je B broj terminala), korisničko ime (ako je korisnik prijavljen na sustav poruke se šalju na njegov terminal) ili udaljeno računalo.

Danas je syslog gotovo izašao iz upotrebe. Njegova ograničenja su vremenom postala preveliki teret. Jedno od značajnih ograničenja sysloga je njegova nemogućnost DNS rezolucije. Razvijene su dvije poboljšane implementacije sustava za upravljanje sistemskim zapisima **syslog-ng** i **rsyslog**.

Uređeni parovi tipa i razine koriste se na sva tri sustava navedena u tečaju, **syslog**, **rsyslog** i **syslog-ng**, ali je sintaksa pravila u **syslog-ng** nešto drugačija od sintakse **syslog** i **rsyslog** sustava.

Syslog se pokreće servisom syslogd naredbom:

```
#/etc/init.d/syslogd start
```

Može se uočiti da se u primjeru ne koristi `systemd` naredba za pokretanje servisa. Razlog tome je da je **syslog** toliko zastario da je teško pronaći **systemd** sa instaliranim **syslogom**.

Syslog je u nekim distribucijama dio **sysklogd paketa** koji se sastoji od dva alata - **syslogd** i **klogd**. Primjer pravila u **/etc/syslog.conf**, središnjoj konfiguracijskoj datoteci sysloga:

```
auth,authpriv.*              /var/log/auth.log
*.*;auth,authpriv.none      -/var/log/syslog
#cron.*                      /var/log/cron.log
daemon.*                    -/var/log/daemon.log
kern.*                      -/var/log/kern.log
lpr.*                        /var/log/lpr.log
mail.*                      -/var/log/mail.log
user.*                      -/var/log/user.log
```

Dakle s „,“ odvajaju se nabrajanja više tipova ili razina, a s „;“ navođenje više prioriteta. Tako su ove dvije linije ekvivalenti.

```
auth,authpriv.*           /var/log/auth.log
```

```
auth.*;authpriv.*         /var/log/auth.log
```

Komentari se označavaju s „**#**“. Tako se linije koje započinju s „**#**“ignoriraju u procesiranju. Važno je napomenuti da se pri definiranju razine ne bilježe samo događaji te razine već svih razina viših od te razine i te razine. Kada se želi definirati da se u datoteku bilježe samo događaji točno određene razine treba dodati znak „**=**“. Tako bi u gornjem primjeru za događaje samo razine **crit** konfiguracija bila:

```
auth,authpriv.=crit            /var/log/auth.log
```

```
auth.=crit;authpriv.=crit       /var/log/auth.log
```

Moguća je i negacija sa specijalnim znakom „!“. Kada se koristi negacija, tada se pravilo odnosi na sve poruke koje nisu te razine ili tipa.

U gornjem primjeru vidljivo je da neki unosi datoteka u koje se zapisuje imaju, a neki nemaju početni minus. Zadano ponašanje je da se sinkronizacija datoteka događa svaki put kada se vrši upis u datoteku, odnosno svaki put kada se dogodi pravilima definiran događaj. Ovo može rezultirati velikim opterećenjem na sustav ako se izvrši veliki broj događaja. Oznaka „**-**“ ispred naziva datoteka isključuje takvu sinkronizaciju. Isključivanje stalne sinkronizacije dodatno štiti sustav i od udaljenih **DoS** (engl. *Denial of Service*) **napada**.

Syslog može slati sistemske zapise na udaljeno računalo kao sredstvo zaštite u slučaju sigurnosne kompromitacije lokalnog sustava. Napadač će moći promijeniti sadržaj lokalnih zapisa i poslati lažne zapise, ali neće moći promijeniti sadržaj postojećih zapisa na udaljenom računalu. Zapis za slanje na udaljeno računalo je oblika:

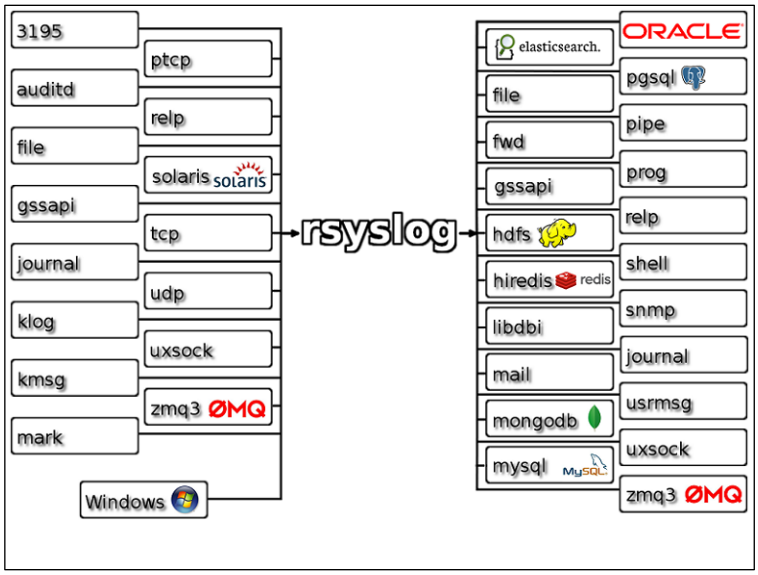
```
auth,authpriv.crit  @<udaljeno_računalo>:<port>
```

Standardni protokol  i port koji se koristi je **UDP 514**, ako se ne definira drugi port. Razlog korištenja UDP protokola je zaštita od eskalacije napada. Naime, središnji poslužitelji za pohranu sistemskih zapisa često prihvaćaju zapise sa više stotina poslužitelja (i više). Važno je naglasiti da iako **syslog** ne može provesti DNS rezoluciju, ipak se u konfiguraciji mogu koristiti imena poslužitelja ako za njih postoje zapisi u **/etc/hosts** datoteci.

### 4.1.2. Servis rsyslog

Akronim **RSYSLOG** znači „*rocket-fast **sy**stem for **log** processing*“. Iako izvorno razvijen kao nova inačica **sysloga**, ovaj je sustav narastao u obimu mogućnosti i namjena. Tako danas rsyslog može prihvaćati unos iz širokog spektra izvora, konvertirati zapise u brojne formate i proslijediti ih na široki spektar odredišta.

Ova elastičnost **rsysloga** ilustrirana je slikom 4.




Slika 4 - Mogućnosti rsysloga

**Rsyslog** u potpunosti podržava sve mogućnosti **sysloga**. Tako se konfiguracijska datoteka **sysloga** može jednostavno nakon instalacije **rsysloga** preimenovati:

```
#mv /etc/syslog.conf /etc/rsyslog.conf
```

Obratno ne vrijedi budući da **rsyslog** posjeduje niz funkcionalnosti koje **syslog** ne podržava. Dodatne funkcionalnosti koje **rsyslog** podržava su:

- može komunicirati na TCP/UDP i ograničiti s kim komunicira (port, izvorišni IP)
- dodatne funkcionalnosti u modulima koji mogu biti aktivirani po potrebi
- filtriranje i usmjeravanje na osnovu programa, izvora, sadržaja poruke, PID-a (engl. *process identification number*) i slično.

Središnja konfiguracijska datoteka **rsysloga** je **/etc/rsyslog.conf**. U njoj se definiraju globalne direktive, moduli i pravila koji se sastoje od filtera i akcija. Filteri u rsyslogu mogu biti (ranije spomenuti) prioritetni filteri, filteri na osnovu svojstva i filteri u obliku izraza. Linije konfiguracije se uz filter sastoje i od akcija koje se poduzimaju kada su uvjeti filtera zadovoljeni.

**Filtriranje na osnovi svojstva**

Linije kojima se definira filtriranje na osnovu svojstva počinje sa „**:**“ i ima oblik:

**:SVOJSTVO, [!]OPERATOR_USPOREDBE, "NIZ"**

Najčešće korištena dostupna svojstva su navedena u tablici 31:

| Svojstvo | Objašnjenje |
|---|---|
| **msg** | Dio zapisa koji je poruka o događaju. |
| **hostname** | Ime računala na kojem je događaj. |
| **fromhost** | Računalo sa kojeg je došla komunikacija (zadnji u lancu). |
| **programname** | Ime programa. |
| **syslogfacility** | Vrsta događaja. |
| **syslogseverity** | Razina događaja. |
| **timereported** | Vrijeme događaja. |

Tablica 31 - Svojstva za rsyslog konfiguraciju

Dostupne operacije usporedbe navedene su u donjoj tablici:

| Operator usporedbe | Objašnjenje |
|---|---|
| contains | Provjerava nalazi li se zadani niz negdje u tekstu. |
| contains_i | Isto kao prethodno, ali ignorira razlike - velika/mala slova. |

| | |
|---|---|
| isequal | Provjerava da je je niz identičan tekstu koji se obrađuje. |
| startswith | Provjerava započinje li dolazni tekst sa zadanim nizom. |
| startswith_i | Provjerava počinje li dolazni tekst zadanim nizom i pri provjeri ignorira razlike mala/velika slova. |
| regex | Zadaje se regularni izraz osnovnog oblika (**POSIX BRE** (Basic Regular Expression)) za usporedbu s tekstom. |
| ereregex | Zadaje se regularni izraz proširenog oblika (**POSIX ERE** (Extended Regular Expression)) za usporedbu s tekstom. |
| isempty | Provjerava je li tekst prazan. |

Tablica 32 - Operacije za rsyslog konfiguraciju

Prvi dio linije definira svojstvo - dio zapisa (u tablici nazivan tekstom) kojeg obrađuje **rsyslog**. Nakon toga dolazi jedan od operatora navedenih u tablici koji može biti negiran. Negacija je jedina trenutno podržana logička operacija. Nekoliko primjera:

```
:msg, contains, "error"
```

Primijenit će akciju na sve rsyslog poruke koje sadrže niz „error“.

```
:hostname, isequal, " poslužitelj_1"
```

Primijenit će akciju na sve rsyslog poruke poslane od poslužitelja poslužitelj_1

```
:msg, !regex, "fatal .* error"
```

Primijenit će akciju na sve rsyslog poruke koje ne sadrže niz „fatal“ nakon kojeg negdje u poruci postoji „error“.

**Filtriranje na osnovi izraza**

Sintaksa filtriranja na osnovi izraza je:

```
if IZRAZ then AKCIJA_1 else AKCIJA_2
```

IZRAZ je logički konstrukt koji se evaluira. Može se sastojati od više jednostavnih logičkih izraza povezanih sa **AND** ili **OR**.

AKCIJA_1 je akcija koja se izvršava ako je IZRAZ evaluiran kao **true**, a AKCIJA_2 je akcija koja se izvršava ako je IZRAZ evaluiran kao **false**. Else dio je opcionalan.

Pri korištenju filtriranja na osnovi izraza moguće je ugnijezditi izraze jedne u drugima koristeći vitičaste zagrade. Moguće je pomoću njih ugnijezditi i filtere drugih tipova. Primjer konfiguracije koja zapisuje za program **prog_1** zapise koji sadrže ili ne sadrže niz „test“u različite datoteke:

```
if $programname == 'prog_1' then {
  action(type="omfile" file="/var/log/prog_1.log")
  if $msg contains 'test' then
    action(type="omfile" file="/var/log/prog_1test.log")
   else
  action(type="omfile" file="/var/log/prog_1-work.log")
}
```

**Akcije**

Akcije definiraju što se čini sa porukama koje su filtrirane prema ranije definiranim filterima. Uglavnom se radi o definiranju gdje će se filtrirane poruke bilježiti. Neki od mogućih oblika zapisa su navedeni u donjoj tablici:

| Oblik zapisa | Objašnjenje |
|---|---|
| **<filter> <putanja>** | Definira se datoteka u koju se zapisuje poruka, jedan tipičan primjer iz prakse je pravilo koje sve cron poruke bilježi u **/var/log/cron.log**:<br><br>cron.* /var/log/cron.log |
| **<filter> –<putanja>** | Minus služi da se zaustavi sinkronizacija pri svakom zapisivanju. Na taj se način poboljšavaju performanse. |
| **<filter> ?Predlozak** | Gdje se zapisuje može se dinamički mijenjati u ovisnosti o drugim parametrima, najčešće o datumu. Prvo treba definirati predložak koji se zatim koristi. Na primjer: $template Predlozak,"/var/log/test/%datum%-test.log". |
| **<filter> @[(z<Broj>)]Računalo:[PORT]** | Slanje na udaljeno računalo, @označava da se radi o udaljenom računalu (koristi se UDP, za TCP treba postaviti @@), „Broj“ definira **zlib** biblioteci koju razinu kompresije da koristi, „Računalo“ i PORT parametri su intuitivno jasni. Primjeri:<br><br>*.* @192.168.0.1<br><br>*.* @@primjer.hr:7732 |
| **<filter> :omfile:$IME_Kanala** | Izlazni kanal definira se s ključnom riječi „omlfile“. Uglavnom se koristi za ostvarivanje rotacije datoteka, a mora biti ranije definiran u zapisu oblika $outchannel IME_Kanala, IME_Datoteke, MAXIMALNA_VELIČINA, AKCIJA.<br><br>Primjer pravila koje definira kanal direktivom *$outchannel* - $outchannel log_rotacija, /var/log/test_log.log, 104857600, /home/joe/log_rotacija_skripta – pravilo definira da kada |

| | |
|---|---|
| | se dosegne limit (100MB) izvrši skripta /home/joe/log_rotacija_skripta, kako bi se moglo kreirati filtriranje - *.* :omfile:$log_rotation – koje jednostavno zapisuje sve događaje koje **rsyslog** prikupi u kanal. |
| **<filter> <korisnik>** | Šalje korisniku na terminal. Moguće je poslati svim korisnicima korištenjem „*“ ili na više korisnika odvajajući listu zarezom. Na primjer: *.crit l201, linux1, root. |
| **&** | & je znak za povezivanje više različitih akcija. Mora biti prvi znak u liniji te nakon njega slijedi akcija. Primjer:<br><br>kern.=crit user1<br><br>& ^test-program;temp<br><br>& @192.168.0.1 |
| **<filter> ~** | Koristi se za odbacivanje poruka. |

Tablica 33 - Oblici zapisa za definiranje akcija u rsyslogu

Dodatno se koriste akcije pokretanja programa s parametrima iz poruke, zapisivanje u bazu podataka.

### 4.1.3. Servis syslog-ng

**Syslog-ng** je proširenje izvornog **syslogd** modela sa kontekstnim filterima, dodatnim mogućnostima filtriranja i dodatnim konfiguracijskim opcijama. Postoje dvije grane **syslog-ng** od kojih je jedna otvorenog kôda, a druga imena *Premium Edition (PE)* nije i sadržava dodatne module i dodatke (engl. *plugins*).

Važno je napomenuti da je što se tiče zadanog sustava na većini Uniksolikih sustava zadani sustav **rsyslog**, a **syslog-ng** je moguće instalirati iz paketa.

Središnji konfiguracijski direktorij je **/etc/syslog-ng/**, a središnja konfiguracijska datoteka je **/etc/syslog-ng/syslog-ng.conf.** Struktura konfiguracijske datoteke je:

- Globalne varijable
- Izvori
- Odredišta
- Filteri
- Usmjeravanja

U prvom se dijelu definiraju **globalne varijable** koje se primjenjuju na izvršavanje nad svim filterima. Konfiguracijom je isključena upotreba DNS-a (izbjegava se zato što troši vrijeme), kao i uporaba fqdn, vlasništvo nad datotekama u koje se zapisuje i tako dalje. Primjer:

```
options { chain_hostnames(off); flush_lines(0); use_dns(no);
use_fqdn(no);
          owner("root"); group("adm"); perm(0640); stats_freq(0);
          bad_hostname("^gconfd$");
};
```

Zatim se definira koji su **izvori** prihvatljivi –prihvaćaju li se samo unutarnji signali ili se prihvaćaju i poruke putem mreže. Primjeri:

```
source s_src {
       system();
       internal();
};
source s_net { tcp(ip(127.0.0.1) port(1000)); };
```

Nakon toga se definiraju **odredišta** – to mogu kao i kod **rsyslog** biti brojni tipovi odredišta: datoteke, udaljenja računala, konzole, korisnici…. Primjeri:

```
destination d_auth { file("/var/log/auth.log"); };
destination d_cron { file("/var/log/cron.log"); };
destination d_console { usertty("root"); };
destination d_console_all { file(`tty10`); };
destination d_xconsole { pipe("/dev/xconsole"); };
destination d_net { tcp("127.0.0.1" port(1000) log_fifo_size(1000)); };
```

Nakon toga se postavljaju **filteri**, to mogu biti kompleksni logički izrazi koji se definiraju prema potrebama sustava. Moguće ih je ugnijezditi i mogu se koristiti više puta, budući da se ovdje samo definiraju, a koriste se kasnije. Primjeri:

```
filter f_dbg { level(debug); };
filter f_crit { level(crit .. emerg); };
filter f_debug { level(debug) and not facility(auth, authpriv, news,
mail); };
filter f_local { facility(local0, local1, local3, local4, local5,
                        local6, local7) and not filter(f_debug); };
filter f_mail { facility(mail) and not filter(f_debug); };
filter f_news { facility(news) and not filter(f_debug); };
```

Konačno na kraju dolaze **usmjeravanja**, a to su uređene trojke **izvor-filter-odredište**.
Oni  definiraju da se poruke s izvora koje ispunjavaju filter usmjeravaju na zadano odredište. Tako su pravila usmjeravanja primjerice:

```
log { source(s_src); filter(f_console); destination(d_console_all);
                                    destination(d_xconsole); };
log { source(s_src); filter(f_crit); destination(d_console); };
```

```
log { source(s_src); destination(d_net); };
```

Kao što je prikazano u primjeru, moguće je izostaviti filter, a tada se sve poruke s izvora prenose na odredište. Također je moguće imenovati više odredišta ili filtera da se ne bi moralo pisati nizove sličnih linija sa visokom redundancijom.

**Sintaksa syslog-ng konfiguracije**

Bit će prikazan veći primjer konfiguracije za usporedbu iste konfiguracije na rsyslog i syslog-ng sustavu. Prvo pogledajmo i protumačimo konfiguraciju na rsyslog sustavu:

```
if ( \
/* kernel do warning osim ako je izvor vatrozid */ \
($syslogfacility-text == 'kern') and \
($syslogseverity <= 4 /* warning */ ) and not \
($msg contains 'IN=' and $msg contains 'OUT=') \
) or ( \

/* do razine error osim tipa authpriv */ \
($syslogseverity <= 3 /* errors */ ) and not \
($syslogfacility-text == 'authpriv') \
) \
then /dev/tty10
& |/dev/xconsole
```

Dakle, u ovom primjeru se na konzolu 10 i u (virtualnu) **/dev/xconsole** datoteku bilježe svi događaji tipa **kern** i razine niže ili jednake **warning** koji nisu od vatrozida (engl. *firewall*) i svi događaje koji nisu tipa **authpriv** a razine su niže ili jednake **error**. Pogledajmo istu konfiguraciju za syslog-ng:

```
filter f_iptables { facility(kern) and message("IN=") and
message("OUT=");
};

filter f_console { level(warn) and facility(kern) and not
filter(f_iptables) or level(err) and not
facility(authpriv); };

log { source(src); filter(f_console); destination(console); };
log { source(src); filter(f_console); destination(xconsole); };
```

Može se uočiti da je struktura u ovoj konfiguraciji čišća i bolje strukturirana. Takva konfiguracija je preglednija i lakša za održavanje. Ovakva struktura je dostupna zbog toga što je filtere u syslog-ng moguće imenovati i koristiti više puta. Na taj je način omogućeno da se na jednom mjestu nalazi neko svojstvo, te da mijenjanje istog svojstva zahtjeva mijenjanje samo jednog parametra u konfiguraciji.

Uz tri sustava koji se koriste za istu funkciju nameće se pitanje koji je sustav najbolji. Budući da je **syslog** zastario i da su druga dva sustava razvijena s ciljem da omoguće funkcionalnosti koje nedostaju u njemu,pitanje je doista **rsyslog** ili **syslog-ng** i zašto.

Pogledajmo svojstva i mogućnosti oba sustava pa zaključimo. **Syslog-ng** je stariji (1998) od **rsyslog** sustava(2004) te je **rsyslog** u početku ispravio neke nedostatke **syslog-ng** konfiguracije. Kada se usporede funkcionalnosti oba sustava dolazi se do poraznog rezultata za sustav **syslog-ng**. Ispada da **syslog-ng** podržava 4 funkcionalnosti koje **rsyslog** ne podržava, a **rsyslog** dvadesetak funkcionalnosti koje ne podržava **syslog-ng**. Što je najvažnije **syslog-ng** nije kompatibilan sa **syslog** konfiguracijama, ali upravo u tome leži njegova snaga.

Naime najveća prednost **syslog-ng** je njegova konfiguracija koja je čišća, urednija i preglednija od konfiguracije **rsyslog**. Da bi se to postiglo zanemarena je mogućnost ostvarivanja kompatibilnosti sa ranijim sustavima i građena je cijela nova sintaksa.

## 4.2. Debian paketi

### 4.2.1. Izrada Debian paketa

Debian paketi su skupovi datoteka koji čine cjelinu koja omogućava određenu funkcionalnost ili izvršavanje niza srodnih operacija. Razlikujemo binarne pakete i pakete izvornog kôda.

Binarni se paketi sastoje od izvršnih datoteka, konfiguracija za rad tih izvršnih datoteka i stranica priručnika, detalja o autorskim pravima i ostale dokumentacije. Ovi paketi imaju sufiks .deb i njima se upravlja sa `dpkg` naredbom.

Paketi izvornog kôda sastoje se od datoteke sufiksa **.dsc** u kojoj se nalazi opis paketa zajedno sa popisom svih sadržanih datoteka, izvornog tarballa i sve diff datoteke koje prikazuju razliku od izvornog tarballa. Sa paketima izvornog kôda upravlja se naredbom `dpkg-source`.

U ovom poglavlju bit će prikazano kako jednostavno izmijeniti postojeći paket te će biti predstavljen kratak pregled izrade paketa.

**Izrada novog paketa**

Novi paket se izrađuje kada se neka nova funkcionalnost ili neki novi softver želi instalirati na niz računala. Paket uz izvorni softver koji omogućava danu funkcionalnost uključuje i upute, dokumentaciju i ovisnosti odnosno konflikte sa ostalim softverom.

Izrada paketa za određenu distribuciju rijetko je zadatak koji provodi ista osoba (ili osobe) koja je/su autor(i) programskog kôda softvera. Standardno se paketi izrađuju na osnovi tuđeg kôda koji je obično dostupan u obliku komprimirane arhive algoritmom gzip (kolokvijalno na engleskom nazvana *upstream tarrball*) ili nekim drugim algoritmom standardnim za izradu arhiva UNIX-a. U nastavku teksta taj paket izvornog kôda imat će naziv **tarball**. Izrada binarnog paketa odvija se u dva koraka – prvo se na osnovi tarballa izrađuje **paket za prevođenje** (engl. *source package*), a zatim se prevodi u **izvršni oblik** (.deb sufiksa).

Postoje izuzetci gdje nije potrebno programsko prevođenje jer ne postoje izvršne datoteke, a također je moguće i preskočiti korak izrade paketa za prevođenje.

Minimalni **paket za prevođenje** sastoji se od tri dijela:

- **tarballa** koji je preimenovan u ime koje odgovara konvenciji imenovanja datoteka u paketima
- direktorij **debian/** u kojem se nalaze sve izmjene tarballa i dodatne datoteke za izradu izvršnog paketa.
- datoteke opisa paketa (sufiksa .dsc) u kojoj su navedene minimalno druge dvije datoteke.

Izrada paketa se odvija u 6 koraka:

1. Preimenovanje tarballa
2. Raspakiravanje tarballa
3. Dodavanje Debian konfiguracijskih datoteka za izradu paketa
4. Izrada paketa
5. Instalacija paketa
6. Testiranje paketa

Za Debian pakete definiran je vrlo strog predložak za **imenovanje** tarballa. Ime mora biti ovako oblikovano: <imepaketa>_<verzija tarballa>.orig.tar.gz. Tako je u prethodnom primjeru ime datoteke bilo **logrotate_3.8.7.orig.tar.gz**. Dakle, naredba u ovom koraku je:

```
$ cp logrotate-3.8.7.tar.gz logrotate_3.8.7.orig.tar.gz
```

Nakon toga, potrebno je **raspakirati** arhivu izvornog tarballa. Naredba koja će se koristiti ovisi o formatu datoteke koja je pribavljena. U primjeru je prikazana naredba za .tar.gz datoteku, ali moguće je korištenje i arhiva drugog tipa.

```
$ tar -xf logrotate_3.8.7.orig.tar.gz
```

Nakon toga potrebno je dodati **konfiguracijske datoteke**, te se datoteke dodaju u direktorij debian/, koji je smješten u direktoriju izvornog kôda.

```
$ cd logrotate-3.8.7/
$ mkdir debian && cd debian
```

Prva je debian/changelog datoteka, u koju se zapisuju izmjene paketa. Najjednostavnije ju je kreirati naredbom `dch`:

```
$ dch --create -v 3.7.8-1 --package logrotate
Select an editor.  To change later, run 'select-editor'.
   1. /bin/nano        <---- easiest
   2. /usr/bin/vim.basic
   3. /usr/bin/vim.tiny
```

Parametri naredbe nisu značajni. Iako će se te vrijednosti zapisati u **changelog** datoteku, odmah će se i pokrenuti odabrani program za uređivanje teksta pa će biti moguće promijeniti parametre ako pogriješimo u pozivu. Sadržaj datoteke kreirane ovom naredbom je:

```
logrotate (3.7.8-1) UNRELEASED; urgency=medium

 * Initial release. (Closes: #XXXXXX)

 -- Linux tux <linux1@l201.srce.local>  Thu, 12 Jan 2017 13:07:03 +0100
```

Važno je napomenuti pravila koje se moraju slijediti pri izradi ove datoteke:

- Ime paketa (logrotate) mora biti isto kao i ime izvornog tarballa.
- Verzija (3.7.8-1) znači da je verzija programa 3.7.8, a Debian paketa 1.
- UNRELEASED dio definira da paket nije još spreman za učitavanje u repozitorij, zbog toga je dobra ideja ostaviti taj zapis dok se paket ne završi i testira. Ako se to ne napravi, alati za automatski upload mogu slučajno prenijeti nevaljanu verziju na repozitorij.
- **Urgency** može imati vrijednost **low**, **medium** i **high** gdje se za standardne izmjene paketa koristi **low**, **medium** za hitnije zakrpe ili ispravke, a high za nadogradnje koje su hitne odnosno kritične zakrpe.
- (Closes: #XXXXXX) dio se koristi za praćenje stanja detektiranih pogrešaka (engl. bug).
- Zadnja linija definira autora paketa, dhc prikuplja i postavlja u datoteku podatke o trenutno aktivnom korisniku, ako isti nisu zadani u naredbenoj liniji.

Druga datoteka je debian/control koja opisuje tarball i binarni paket. Ovdje se nalaze podaci o autoru izvornog kôda, ovisnosti o drugim paketima i opis samog programa. Datoteka se sastoji od dva dijela, u prvom se nalaze definicije koje opisuju zahtjeve i ovisnosti pri programskom prevođenju, a drugi dio opisuje zahtjeve i ovisnosti pri instalaciji binarnog paketa. Za opisani primjer sadržaj bi izgledao ovako:

```
Source: logrotate
Section: admin
Priority: important
Maintainer: Paul Martin <pm@debian.org>
Build-Depends: libpopt-dev, debhelper (>= 9),
 libselinux1-dev [linux-any], libacl1-dev [linux-any]
Vcs-Svn: http://svn.fedorahosted.org/svn/logrotate/
Homepage: https://fedorahosted.org/logrotate/
Standards-Version: 3.9.5

Package: logrotate
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}, cron | anacron | cron-
daemon, base-passwd (>= 2.0.3.4)
Breaks: postgresql-common (<= 126)
Recommends: mailx
Description: Log rotation utility
 The logrotate utility is designed to simplify the administration of
```

```
 log files on a system which generates a lot of log files.  Logrotate
 allows for the automatic rotation compression, removal and mailing of
log files.  Logrotate can be set to handle a log file daily, weekly,
monthly or when the log file gets to a certain size.  Normally,
logrotate runs as a daily cron job.
```

Na kraju je potrebno u debian/source/format datoteci definirati inačicu za format paketa izvornog kôda. Danas je to 3.0 (quilt), dakle:

```
$ mkdir debian/source && echo "3.0 (quilt)"> debian/source/format
```

Nakon ovoga je paket spreman za prevođenje. Naredba je:

```
$ debuild -us -uc
```

Potrebno je napomenuti da ova naredba generira veliku količinu podataka pri izvršavanju i ako je sve dobro postavljeno stvara brojne nove datoteke u direktoriju. Stvoriti će se datoteke

- logrotate_3.8.7-1.debian.tar.xz
- logrotate_3.8.7-1.dsc
- logrotate_3.8.7-1_i386.deb
- logrotate_3.8.7-1_i386.changes
- logrotate_3.8.7-1_i386.build

Paket je stvoren i spreman za instalaciju. Jednostavno kao i svaki drugi paket instalira se naredbom `dpkg -i`:

```
# dpkg -i logrotate_3.8.7-1_i386.deb
```

Nakon toga se naredbom `dpkg -i` može provjeriti verzija izrađenog paketa:

```
# dpkg -l |grep logrotate
ii  logrotate   3.8.7-1  i386 Log rotation utility
```

Važno je napomenuti da se izrada paketa i instalacija te kasnije testiranje provodi na neprodukcijskom stroju zbog toga što može doći do neželjenih posljedica, odnosno ako se ove naredbe izvršavaju s root ovlastima, dovesti sustav u neispravno stanje.

**Korištenje dh_make alata**

U prethodnom primjeru je prikazano kreiranje paketa iz izvornog kôda ručnim kreiranjem i uređivanjem sadržaja datoteka u **debian** direktoriju. **dh_make** alat koristi se za pretvaranje izvornog kôda u oblik koji zadovoljava Debianovu politiku za kreiranje paketa. Za primjer ćemo kreirati paket koji sadrži jednu jednostavnu skriptu ljuske.

Sadržaj datoteke je sljedeći:

```
# cat pozdrav.sh
#!/bin/bash
echo "Dobar dan narode!!!"
```

**dh_make** alat kreira potrebne datoteke za izradu Debian paketa koristeći informacije sadržane u samom izvornom kôdu u kombinaciji sa vrijednostima nekih varijabli. Prije korištenja alata postaviti ćemo vrijednosti varijabli DEBMAIL I DEBFULLNAME pomoću kojih će dh_make postaviti zapise o autoru paketa i njegovoj adresi elektronske pošte.

```
# vim ~/.bashrc
```

U datoteku dodajemo:

```
DEBEMAIL="l201@srce.hr"
DEBFULLNAME="l201 PackageMaster"
export DEBEMAIL DEBFULLNAME
)
#  .    ~/.bashrc
```

Zadnja naveden naredba će ponovno izvršiti datoteku i tako aktivirati sve izmjene unesene u **.bashrc** datoteku od pokretanja ljuske.

Sada izradimo direktorij naziva **pozdravljanje-0.1.** Naziv ovog direktorija je važan jer će **dh_make** alat imenovati paket onako kako se zove taj direktorij. U kreirani direktorij kopiramo datoteku ljuske i pokrenemo naredbu `dh_make`:

```
# mkdir pozdravljanje-0.1 && cd pozdravljanje-0.1
# mv ../pozdrav.sh .
# dh_make --indep --createorig
ail-Address       : l201@srce.hr
License             : blank
Package Name        : pozdravljanje
Maintainer Name     : l201 PackageMaster
Version             : 0.1
Package Type        : indep
Date                : Wed, 24 Jan 2018 10:30:05 +0100
Are the details correct? [Y/n/q]
Currently there is not top level Makefile. This mayrequire additional
tuning
Done. Please edit the files in the debian/ subdirectory now.
```

Pri gornjem pozivu su korištene dvije mogućnosti. Prvo pomoću **--indep** definiramo da se kreira binarni paket koji je neovisan (engl. *independent*) o arhitekturi. Zatim sa **--createorig** definiramo de se na osnovi trenutnog direktorija kreira orginalni paket izvornog kôda koji je potreban za kreiranje paketa.

Rezultat izvršavanja gornjih naredbi je kreiranje direktorija **./debian** i arhive **../pozdravljanje_0.1.orig.tar.xz**. Pogledamo li sadržaj direktorija **./debian** vidjeti ćemo niz datoteka sa sufiksom .ex ili .EX. Te datoteke sadrže primjere kôda i ne koriste se pri izradi paketa. Kada želimo da se određena funkcionalnost doda u paket tada jednostavno maknemo sufiks i uredimo datoteku. Primjerice kada bi željeli da se doda neki niz akcija pri ukljanjanu paketa, a nakon samog uklanjanja paketa maknuli bi sufiks datoteci ./debian/postrm.ex i zatim unijeli željene izmjene u datoteku.

```
# mv ./debian/postrm.ex ./debian/postrm
# vim ./debian/postrm
```

Potrebno je još dodati datoteku **.debian/install** u kojoj se definira gdje se pojedinačne datoteke paketa instaliraju. **dh_make** alat ove podatke uobičajeno prikuplja iz definicija za make alat. Budući da je ovo minimalni paket bez tih definicija biti će još nužno postaviti te vrijednosti.

```
# echo "pozdrav.sh usr/bin" > debian/install
```

Budući da je paket vrlo jednostavan i da smo pripremili sve za uspješnu pripremu pomoću alata **dh_make** sada je jednostavno izraditi i instalirati paket.

Prvo naredbom **debuild** izradimo paket. Mogućnosti **–us** i **–uc** definiraju da se neće izvršiti potpisivanje paketa.

```
#  debuild -us -uc
 dpkg-buildpackage -rfakeroot -us -uc
…
 dpkg-source -b pozdravljanje-0.1
dpkg-source: info: using source format '3.0 (quilt)'
dpkg-source: info: building pozdravljanje using existing
./pozdravljanje_0.1.orig.tar.xz
…
dpkg-buildpackage: info: full upload (original source is included)
Now running lintian...
warning: the authors of lintian do not recommend running it with root
privileges!
…
W: pozdravljanje: binary-without-manpage usr/bin/pozdrav.sh
Finished running lintian.
```

Zatim naredbom `dpkg -i` instaliramo paket izravno koristeći datoteku.

```
## dpkg -i ../pozdravljanje_0.1-1_all.deb
Selecting previously unselected package pozdravljanje.
(Reading database ... 177484 files and directories currently installed.)
Preparing to unpack ../pozdravljanje_0.1-1_all.deb ...
Unpacking pozdravljanje (0.1-1) ...
Setting up pozdravljanje (0.1-1) ...
```

Nakon instalacije je datoteka pozdrav.sh dostupna u direktoriju /usr/bin, a budući da je ta putanja standardni dio

```
    # pozdrav.sh
    # pozdrav.sh
```

### Izmjena postojećeg paketa

Da bi se postojeći paket mogao pribaviti, izmijeniti te tako izraditi novi paket, potrebne su neke predradnje. Prvo je potrebno instalirati potrebne pakete **build-essential**, **fakeroot I devscripts**.

```
#apt-get install build-essential fakeroot devscripts -y
```

Zatim je s naredbom `apt-get source` potrebno pribaviti izvorni kôd nekog paketa sa svim konfiguracijama za izradu paketa. Naredba `apt-get source` pribavlja sve potrebne datoteke u trenutni radni direktorij. Da bi naredba radila, mora postojati definicija **deb-src** u izvorima paketa za apt u **/etc/apt/sources.list** datoteci. Tako je moguće definirati iz kojeg se repozitorija pribavljaju paketi.

Potrebno je još instalirati sve pakete koji su potrebni za programsko prevođenje ciljanog paketa. Naredba za instalaciju tih ovisnosti je **apt-get build-dep**. Primjerice, pri izradi novog programskog paketa za logrotate, naredbe bi bile sljedeće:

```
# mkdir -p ~/tmp/izrada_paketa/
# cd ~/tmp/izrada_paketa/
# apt-get source logrotate
Reading package lists... Done
Building dependency tree
Reading state information... Done
NOTICE: 'logrotate' packaging is maintained in the 'Svn' version control system at:
http://svn.fedorahosted.org/svn/logrotate/
Need to get 80.8 kB of source archives.
Get:1 http://ftp.hr.debian.org/debian/ jessie/main logrotate 3.8.7-1 (dsc) [1,804 B]
Get:2 http://ftp.hr.debian.org/debian/ jessie/main logrotate 3.8.7-1 (tar) [58.9 kB]
Get:3 http://ftp.hr.debian.org/debian/ jessie/main logrotate 3.8.7-1 (diff) [20.1 kB]
Fetched 80.8 kB in 0s (480 kB/s)
dpkg-source: info: extracting logrotate in logrotate-3.8.7
dpkg-source: info: unpacking logrotate_3.8.7.orig.tar.gz
dpkg-source: info: unpacking logrotate_3.8.7-1.debian.tar.xz
dpkg-source: info: applying deb-config-h.patch
dpkg-source: info: applying datehack.patch
dpkg-source: info: applying manpage.patch
dpkg-source: info: applying cpp-crossbuild.patch
dpkg-source: info: applying chown-484762.patch
dpkg-source: info: applying mktime-718332.patch
dpkg-source: info: applying man-su-explanation-729315.patch
/tmp/izrada_paketa # apt-get build-dep logrotate
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  libacl1-dev libattr1-dev libpopt-dev libselinux1-dev libsepol1-dev
```

```
0 upgraded, 5 newly installed, 0 to remove and 0 not upgraded.
Need to get 516 kB of archives.
After this operation, 1,405 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Paket je kreiran i spreman za instalaciju.

**Skripte održavatelja**

**postinst, preinst, postrm** i **prerm** skripte nazivaju se skriptama održavatelja (engl. *maintainer scripts*). Ove skripte pokreče dpkg alat kada se paket instalira, deinstalira ili nadograđuje.

Izrada ovih skripti je najzahtjevniji dio izrade paketa. Uloga skripta održavatelja je da osiguraju uspješno i glatko manipuliranje paketom na svim sustavima. Problem nastaje jer se stanje na poslužiteljima može razlikovati značajno od stanja na sustavu za razvoj paketa, a u skriptama trebaju biti pokrivene sve moguće situacije. Primjerice ako nije instalirana prethodna verzija paketa već neka značajno starija, situacija gdje je softver potreban za rad paketa instaliran iz različitih repozitorija i slično.

Početnici bi iz toga razloga trebali izbjegavati mijenjanje postojećih skripti održavatelja.

### 4.2.2. Izrada paketa s modulima jezgre

Neki moduli jezgre nisu sastavni dio tarballa jezgre, ali su dostupni kao tarball vanjskih autora (engl. *third party,* prvi i drugi bi bili autor i korisnik jezgre). Za popularne module koji nisu dio standardne distribucije postoje binarni paketi za instalaciju modula. Primjerice, za korisnike datotečnog sustava **squashfs** postoji squashfs-modules-3.2.0-2-686-pae binarni paket. Taj paket je sve što je potrebno instalirati da bi omogućili podršku za korištenje datotečnog sustava.

Problem nastaje ako ne postoji izvršni paket za modul koji se želi koristiti. U tom slučaju možda postoji paket izvornog kôda (*-source*). U tim paketima nalazi se prilagođen oblik izvornog kôda za Debian. Takav je oblik namijenjen za korištenje s **module-assistant** (kratki oblik m-a) skriptom iz **module-assistant** paketa. Koraci za izgradnju modula su:

1. Identificirati verziju jezgre (na računalu na kojem će se instalirati paket modula):

```
# uname -a
Linux l201 3.13-1-486  #1 Debian 3.16.36-1+deb8u2 (2016-10-19) i686 GNU/Linux
```

2. Nakon toga, na računalu na kojem se izrađuje paket potrebno je instalirati datoteke zaglavlja jezgre (engl. *kernel header*). U gornjem primjeru je verzija jezgre je 3.16.0-4. Također treba instalirati i **module-assistant** paket, ako već nije instaliran.

```
# apt-get install linux-headers-3.13-1-486  module-assistant -y
```

3. Nakon toga se instalira paket s prilagođenim izvornim kôdom:

```
# apt-get install squashfs-source –y
```

Na kraju je još potrebno s **module-assistant naredbom** kreirati Debian paket.

```
# m-a build squashfs
```

Paket će biti smješten u direktorij **/usr/src**, i otuda ga je moguće kopirati na računalo na kojem će biti instaliran.

### 4.2.3. Izrada paketa s modulima za Perl

**Perl** je programski jezik široke namjene i velike moći, a dobar dio njegove moći proizlazi iz modula izrađenih od članova zajednice i dostupnih u CPAN repozitoriju. Debian pruža niz alata koji olakšavaju izradu unificiranih paketa Perl modula.

Paket za olakšavanje izrade Debian paketa za Perl modul je **dh-make-perl** paket koji se instalira standardno kao bilo koji drugi paket:

```
# apt-get install dh-make-perl
```

Za primjer će se koristiti Modul HTML::Template::JIT. Prvi korak je pribavljanje tarballa koji sadrži paket i otpakiravanje arhive:

```
# wget http://search.cpan.org/CPAN/authors/id/S/SA/SAMTREGAR/HTML-
Template-JIT-0.04.tar.gz
# tar -pzxvf HTML-Template-JIT-0.04.tar.gz
```

Nakon toga je potrebno pomoću `dh-make-perl` naredbe kreirati konfiguracijske datoteke za izradu paketa:

```
# dh-make-perl HTML-Template-JIT-0.04/
```

Ova će naredba kreirati **debian/** direktorij i u njemu sve potrebne datoteke za izradu paketa. Naravno, ove su zadane postavke dovoljne samo za pakete koje korisnik namjerava sam koristiti. Kada je cilj napraviti paket modula koji se želi podijeliti putem službenih Debian repozitorija potrebno je ručno prilagoditi cijeli niz postavki.

Kada `dh-make-perl` naredba završi s radom, potrebno je još samo izraditi paket naredbom `debuild`:

```
# cd HTML-Template-JIT-0.04
# debuild
```

Ako je izvršavanje bilo uspješno, u direktoriju gdje se nalazi direktorij izvornoga kôda nalazit će se binarni paket imena **libhtml-template-jit-perl_0.04-1_all.deb**. Taj se paket može instalirati kao bilo koji drugi paket `dpkg` naredbom.

Ovakva izrada paketa je izuzetno brza, ali ima i veliki nedostatak. Naime, u paketu se ne nalaze ovisnosti modula. Tako modul iz primjera **HTML::Template::JIT** za rad treba **HTML::Template modul**. Paket koji je napravljen ne ovisi o **paketu libhtml-template** i zbog toga treba napraviti

jednu od dvije stvari: Ručno dodati ovisnosti u konfiguracijske datoteke u ./debian/ direktoriju ili neprestano paziti na ove ovisnosti i biti ih svjestan pri svakom brisanju modula s računala.

### 4.2.4. Izrada Java paketa

Postoji paket za Javu naziva **OpenJDK**, ali za **Oracle JDK** u repozitorijima distribucija ne postoji paket. **OpenJDK** je referentni model i otvorenog kôda, dok je **Oracle JDK** jedna implementacija i nije otvorenog kôda. OpenJDK je objavljen pod **GPL v2 licencom**, a Oracle JDK je licenciran pod Oraclovom ***Oracle Binary Code License Agreement***. Oracle JDK također je besplatan, ali nije dopuštena redistribucija. Upravo zbog toga ne može postojati paket na repozitorijima. Budući da nitko ne smije distribuirati Oracle JDK, jasno je da svaki korisnik mora sam napraviti paket.

Paket **java-package** pruža mogućnost izrade Debian paketa izravno iz Java binarnog paketa. Središnji alat za kreiranje java paketa je **make-jpkg**, a tipični proces bit će prikazan na primjeru najnovije inačice Oracle JDK.

1. Prvi korak je pribavljanje **Oracle JDK tarballa**. Za pristup datoteci potrebno je prihvatiti ***Oracle Binary Code License Agreement,*** a kako bi se to automatski dogodilo potrebno je koristiti dodatne **wget parametre**:

```
# wget --no-check-certificate --no-cookies --header "Cookie:
oraclelicense=accept-securebackup-cookie" http://download.oracle.com/otn-
pub/java/jdk/8u121-b15/jdk-8u121-linux-x64.tar.gz
```

2. Nakon toga, potrebno je instalirati **java-package** paket. Prvo treba dodati konfiguraciju u **/etc/apt/sources.list** koja uključuje **contrib komponentu**.

```
# Debian  9 "Stretch"
deb http://httpredir.debian.org/debian/ stretch main contrib
```

3. Nakon toga, potrebno je instalirati paket:

```
# apt-get update && apt-get install java-package && exit
```

4. Zatim naredbom `make-jpkg` izraditi Java paket:

```
$ make-jpkg jdk-8u121-linux-i586.tar.gz
```

Nakon niza zapisa na ekranu, ako je sve bilo u redu, generira se **oracle-java8-jdk_8u121_i386.deb** datoteka.

```
$ ls
jdk-8u121-linux-i586.tar.gz  oracle-java8-jdk_8u121_i386.deb
```

5. Budući da Java nema ovisnosti ovaj se paket može instalirati s:

```
linux1@l201:~/java_debmake# dpkg -i oracle-java8-jdk_8u121_i386.deb
```

Ovim je korakom instalacija gotova. U nastavku će biti proučena konfiguracija Jave.

**Java konfiguracija**

**Debian *Alternatives* sustav** daje mogućnost da više programa koji obavljaju istu ulogu, a svi su instalirani na računalu, budu navedeni kao alternativne implementacije i da jedna od njih bude postavljena kao zadana. Zadane postavke od **DebianAlternatives** automatski postavljaju najnoviju verziju Jave kao aktivnu. Svojstvo alternatives je da pokušava ne mijenjati korisničke postavke. Ako su neke izmjene provedene, a želi se vratiti na zadane vrijednosti naredba je **update-alternatives –auto**, u ovom primjeru **update-alternatives –auto java**.

Naredba za postavljanje neke druge verzije umjesto zadane je `update-alternatives -config`. Iako postoje brojne naredbe za pregled stanja, ova naredba pri izboru verzije daje pregled dostupnih inačica jave. U primjeru će se prvo provjeriti verzija Jave (1), a zatim promijeniti verzija (2) te ponovno provjeriti verziju Jave (3).

```
(1)# java -version
java version "1.7.0_111"
OpenJDK Runtime Environment (IcedTea 2.6.7) (7u111-2.6.7-2~deb8u1)
OpenJDK Client VM (build 24.111-b01, mixed mode, sharing)
(2)# update-alternatives --config java
There are 2 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                            Priority   Status
------------------------------------------------------------
* 0            /usr/lib/jvm/java-7-openjdk-i386/jre/bin/java   1071      auto mode
  1            /usr/lib/jvm/java-7-openjdk-i386/jre/bin/java   1071      manual mode
  2            /usr/lib/jvm/jdk-8-oracle-i586/jre/bin/java     318       manual mode

Press enter to keep the current choice[*], or type selection number: 2
update-alternatives: using /usr/lib/jvm/jdk-8-oracle-i586/jre/bin/java to provide
/usr/bin/java (java) in manual mode
(3)# java -version
java version "1.8.0_121"
Java(TM) SE Runtime Environment (build 1.8.0_121-b13)
Java HotSpot(TM) Client VM (build 25.121-b13, mixed mode)
```

Kao što je vidljivo u primjeru, simbolička se poveznica promijeni i vrlo se jednostavno na cijelom sustavu može promijeniti koja se Java koristi.

## 4.3. Perl skripte

### 4.3.1. Usporedba programskih jezika i skripti ljuske

Perl je programski jezik koji je svoju popularnost stekao u kasnim 1990-im kao skriptni CGI (engl. *Common Gateway Interface*) jezik sa tada nenadmašnim mogućnostima manipulacije nad nizovima i implementacije regularnih izraza. Uz ta svojstva ovaj programski jezik posuđuje brojne značajke iz drugih programskih jezika, a najviše iz C-a, sh skriptnog jezika, sed-a i AWK-a.

Svaki napredniji *Linux* korisnik susreo se sa **skriptama ljuske**, a vjerojatno je i autor barem jedne. U skriptama ljuske postoje brojni standardni elementi programskih jezika, ali skripte ljuske se ne koriste nikada za ozbiljne razvojne projekte. S druge strane, programski jezici mogu sve što mogu i skripte ljuske i još mnogo toga – pa je moguće pitanje: Koja je razlika skripti ljuske i programskih jezika i kada koristiti koje?

Skripte ljuske su puno jednostavnije za korištenje – put od ideje do realizacije u nekom jednostavnom scenariju je značajno kraći. Nije potrebno učenje jer se u skriptama ljuske koriste pozivi dostupni u ljusci. Također nije potrebno prevođenje, budući da su skripte ljuske već pisane u jeziku ljuske – sama ljuska odrađuje prevođenje u strojne instrukcije. Iz ovog svojstva dolazi i nedostatak skripti ljuske, a to je neportabilnost. Skripte ljuske mogu funkcionirati samo na računalima koja imaju instaliranu zadanu ljusku, a to ih ograničava na **Uniksolike sustave**.

S druge strane, programski jezici često trebaju značajno više kôda za jednostavne funkcije, naravno kada se ne radi o zadatcima za koje je taj programski jezik specijaliziran. Razlog tome je što su programski jezici standardno segmentirani u dijelove. Ti dijelovi moraju postojati kako bi se minimalizirala veličina izvršnog kôda. Naime, ako imamo jednostavni program koji ne procesira tekst, tada nema smisla u izvršni kôd unositi rutine za procesiranje teksta. Također, standardno je programski kôd brži i manje zahtjevan za izvršavanje.

Osim standardnih programskih jezika, kod kojih se prije izvršavanja kôd mora prevesti u **izvršni oblik** (c, c#, visual basic, c++ itd.), postoje i skriptni jezici. **Skriptni jezici** posebni su po tome što se interpretiraju pri izvođenju, a ne prevode u izvršni oblik. Obično su ti jezici jednostavni za učenje i izradu, zbog jednostavne semantike i sintakse. Standardno se skripte izvršavaju od početka do kraja slijedno (bez središnje rutine / ulaznog mjesta (engl. *entry point*)/ konstruktora). Često korišten jezik kao alternativa skriptama ljuske je **Perl**. Na nekoliko primjera usporedit će se mogućnosti i kompleksnost skripte ljuske i Perla.

**1. Ponašanje u slučaju problema pri izvršavanju (engl. *job fail*)**

U slučaju neuspješnog izvršavanja neke naredbe kôd treba reagirati i izvršiti naredbu za oporavak. Prvo slijedi primjer skripte ljuske:

```
if ! Naredba_1
then
   Naredba_oporavka
fi
```

Za  Perl taj primjer izgleda ovako:

```
use autodie qw< :all >;
use Try::Tiny;      # TryCatch je bolja mogućnost, ali troši više resursa
try
{
    system("Naredba_1");
}
catch
{
    system("Naredba_oporavka ");
};
```

**2. Izvršavanje pri izlasku iz programa**

Cilj je da se neka naredba izvrši pri završetku programa.

Slijedi primjer skripte ljuske:

```
trap "Naredba_1" EXIT
```

A zatim primjer za Perl:

```
END
{
    system("Naredba_1");
}
```

Iako se čini da oba kôda rade istu stvar, potrebno je napomenuti da je ponašanje značajno drugačije. Naime, naredba u **skripti ljuske** će se izvršiti bez obzira što je razlog završavanja izvršavanja skripte (sve osim kill -9). Kod **Perla** se naredba neće izvršiti u nizu izlazaka iz programa, npr. ako se sa **exec** poziva izvršavanje drugog programa, ako se prekida izvršavanje signalom i slično.

**3. Obrada izlaza neke naredbe**

Cilj je obraditi svaku liniju koja je rezultat izvršavanja određene naredbe. Ovdje postoji velika razlika u načinu obrade, s obzirom na mogućnost naredbe da na izlazu kreira linije s ili bez razmaka. Primjer **bash** - kada ne postoji mogućnost razmaka:

```
for linija in $(Naredba_1)
do
    Naredba_Procesiranja"$linija"
Done
```

Slijedi primjer s mogućnošću razmaka:

```
OIFS="$IFS"
IFS="
"
for linija in $( Naredba_1)
do
    Naredba_Procesiranja "$linija"
done
IFS="$OIFS"
```

Ovaj je oblik sličan u složenosti Perl varijanti:

```
use autodie qw< :all >;
open(PIPE, "run some command|");
while ( <PIPE> )
{
    chomp;
   system(qw< process each >, $_);
}
close(PIPE);
```

Uglavnom se može vidjeti da je za osnovne namjene lakše, jednostavnije i intuitivnije koristiti skripte ljuske. Čim bi složenost rasla, razlika bi se smanjivala, dok ne bi nastupilo stanje gdje je lakše koristiti Perl. Ipak, i pri manjim problemima postoje situacije gdje je bolje koristiti Perl:

- Kôd će biti brži, zato što za mnoge radnje ne kreira novi proces kao što je to slučaj u skriptama ljuske.
- Procesiranje nizova teksta je pomalo nekonzistentno u skriptama ljuske, a Perl to obavlja vrhunski i pouzdano.
- Uvjetovanje u skriptama ljuske svakako ima složenu sintaksu.
- Upotreba navodnika u skriptama ljuske zahtijeva puno truda ako se u citatu nalaze specijalni znakovi.
- Nizovi i hashovi puno su bolje riješeni u Perlu.
- CPAN – repozitorij modula omogućava da se riješe kompleksni problemi kada postoji modul koji ima potrebne funkcionalnosti.

### 4.3.2. Korištenje Perlovih modula

Perl je, kako je ranije objašnjeno, programski jezik koji se interpretira pri izvođenju. Perl je dio standardne distribucije i dolazi u paketu istog imena. Iako ima značajno šire mogućnosti od skripti ljuske, uvjet za datoteku da bude **Perl skripta** sličan je kao i za skripte ljuske - prva linija mora pokazivati na interpreter za Perl. Perl interpreter može se jednostavno naći naredbom `which perl`:

```
# which perl
/usr/bin/perl
```

Dakle, Perl skripte trebaju počinjati s:

```
#!/usr/bin/perl
```

Perl će se instalirati sa sustavom na Debian distribucijama, ali ne i njegova dokumentacija. Iako postoji naredba `perldoc`, ta je naredba beskorisna dok se ne instalira **perl-doc paket**. Nakon toga se pomoću naredbe `perldoc` može doći do dokumentacije programskog jezika. Kako se može i očekivati, perldoc je samo osnovna naredba, a s nizom mogućnosti može se pribaviti pomoć o raznim elementima Perla. Najkorisnije su mogućnosti navedene u donjoj tablici.

| Mogućnost | Objašnjenje |
|---|---|
| **-f <ime_funkcije>** | Vraća dokumentaciju imenovane funkcije. |
| **q <predložak>** | Vraća sve FAQ (engl. *Frequently Asked Question*) unose u kojima je pronađen predložak upita. |
| **-v <Varijabla>** | Perl sadrži predefinirane varijable. Ovim se upitom dobiju informacije o njima, primjerice @ARGV. |
| **perlintro** | Kratki, ali vrlo informativni uvod u Perl programski jezik. |

Tablica 34 - Često korištene mogućnosti naredbe perldoc

**Paket** je u Perlu jednostavno kolekcija kôda izdvojena jedinstvenim imenovanjem (engl. *namespace*). Paketi omogućavaju konstrukciju **modula** koji se zbog toga neće kositi sa varijablama i funkcijama van modula. **Perl modul** je paket (standardno namijenjen široj uporabi) u datoteci biblioteke koja je istog imena kao i paket (sa sufiksom .pm). Modul objedinjuje kôd koji pruža određenu funkcionalnost i koji je tako izdvojen zbog potencijala višekratnog korištenja. Primjer modul imena „Lacadamy.pm“:

```
$ cat Lacadamy.pm
#!/usr/bin/perl

package Lacadamy;
sub pocetak {
   print "Hello $_[0]\n"
}
sub kraj {
   print "World $_[0]\n"
}
1;
```

Modul može biti pozvan pozivima **require** i **use**. Glavna je razlika da se kod **require** mora koristiti potpuna klasifikacija pri pozivu funkcija, a kod **use** ne. Primjeri za oba poziva:

```
$ cat require_exampl
#!/usr/bin/perl

require Lacadamy;

Lacadamy::pocetak( "a" );
Lacadamy::kraj( "b" );

$ cat use_exampl
#!/usr/bin/perl

use Lacadamy;

pocetak( "a" );
kraj( "b" );
```

Ipak, ako pokušamo izvršiti ove dvije Perl skripte pojavit će se greška:

```
$ perl require_exampl
Undefined subroutine &Foo::pocetak called at require_examp line 5.
$ perl use_exampl
Undefined subroutine &main::pocetak called at use_examp line 5.
```

Modul još nije instaliran i zbog toga nije dostupan njegov kôd, odnosno funkcije. Da bi modul mogao biti instaliran mora postajati više datoteka koje opisuju modul. **Alat h2xs** generira direktorij sa svim potrebnim datotekama. Kada se želi napraviti modul imena Lacadamy, tada se poziva ova naredba:

```
root@l201:/home/linux1/perl_modul# h2xs -AX -n  Lacadamy
Defaulting to backwards compatibility with perl 5.20.2
If you intend this module to be compatible with earlier perl versions,
please
specify a minimum perl version with the -b option.
Writing Lacadamy/lib/Lacadamy.pm
Writing Lacadamy/Makefile.PL
Writing Lacadamy/README
Writing Lacadamy/t/Lacadamy.t
Writing Lacadamy/Changes
Writing Lacadamy/MANIFEST
```

Kao što je vidljivo, u direktoriju se stvara i **Lacadamy /lib/ Lacadamy.pm** datoteka, koja je okvir za modul koji želimo izraditi. Budući da nas u testiranju ne smetaju upozorenja koja će stvoriti minimalna datoteka bez očekivanih parametara, ta datoteka može se jednostavno zamijeniti s ranije stvorenom:

```
root@l201:/home/linux1/perl_modul# mv Lacadamy.pm
Lacadamy/lib/Lacadamy.pm
```

Sada su spremne sve potrebne datoteke te je dovoljno `s perl Makefile.PL` i `make` napraviti modul, a `s make install` ga instalirati:

```
root@l201:/home/linux1/perl_modul/Lacadamy# perl Makefile.PL
Checking if your kit is complete...
Looks good
WARNING: Setting VERSION via file 'lib/Lacadamy.pm' failed
 at /usr/share/perl/5.20/ExtUtils/MakeMaker.pm line 647.
WARNING: Setting ABSTRACT via file 'lib/Lacadamy.pm' failed
 at /usr/share/perl/5.20/ExtUtils/MakeMaker.pm line 659.
Generating a Unix-style Makefile
Writing Makefile for Lacadamy
Writing MYMETA.yml and MYMETA.json
root@l201:/home/linux1/perl_modul/Lacadamy# make
cp lib/Lacadamy.pm blib/lib/Lacadamy.pm
root@l201:/home/linux1/perl_modul/Lacadamy# make install
Installing /usr/local/share/perl/5.20.2/Lacadamy.pm
```

Kao i kod običnih paketa, sve naredbe osim zadnje mogu se (i poželjno je da tako i bude) izvršiti bez administratorskih ovlasti. Ako se nakon instalacije modula ponovno pokušaju izvršiti naredbe skripte s pozivima modula, dobije se sljedeći rezultat:

```
root@l201:/home/linux1/perl_modul# perl require_exampl
Hello a
World b
root@l201:/home/linux1/perl_modul# perl use_exampl
 at /usr/share/perl/5.20/ExtUtils/MakeMaker.pm line 647.
```

Razlog zbog kojeg primjer **use** ne radi je nedostatak naredbe za eksplicitno dijeljenje funkcija. Ako se dodaju sljedeće linije u **Lacadamy.pm**:

```
require Exporter;
@ISA = qw(Exporter);
@EXPORT = qw(bar blat);require Exporter;
```

te se preimenuje u **Lacadamy_3.pm**, zatim prilagodi **use_exampl** da koristi taj modul, tada će po instalaciji tog modula skripta **use_exampl** ispravno raditi:

```
root@l201:/home/linux1/perl_modul/new# perl use_exampl
Hello a
World b
```

### 4.3.3. Instalacija Perlovih modula

Prvi način za instalaciju Perl modula prikazan je i u prethodnom poglavlju. To je instalacija iz izvornog kôda, koji se pribavlja u obliku **tarballa**. Nakon toga slijedi 5 koraka:

| Korak | Naredba |
|---|---|
| **(1) Raspakiravanje** | tar –xfvz Modul.tar.gz. |
| **(2) Konfiguracija** | perl Makefile.pl |
| **(3) Izrada modula** | make |
| **(4) Testiranje** | make test |
| **(5) Instalacija** | make install. |

Tablica 35 - Koraci pri instalaciji perl modula iz izvornog kôda

Budući da uvijek postoji rizik instalacije ili pokretanja zloćudnog kôda treba izbjegavati korištenje administratorskog korisničkog računa za rad, a to je u primjeru naglašeno korištenjem `$su -c` naredbe. U poglavlju 4.3.2 već je bila prikazana instalacija iz izvornog kôda, ali je taj kôd bio lokalno izrađen, a ne pribavljen u obliku tarballa. Na primjeru modula Color: Library prikazat će se svi koraci takve instalacije:

```
$ wget https://cpan.metacpan.org/authors/id/R/RO/ROKR/Color-Library-
0.021.tar.gz
$ tar -xfvz Color-Library-0.021.tar.gz
$ cd Color-Library-0.021/ && perl Makefile.PL
Generating a Unix-style Makefile
Writing Makefile for Color::Library
Writing MYMETA.yml and MYMETA.json
$ make
$ make test
All tests successful.
Files=2, Tests=196,  2 wallclock secs ( 0.02 usr  0.10 sys +  0.14 cusr
0.85 csys =  1.11 CPU)
Result: PASS
$ su -c "make install"
Password:
Appending installation info to /usr/local/lib/i386-linux-
gnu/perl/5.20.2/perllocal.pod
```

Problematično je što za Perl postoje tisuće modula, a neki od njih ovise o drugim modulima. Zato se često dogodi da nakon što je pronađen modul koji ispunjava željenu funkcionalnost te on bude instaliran prije nego isti bude funkcionalan potrebno je dodatno instalirati još cijeli niz modula o kojima ovisi.

Postoji i sigurnosni aspekt problema instalacije iz izvornog kôda. Autor modula lako može dodati zlonamjerni kôd i instalacija takvog modula može omogućiti provalu na računalo. Moduli se mogu instalirati korištenjem paketa Deb ili RPM.

Napredni paketni sustav za **Perl** module je **cpan** alat. Pri prvom korištenju alat postavlja osnovne postavke potrebne za rad i provjerava postoje li svi alati potrebni za njegov rad. Važno je napomenuti da uz osnovnu instalaciju za rad **cpan** treba još samo **make** alat.

**cpan** alat je dosta star (iz 1998.), a pokušaj da ga se nadogradi, proširi i poboljša je propao, pa je umjesto toga napravljen **cpanm (cpanminus)** koji smanjuje količinu nepotrebnih informacija na terminalu pri instalaciji modula. Oba alata rade jako jednostavno:

```
cpan|cpanm <ime_modula>
```

Ovo je naredba za instalaciju modula. Je li neki modul instaliran na sustavu može se provjeriti **perl naredbom** s mogućnostima **-M<ime_modula> -e 1**. Primjer provjere za modul koji je ranije instaliran iz izvornog kôda:

```
$ perl -MColor::Library -e 1
$
```

Naredba samo javlja informacije na izlaz za modul koji nije instaliran. Na primjer:

```
$ perl -MColo2r::Library -e 2
Can't locate Colo2r/Library.pm in @INC (you may need to install the
Colo2r::Library module) (@INC contains: /etc/perl /usr/local/lib/i386-
linux-gnu/perl/5.20.2 /usr/local/share/perl/5.20.2 /usr/lib/i386-linux-
gnu/perl5/5.20 /usr/share/perl5 /usr/lib/i386-linux-gnu/perl/5.20
/usr/share/perl/5.20 /usr/local/lib/site_perl .).
BEGIN failed--compilation aborted.
```

Važno je reći da je modul lako instalirati, ali cpan ne podržava naredbu za deinstalaciju.

## 4.4. Upravljanje i nadzor procesa

### 4.4.1. Ručna provjera aktivnih procesa

**Ručna provjera aktivnih procesa** najčešće se provodi kada su problemi već nastali.

Svi poslovi na Linux sustavu su procesi. Neki od procesa su vrlo kratki, a tek nekoliko naredbi ljuske poput **cd** ne pokreću proces pri izvršavanju. Kada nastanu problemi na sustavu, jedan od prvih koraka u utvrđivanju stanja sustava je pregled stanja aktivnih procesa. Procesi mogu ponekada uzrokovati probleme poput nemogućnosti pokretanja drugih procesa, nezaustavljanja na poziv, potrošnja neočekivano velikih resursa i slično.

Naredbe za pregled stanja aktivnih procesa su `ps`, `pstree`, `top` i `htop`. Jedino `htop` naredba nije dio standardne instalacije *Linux* sustava i instalira se iz istoimenog paketa.

`ps` naredba podržava niz mogućnosti. Za prikaz svih procesa korisnika koristi se poziv bez parametara:

```
# ps
  PID TTY          TIME CMD
 1550 pts/0    00:00:00 su
 1553 pts/0    00:00:02 bash
24508 pts/0    00:00:00 ps
#
```

PID prikazan u gornjem primjeru je **identifikacijski broj procesa** (engl. *process identification number*) i svaki ga proces posjeduje. Ovaj podatak je najvažniji podatak koji prikazuje `ps` naredba zbog toga što se pomoću reference na njegov PID može izvršiti niz operacija nad procesom. Moguće je promijeniti prioritet procesa, zaustaviti ga, prisilno zaustaviti i slično.

Ovakav poziv pokazuje tek djelić svih aktivnih procesa na sustavu, čak i kada je pozvan od administratora sustava, budući da većina procesa nije pokrenuta od korisnika, već su automatski pokrenuti i dodijeljeni virtualnom korisniku. Za prikaz svih procesa od svih korisnika koriste se mogućnosti `-A`, `-e` ili `ax`. Svaki od ovih poziva javlja sve procese, ali se razlikuju u formatu prikaza. Svaka linija opisuje jedan aktivan proces. Ako se proslijedi izlaz iz ovih naredbi u brojač linija, riječi i znakova vidljivo je da svaka od navedenih naredbi javlja isti broj procesa. Na primjer:

```
# ps ax|wc
    163     931    9137
# ps -e|wc
    163     652    5769
# ps -A|wc
    163     652    5769
```

Broj od 163 procesa je očekivan na poslužitelju koji je malo aktivan. Za filtriranje podataka iz `ps` naredbe koristi se `grep` naredba, ali budući da `ps` naredba daje podatke o svim aktivnim procesima potrebno je filtrirati i samu naredbu koja sadrži `grep`:

```
# ps -ef|grep -i gnome-shell|grep -v grep
linux1    1192  1014  0 Jan12 ?        00:03:04 /usr/bin/gnome-shell
linux1    1214     1  0 Jan11 ?        00:00:00 /usr/lib/gnome-
shell/gnome-shell-calendar-server
```

Najvažnije mogućnosti naredbe `ps` navedene su u sljedećoj tablici:

| Mogućnost | Objašnjenje |
|---|---|
| **-e, -ef, -eF, -ely, ax, aux** | Prikaz svih procesa. Svaki poziv se razlikuje na nekin način u format prikaza. |
| **-ejH, axjf** | Prikaz stabla procesa. |
| **-eLf, axms** | Prikaz dretvi. |
| **U root –u root u** | Prikaz svih procesa pokrenutih pod root korisnikom. |
| **<broj>, -<broj>** | Prikaz procesa sa PID-om <broj>. |

| | |
|---|---|
| **-s <sjednica>** | Prikaz procesa određene sjednice. |
| **-t <terminal>** | Prikaz procesa određenog terminal. |
| **-u <korisnik>, -U <korisnik>** | Prikaz procesa korisnika. |

Tablica 36 - Mogućnosti naredbe ps

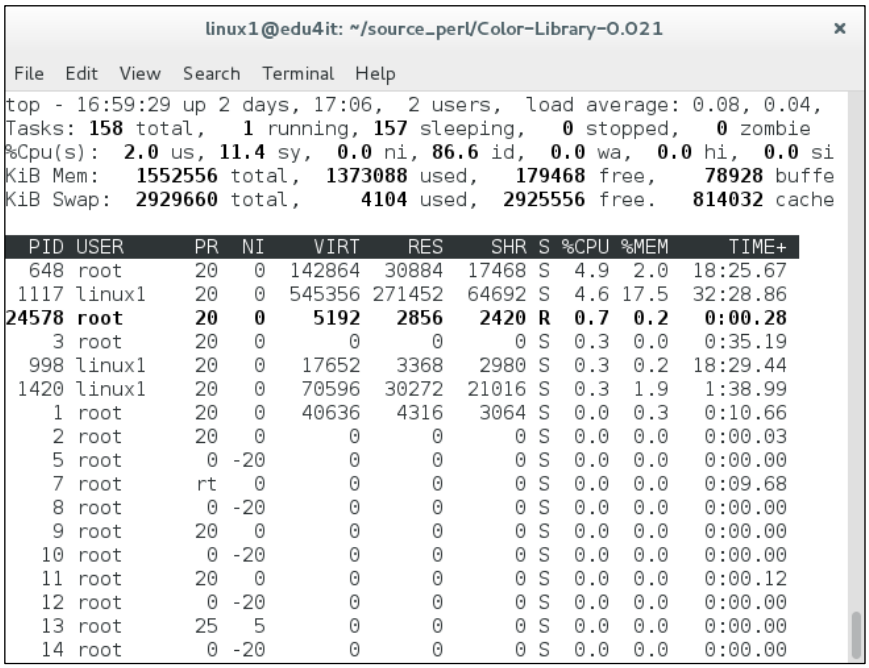
`pstree` naredba prikazuje stablo procesa. Naime, svi procesi imaju jednog zajedničkog pretka **systemd** koji ima PID vrijednost 1. **systemd** proces pokreće jezgra kada ona završi učitavanje.

```
systemd-+-ModemManager-+-{gdbus}
        |               `-{gmain}
        |-NetworkManager-+-dhclient
        |                |-{NetworkManager}
        |                |-{gdbus}
        |                `-{gmain}
        |-VBoxClient---VBoxClient---{SHCLIP}
        |-VBoxClient---VBoxClient
        |-VBoxClient---VBoxClient---{X11 events}
        |-VBoxClient---VBoxClient-+-{dndHGCM}
        |                         `-{dndX11}
        |-VBoxService-+-{automount}
        |             |-{control}
        |             |-{cpuhotplug}
        |             |-{memballoon}
        |             |-{timesync}
        |             |-{vminfo}
        |             `-{vmstats}
        |-accounts-daemon-+-{gdbus}
        |                 `-{gmain}
        |-acpid
        |-agetty
        |-at-spi-bus-laun-+-dbus-daemon
```

Ovaj pogled može ukazati na procese s neobično velikim brojem djece (engl. *children*) procesa, ili na procese koji se rekurzivno pozivaju. Svrha ove naredbe je primarno pronalaženje roditelja ili djece određenog procesa.

Kada nastanu problemi na sustavu koji rezultiraju usporenjem sustava obično je u pitanju problem da neki proces koristi neočekivano velike količine resursa. Naredba za prikaz potrošnje resursa su `top` i `htop`.

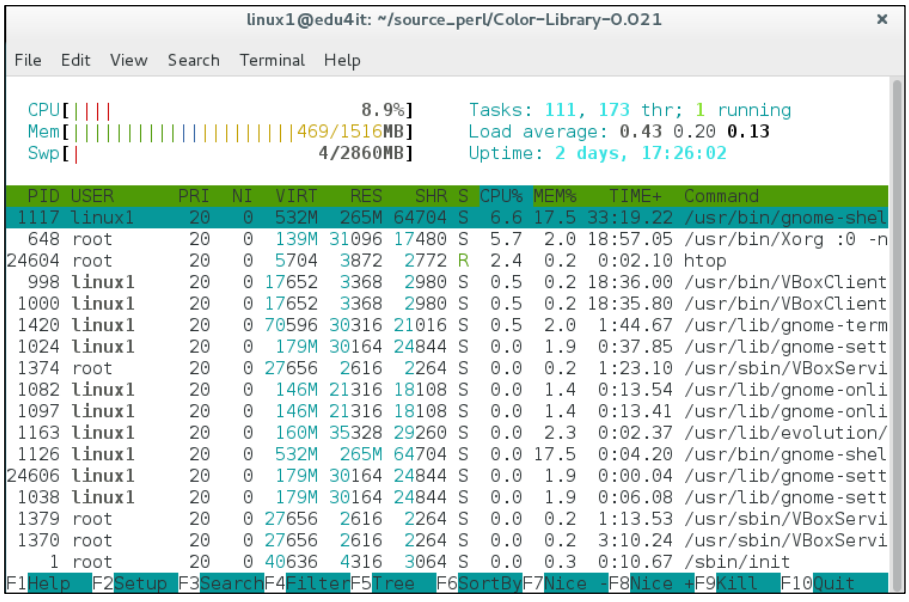
Primjer poziva naredbe top prikazan je na slici 5.




Slika 5 - Primjer poziva naredbe top

Pozivom naredbe pokreće se interaktivna okolina u kojoj se prikazuju procesi i njihova potrošnja resursa. Iz naredbe `top` izlazi se pomoću tipke **q** ili **CTRL+c**. Pregled mogućih naredbi u okolini dobiva se pozivom pomoću tipke **h**. Po zadanim postavkama procesi se osvježavaju svake sekunde, a sortirani su po potrošnji procesora. Pomoću ove naredbe je puno lakše pronaći proces koji se „devijantno“ ponaša.

Naredba `htop` naprednija je od naredbe `top` i nudi pregledniji pogled na stanje sustava, ali i troši više resursa. Poziv naredbe `htop` je prikazan na slici 6:




Slika 6 - Primjer poziva naredbe htop

Naredba `htop` funkcionira isto kao `top` naredba, ali velika prednost je donja naredbena traka koja se interaktivno mijenja i omogućava brži i intuitivniji rad. Naredbe `htop` i `top` uz pregled omogućavaju i upravljanje procesima. Pomoću njih je moguće promijeniti prioritet ili zaustaviti proces koji se „sumnjivo“ ponaša izravno u interaktivnoj okolini.

### 4.4.2. Automatski nadzor stanja procesa i odlučivanje

**Automatski nadzor procesa** i odlučivanje izvode se nad procesima koji učestalo uzrokuju probleme ili kada se problemi očekuju (testna faza implementacije). Sustavi za automatski nadzor servisa i aktivnosti pokušavaju prepoznati neobično ili pogrešno ponašanje dijelova sustava i spriječiti nastanak štete.

Na poslužitelju je moguća pojava problema u radu. Probleme kmogu uzrokovati neobično (netipično, nestandardno, neadekvatno ili pogrešno) konfigurirana aplikacija, nepouzdan hardver, nepouzdana mreža (tehnički mana okoline, ali utječe na rad poslužitelja). Neke od tih mana se mogu trajno ukloniti (npr. zamjena hardvera pouzdanijim ), ali iz raznih razloga (najčešće financijskih) često ih nije moguće sve ukloniti. Budući da u slučaju produkcijskih poslužitelja te mane mogu rezultirati padom sustava, nedostupnošću usluga ili gubitkom podataka, poželjno je postaviti dodatne zaštite od poznatih prijetnji pouzdanosti servisa.

Kada je poznato koji su rizici mogući na sustavu moguće je napisati i postaviti testove koji provjeravaju trenutno stanje i obavještavaju o „rizičnim“ promjenama. Alat izbora *Linux* administratora za ovu namjenu su **skripte ljuske**, **Perl skriptni jezik** ili neki drugi programski jezik. U kojim slučajevima koristiti svaki od ovih alata objašnjeno je u poglavlju 4.3.1.

U nastavku će biti prikazano nekoliko primjera kôda za automatski nadzor. U svakom primjeru će biti objašnjen kôd i ograničenja istog.

**(1) Provjera postojanja procesa**

```
#!/bin/bash
PROCESS=$1
if ps aux | grep "$PROCESS" | grep -v grep| grep –v $0 >/dev/null ; then
        echo Process $PROCESS is running > /var/log/$PROCESS.log
else
        echo Process $PROCESS is stopped – Restarting it ... >
/var/log/$PROCESS.log
        /etc/rc.d/init.d/$PROCESS start > /dev/null
fi
```

Ovaj program prihvaća parametar koji je ime programa i provjerava postoji li proces tog programa. Kada taj program postoji, samo zapisuje da je provjera napravljena, a kada proces ne postoji, onda se poruka s tom informacijom zapisuje u log datoteku i pokušava se pokrenuti proces.

Ova je skripta primjer programa koji se periodički pokreću pomoću **crona** i kontroliraju servise koji su skloni ispadu ili koji su ključni i ne smiju imati vrijeme nedostupnosti.

Može se uočiti da ne postoji naredba za zapisivanje vremenske oznake u datoteku, stoga će biti teško odrediti kada je došlo do problema. Takve stvari treba uočiti i ispraviti na vrijeme, budući da samo potpuna skripta nudi pravu pomoć u slučaju problema. Primjer sa zapisom vremena bi bio:

```
#!/bin/bash
PROCESS=$1
if ps aux | grep "$PROCESS" | grep -v grep | grep –v $0 >/dev/null ;
then
       echo Process $PROCESS is running >> /var/log/$PROCESS.log
      /bin/date >> /var/log/$PROCESS.log
else
       echo Process $PROCESS is stopped – Restarting it ... >>
/var/log/$PROCESS.log
      /bin/date >> /var/log/$PROCESS.log
       /etc/rc.d/init.d/$PROCESS start > /dev/null
fi
```

**(2) Provjera komunikacijskog kanala**

```
#!/bin/bash
while (true)
do
#Prikupljanje trajanja od 10 ping zahtjeva
   x=$(ping -c 10 $1 | cut -d"=" -f4 | tail -n +2 |head | sed "s/ms//")
#Tražimo vremena dulja od 15 ms
   for times in $x
     do
         dectimes=$(echo $times | cut -d. -f1) # get an integer
         if [ $(($dectimes-15)) -gt 0 ]; then
            echo Ping dulji od 15ms:  $times
         fi
       done
done
```

Ova skripta može kontrolirati komunikacijski kanal između dva računala koja zajedno funkcioniraju, ostvarujući funkcionalnost preko mreže. Skripta može i provjeravati brzinu komunikacije prema usmjerniku ili prema DNS poslužitelju. Nedostatak skripte je da provjerava samo **ping**. Budući da **ping** koristi specijalni protokol, problemi u ovoj skripti ne moraju značiti i stvarne probleme u produkciji. Ovisno o cilju provjere umjesto **pinga** mogu se koristiti drugi alati poput **telneta**.

**(3) Parsiranje logova i prosljeđivanje važnih informacija**

```
#!/usr/bin/perl
$LOGFILE="/tmp/lastlog";
$line="0";
system("last> $LOGFILE");

open (MAIL, "| mail root");

if (open (FILE,"$LOGFILE")) {

        while ($line ne "") {
             $line=;
              if ($line =~ still) {
                  print MAIL $line;
              }
        }
}
close MAIL;
close FILE;
```

Ova skripta čita **/var/run/utmp** datoteku pomoću naredbe `last`, zatim šalje administratoru podatke o tome tko je sve trenutno prijavljen na sustav putem elektroničke pošte. Ovo je primjer skripte koju **cron** može pokretati svakih sat vremena.

Nedostatak skripte je što generira puno poruka koje netko mora pročitati da bi bila korisna. Poboljšanje bi bilo slanje poruka samo u posebnim okolnostima.

Ovo su samo tri osnovna primjera koji prikazuju kako se poslužitelj može dodatno zaštititi. Osnovni problem sa skriptama koje piše sam administrator je što su one reakcija na već nastali problem. Da bi došli na ideju za pisanje skripte koja nešto (servis, uređaj, modul ili neku njihovu karakteristiku) nadzire prvo moraju postojati indicije da je to potrebno. Najčešće u obliku (barem jednog) ispada sustava.

### 4.4.3. Automatski nadzor servisa i aktivnosti na poslužitelju

Postoje brojni alati za nadzor servisa na poslužitelju. Jedna od motivacija pri razvoju systemd je bila i mogućnost opsežnijeg upravljanja servisima, poput automatskog pokretanja u slučaju ispada. Alati su doista brojni: OpenNMS, SysUsage, brainypdm, PCP, KDE system guard, Munin, Nagios, Zenoss, Cacti, Zabbix, nmon, conly, monit i mnogi drugi. Ovi se alati razlikuju u mogućnostima, potrošnji resursa i cilju. Neki su namijenjeni samo nadgledanju i obavještavanju, neki analizi mrežnih aktivnosti, neki prate samo procese i tako dalje.

Ono što većina naprednijih alata nudi jest sučelje za jednostavan nadzor nekog servisa. Oni omogućavaju aktivaciju skripti za nadzor, koje bi inače administrator morao sam napisati. Osim toga, testovi koje provode su opsežni i prilagođeni potrebama nadzora.

U nastavku će biti prikazan servis **monit** – svi koraci od instalacije do rada u okolini.

**monit** se instalira iz istoimenog paketa:

```
# apt-get install monit -y
```

Nakon instalacije **monit** servis se pokreće, ali ne sa upravljačkim web sučeljem. Za rad sučelja poželjno ga je zaštititi sa zaporkom pa je potrebno i izraditi valjani certifikat. Po zadanim postavkama certifikat treba biti u **/var/certs/monit.pem** datoteci. Prvo je potrebno stvoriti datoteku za kreiranje certifikata:

```
root@l201:/var/certs#vim /var/certs/monit.cnf

# create RSA certs - Server

RANDFILE = ./openssl.rnd

[ req ]
default_bits = 2048
encrypt_key = yes
distinguished_name = req_dn
x509_extensions = cert_type

[ req_dn ]
countryName = Country Name (2 letter code)
countryName_default = HR

stateOrProvinceName             = State or Province Name (full name)
stateOrProvinceName_default     = Zagreb

localityName                    = Locality Name (eg, city)
localityName_default            = Cvjetno

organizationName                = Organization Name (eg, company)
organizationName_default        = SRCE

organizationalUnitName          = Organizational Unit Name (eg, section)
organizationalUnitName_default  = l201

commonName                      = Common Name (Name of your server)
commonName_default              = l201.srce.local

emailAddress                    = Email Address
emailAddress_default            = root@l201.srce.local

[ cert_type ]
nsCertType = server
```

Nakon toga se slijedom od 4 naredbe stvara valjani certifikat:

```
openssl req -new -x509 -days 365 -nodes -config ./monit.cnf -out
/var/certs/monit.pem -keyout /var/certs/monit.pem
openssl gendh 1024 >> /var/certs/monit.pem
openssl x509 -subject -dates -fingerprint -noout -in
/var/certs/monit.pem
chmod 600 /var/certs/monit.pem
```

Sada je moguće pokrenuti servis:

```
# systemctl start monit
# systemctl status monit
● monit.service - LSB: service and resource monitoring daemon
   Loaded: loaded (/etc/init.d/monit)
   Active: active (running) since Mon 2017-01-11 11:24:47 CET; 8min ago
  Process: 7957 ExecStop=/etc/init.d/monit stop (code=exited,
status=0/SUCCESS)
  Process: 8580 ExecStart=/etc/init.d/monit start (code=exited,
status=0/SUCCESS)
   CGroup: /system.slice/monit.service
           └─8584 /usr/bin/monit -c /etc/monit/monitrc

Jan 11 11:24:47 l201 systemd[1]: Started LSB: service and resource
monitoring daemon.
Jan 11 11:24:47 l201 monit[8580]: Starting daemon monitor: monit.
```

Pri instalaciji se stvara standardna konfiguracijska **/etc/monit/monitrc** datoteka koja predstavlja središnju konfiguracijsku datoteka servisa. U zadanim postavkama monit konfiguracija ne nadzire ništa. Slijedi primjer **/etc/monit/monitrc** datoteke:

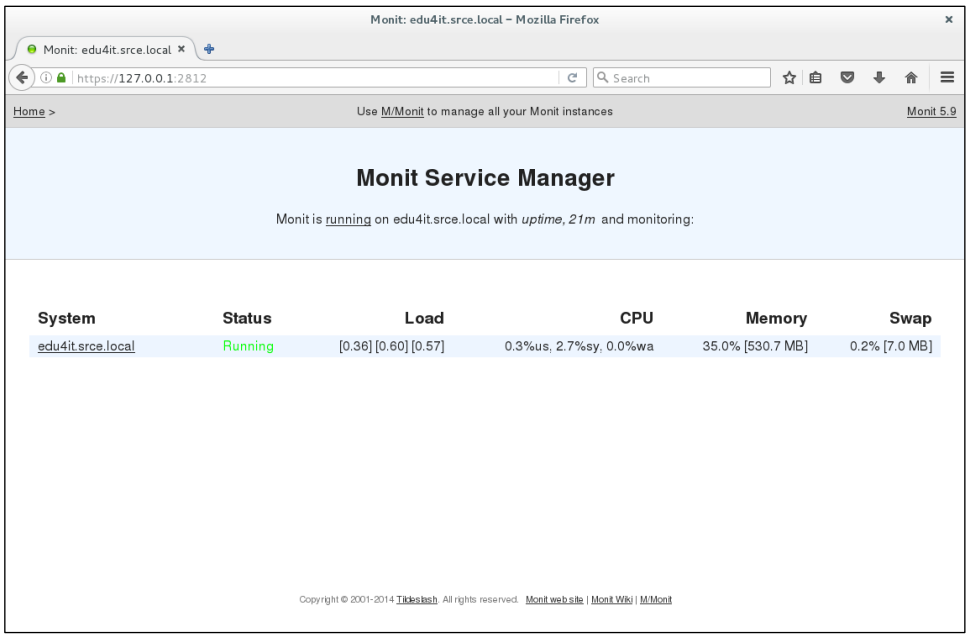
```
# less /etc/monit/monitrc_orig|grep -v "^#"
  set daemon 120
  set logfile /var/log/monit.log
  set idfile /var/lib/monit/id
  set statefile /var/lib/monit/state
  set eventqueue
      basedir /var/lib/monit/events
      slots 100
   include /etc/monit/conf.d/*
```

U primjeru je navedeno da se monit pokreče kao pozadinski proces svakih 120 sekundi vršeći provjere. Definirane su također datoteka za zapisivanje aktivnosti **monit** servisa, pohranu PID-a servisa i stanja servisa. Zatim je definirano gdje (/**var/lib/monit/events**) se pohranjuju događaji i koliko njih se pohranjuje. Na kraju je zadano da se učitaju datoteke smještene u **/etc/monit/conf.d/** direktoriju.

Kao što je vidljivo, ova konfiguracija samo definira ponašanje servisa. Ako se želi aktivirati i web sučelje za **monit**, potrebno je dodati sljedeće linije u konfiguraciju:

```
set httpd port 2812 and
 SSL ENABLE
 PEMFILE /var/certs/monit.pem
 allow admin:l201
```

Kada se ponovno pokrene servis, moguć je pristup preko **porta 2812,** uz korisničko ime **admin** i lozinku **l201** te putem web preglednika monit konzoli kako je prikazano na slici 7.




Slika 7 - Izgled monit web konzole

Kao što je vidljivo, **monit** trenutno nadzire samo stanje cijelog sustava. Budući da je ranije u konfiguraciji postavljeno čitanje konfiguracije u **direktoriju /etc/monit/conf.d/,** u njega će se smjestiti datoteke s konfiguracijom za nadzor servisa. Nadzirat će se servisi **ssh** i **ftp**. Slijede pravila za ssh servis:

```
# vim ssh
check process sshd with pidfile /var/run/sshd.pid
start program  "/etc/init.d/sshd start"
stop program  "/etc/init.d/sshd stop"
if failed port 22 protocol ssh then restart
```

U ovom primjeru se zadaje nadzor **sshd** procesa koji koristi **/var/run/sshd.pid** kao datoteku za identifikacijski broj procesa. Zadano je da se proces pokreče ukoliko na portu 22 ne sluša **ssh** protokol.

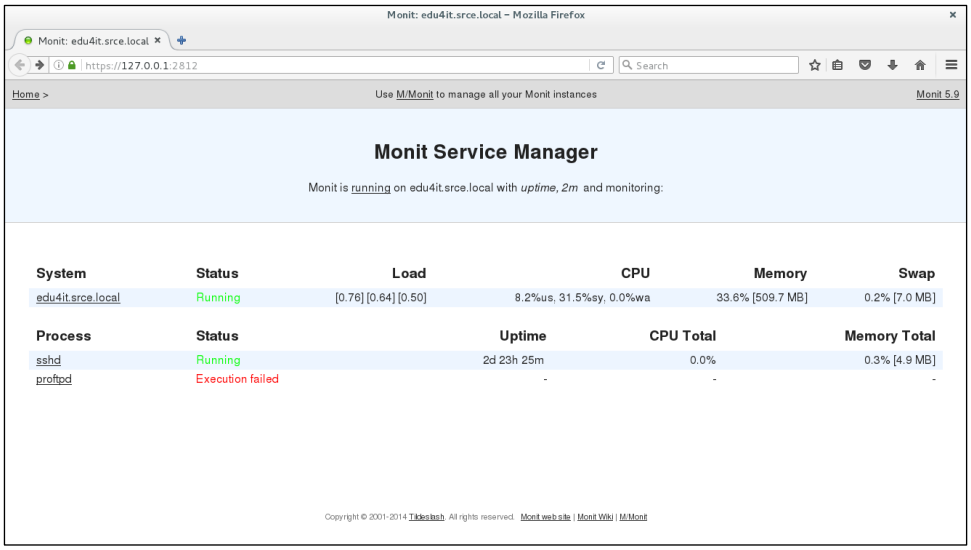
Važno je napomenuti da je sve pozive izvršnih datoteka potrebno provesti koristeći potpunu putanju jer servis pri pokretanju ne koristi osnovne postavke ljuske već ima samo ograničeni pristup.

Zatim za **ftp** servis:

```
# vim proftpd
check process proftpd with pidfile /var/run/proftpd.pid
   start program = "/etc/init.d/proftpd start"
   stop program  = "/etc/init.d/proftpd stop"
   if failed port 21 protocol ftp then restart
```

U ovom primjeru se zadaje nadzor **proftpd** procesa koji koristi **/var/run/sshd.pid** kao datoteku za identifikacijski broj procesa. Zadano je da se proces pokreče ukoliko na portu 21 ne sluša **ftp** protokol.

Potrebno je ponovno pokrenuti monit servis i promotriti web sučelje. Kako je prikazano na slici 8.




Slika 8 - Popis servisa u monit web konzoli

Dakle, vidljivo je da ssh servis radi, ali nije uspjela provjera **ftp** servisa. Budući da **proftpd** nije instaliran, to je i očekivano. **monit** može obavljati još brojne funkcije, kao što su javljanje na liste, provjera aktivnosti, postavljanje praga, automatski restart i slično.

Danas su **monit** i slični programi obavezno dio instalacije produkcijskih poslužitelja. Sve mogućnosti koje imaju trebaju biti ručno prilagođene potrebama pojedinih poslužitelja/servisa, ali taj se trud višestruko isplati, zbog dobre zaštite od ispada ili smanjenja vremena reakcije na ispad.

## 4.5. Vježba: Instalacija i prilagodba sustava monit

1. Prije početka rada odaberite sliku stanja virtualnoga računala (engl. Snapshot) za početak ove vježbe.
2. Prijavite se na računalo kao korisnik l201. U GUI-ju pokrenite **Terminal** (**Activities** → **Terminal**). Izvršite su - naredbu da postanete administrator (lozinka: L201).
3. Provjerite jesu li servisi **apache2 i ssh** pokrenuti na računalu.

---

4. Instalirajte monit naredbom:

   `# apt-get install monit -y`

5. Potrebno je prilagoditi monit:

   Prvi korak je kreiranje certifikata za https komunikaciju. Izvršite sljedeće naredbe:

   `root@l201# vim /var/certs/monit.cnf`

   (unesite sljedeče u knofiguracijsku datoteku)

```
# create RSA certs - Server
RANDFILE = ./openssl.rnd

[ req ]
default_bits = 2048
encrypt_key = yes
distinguished_name = req_dn
x509_extensions = cert_type

[ req_dn ]
countryName = Country Name (2 letter code)
countryName_default = HR

stateOrProvinceName             = State or Province Name (full name)
stateOrProvinceName_default     = Zagreb

localityName                    = Locality Name (eg, city)
localityName_default            = Cvjetno

organizationName                = Organization Name (eg, company)

organizationName_default        = SRCE

organizationalUnitName          = Organizational Unit Name (eg, section)
organizationalUnitName_default  = l201

commonName                      = Common Name (Name of your server)
commonName_default              = l201.srce.local

emailAddress                    = Email Address
emailAddress_default            = root@l201.srce.local
```

```
[ cert_type ]
nsCertType = server
```

Nakon toga se slijedom od 4 naredbe stvara valjani certifikat:

```
# openssl req -new -x509 -days 365 -nodes -config ./monit.cnf -out
/var/certs/monit.pem -keyout /var/certs/monit.pem
# openssl gendh 1024 >> /var/certs/monit.pem
# openssl x509 -subject -dates -fingerprint -noout -in /var/certs/monit.pem
# chmod 600 /var/certs/monit.pem
```

*U posljednjoj openssl naredbi odaberite proizvoljne vrijednosti. Bez obzira na to kakve vrijednosti odabrali, web-preglednik će identificirati samopotpisani certifikat kao sigurnosnu opasnost i morat će se dodati iznimka za pristup.

6. Sada je potrebno urediti datoteku **/etc/monit/monitrc**:

```
# cp /etc/monit/monitrc /etc/monit/monitrc_old
# cp /dev/null /etc/monit/monitrc
# vim /etc/monit/monitrc
```

Dodajemo sljedeće linije:

```
set daemon 60
set logfile /var/log/monit.log set idfile /var/lib/monit/id
set statefile /var/lib/monit/state set eventqueue

basedir /var/lib/monit/events slots 100
include /etc/monit/conf.d/* set httpd port 2812 and
SSL ENABLE
PEMFILE /var/certs/monit.pem allow admin:L201
```

7. Ponovno pokrenite monit.

---

8. Testirajte pristup do web-sučelja nove skupine spremišta.

   `# firefox https://127.0.0.1:2812/ &`

   U pregledniku odaberite Advanced→Add exception→Confirm security exception. **Korisničko ime**: **admin**, **Lozinka**: **L201**.

   Što sve postoji u sučelju?

---

9. Potrebno je namjestiti nadzor **ssh** servisa, tako da se dodaju sljedeće linije u **/etc/monit/monitrc** na kraj datoteke:

```
check process sshd with pidfile /var/run/sshd.pid start program
"/bin/systemctl start ssh" stop program "/bin/systemctl stop ssh"
if failed port 22 protocol ssh then restart
```

10. Ponovno pokrenite servis **monit** i provjerite sadržaj web-sučelja.

(Potrebno je nekoliko puta osvježiti preglednik da bi izmjene bile vidljive.)

11. Otvorite novi teminal za **root** korisnika. U njemu pokrenite

`# tail -f /var/log/monit.log` i pratite što se događa u tom terminalu.

12. Isključite **ssh** servis i provjerite stanje sustava u web-sučelju (**[CTRL]+[F5]** forsira osvježavanje preglednika). Što se promijenilo?

Pratite i stanje u terminalu gdje se izvršava tail naredba. Što se tamo događa?

## 4.6. Vježba: Perl skripte

1. Prije početka rada odaberite sliku stanja virtualnoga računala (engl. Snapshot) za početak ove vježbe.
2. Prijavite se na računalo kao korisnik l201. U GUI-ju pokrenite **Terminal** (**Activities** → **Terminal**). Izvršite su - naredbu da postanete administrator (lozinka: L201).
3. Kao administrator stvorite direktorij /usr/log.
4. Stvorite datoteku **/root/monit_edu.pl** sljedećega sadržaja (dodane su točke da bi bilo lakše vidjeti nove linije):

```
•   #!/usr/bin/perl
•   use strict;
•   use warnings;
•   package l201_edu;
•   sub getLoggingTime {
•   my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$i sdst)=localtime(time);
•   my $nice_timestamp = sprintf ( "%04d%02d%02d %02d:%02d:%02d",
$year+1900,$mon+1,$mday,$hour,$min,$sec);
•   return $nice_timestamp;
•   }
•   our $log;

•   while ( 1 ) {

•   if ( -f "/usr/log/monit_edu.log" ) {
•                       #print "Datoteka /usr/log/monit_edu.log postoji otvaram\n";
•   open($log, ">>", "/usr/log/monit_edu.log") or die "Ne mogu otvoriti
/usr/log/monit_edu.log : $!";
•                       } else {
•                       #print "Datoteka /usr/log/monit_edu.log ne postoji, Kreiram\n";
•   open($log, ">", "/usr/log/monit_edu.log") or die "Ne mogu otvoriti
usr/log/monit_edu.log: $!";
•   }
•   if(-f "/var/run/sshd.pid") {
•               my $timestamp = getLoggingTime();
•                       print $log "$timestamp ssh Radi \n";
•   sleep(2);
•   }           else {
•   print $log "Datoteka /var/run/sshd.pid ne postoji!!!\n";
•                       sleep (5);
•   system ('systemctl start ssh') or print "NE mogu pokrenuti ssh";
•   }
•   close $log; #ovdje se zapisuje
•   }
```

5. Pokušajte pokrenuti skriptu.

Zapišite koje ste greške dobili kada ste pokušali pokrenuti skriptu, a krivo ste unijeli zapise datoteke (uz pretpostavku da se to dogodilo).

6. Proučite skriptu. Zapišite što radi (prema Vašoj pretpostavci) koji dio.

7. Pokrenite skriptu (možete ju pokrenuti kao pozadinski proces, a ako se ne pokrene tako, trebat će otvoriti novi terminal).
8. Otvorite dodatni terminal i pratite što se zapisuje u datoteku.

   `/usr/log/monit_edu.log.`

9. Otvorite još jedan terminal i u njemu zaustavite **ssh** servis.

10. Što se zapisalo u log? Zašto je više puta zapisano?

11. Pokušajte se spojiti na **localhost** kao korisnih l201 (preko ssh). Što se dogodilo?

## 4.7. Vježba: Izrada i instalacija Perl modula

(Ova vježba je nastavak vježbe 4.6. i za početak je potrebno uspješno završiti vježbu 4.6.)

1. U trenutačnom direktoriju stvorite direktorij – stablo za izradu **Perl** modula naziva **monit_edu**.

2. Kopirajte **monit_edu.pl** u **./monit_edu/lib/monit_edu.pm** datoteku. Odgovorite da (engl. *yes*) na pitanje želite li prepisati postojeću datoteku.

3. Stvorite paket i instalirajte ga. Ignorirajte upozorenja.

4. Pokušajte pokrenuti instalirani modul. Koja je razlika u odnosu na ponašanje programa pri izravnom pozivu?

## 4.8. Vježba: Instalacija i prilagodba openssh paketa

1. Prije početka rada odaberite sliku stanja virtualnoga računala (engl. Snapshot) za početak ove vježbe.
2. Prijavite se na računalo kao korisnik l201. U GUI-ju pokrenite **Terminal** (**Activities** → **Terminal**). Izvršite su - naredbu da postanete administrator (lozinka: l201).
3. Pomoću naredbe **apt-get** pribavite izvorni kôd paketa **openssh-server** u direktorij **/home/linux1/paketi** (izradite taj direktorij ako ne postoji).

4. Instalirajte pakete potrebne za izradu paketa **openssh-server**.

5. Uđite u direktorij openssh-<verzija>.

6. Izmijenite datoteku sshd_config tako da umjesto

   `„#PermitRooLogin yes“`

   piše

   `„PermitRootLogin no“`

7. Izradite binarnu verziju paketa.

   (Ovaj korak traje dugo – pauza)

8. Zašto je izrada paketa trajala tako dugo? (Hint: Provjerite koji se sve **.deb** paketi nalaze u direktoriju iznad.)

9. Pomoću naredbe **apt-get** uklonite **openssh-server** paket zajedno sa svim konfiguracijskim datotekama.

10. Pomoću naredbe **dpkg** instalirajte kreirani **openssh-server** paket.

11. Provjerite stanje servisa. Pokušajte se pomoću **ssh** spojiti na localhost s korisnikom l201.

12. Pogledajte sadržaj datoteke **/usr/share/doc/openssh- client/examples/sshd_config**. Posebno obratite pažnju na vrijednost parametra **Permitrootlogin**.

# I. Rješenja

**Vježba 1.3.**

6.

# tar xvf linux-5.17.9.tar.gz

7.

# cd linux-5.17.9

9.

initrd.img-5.10.0-17-amd64

System.map-5.10.0-17-amd64

vmlinuz-5.10.0-17-amd64

10.

cp initrd.img-5.10.0-17-amd64 System.map-5.10.0-17-amd64 vmlinuz-5.10.0-17-amd64 /boot/

12.

Dostupna je

13.

Ne radi - napraviti jezgru koja radi nije trivijalna stvar a posebno na virtualnim sustavima...

**Vježba 1.4.**

5. # cp /etc/systemd/system/sshd.service /etc/systemd/system/sshd-second.service

7.

# systemctl start sshd-second.service

# systemctl enable sshd-second.service

8.

# systemctl status sshd-second.service

# systemctl status sshd.service

9.

# ssh l201@127.0.0.1 -p 2222

1.5.

3.

# touch /etc/systemd/system/emacs.service

4.

# chmod 644 /etc/systemd/system/emacs.service

6.

# systemctl daemon-reload

7.

# systemctl start emacs.service

8.

# systemctl enable emacs.service

9.

# systemctl status emacs.service

10.

# systemctl start reboot.target

11.

# systemctl status emacs.service

**Vježba 2.2**.

3.

# parted /dev/sdb --align none

(parted) mklabel msdos

(parted) mkpart p ext2 1 2000

(parted) mkpart p ext3 2000 4000

(parted) mkpart p ext4 4000 6000

(parted) mkpart p xfs 6000 8000

(parted) print

(parted) quit

4.

# mkfs -t ext2 /dev/sdb1

# mkfs -t ext3 /dev/sdb2

# mkfs -t ext4 /dev/sdb3

# mkfs -t xfs /dev/sdb4

5.

# mkdir /mnt/disk1 /mnt/disk2 /mnt/disk3 /mnt/disk4

6.

# mount /dev/sdb1 /mnt/disk1

# mount /dev/sdb2 /mnt/disk2

# mount /dev/sdb3 /mnt/disk3

# mount /dev/sdb4 /mnt/disk4

7.

#df

kapaciteti nisu identicni

8.

# umount /dev/sdb2 /mnt/disk2

# umount /dev/sdb3 /mnt/disk3

# umount /dev/sdb4 /mnt/disk4

# umount /dev/sdb1 /mnt/disk1

9.

# parted /dev/sdb --align none

(parted) rm 2

(parted) print

10.

(parted) resizepart 1 3000

11.

ne možemo jer su slobodni blokovi 3000 do 4000 a posljednja pocinje na 6000 te odatle u parted možemo samo birati završetak.

12.

# mount /dev/sdb1 /mnt/disk1

# df

kapacitet je nepromijenjen jer treba povecati sa resize2fs

13.

# umount /dev/sdb1

# resize2fs /dev/sdb1

(„e2fsck –f /dev/sdb1“)

# mount /dev/sdb1 /mnt/disk1

# df

14.

# umount /mnt/disk1

# parted /dev/sdb --align none

# (parted) rm 1

# (parted) rm 3

# (parted) rm 4

**Vježba 2.3.**

3.

# parted /dev/sdb --align none

(parted) mklabel msdos

(parted) mkpart p ext4 1 1000

(parted) quit

# mkfs.ext4 /dev/sdb1

6.

# df –h

7.

# systemctl restart autofs

8.

# df –h

10.

Pojavi se disk jer ls naredba aktivira automount

Sada nema diska.

**Vježba 3.5.**

3.

```
# parted /dev/sdb --align none
(parted) mklabel msdos
(parted) mkpart e ext4 1 8000
(parted) mkpart l ext4 1 1000
(parted) mkpart l ext4 1000 2000
(parted) mkpart l ext4 2000 3000
(parted) mkpart l ext4 3000 4000
(parted) mkpart l ext4 4000 5000
(parted) mkpart l ext4 5000 6000
(parted) print
```

4.

```
(parted) set 5 raid on
(parted) set 6 raid on
(parted) set 7 raid on
(parted) set 8 raid on
(parted) set 9 raid on
(parted) set 10 raid on
(parted) quit
```

5.

```
# mdadm --create /dev/md0 --level=5 --raid-devices=3 /dev/sdb5 /dev/sdb6 /dev/sdb7
```

6.

```
# mkfs.ext4 /dev/md0
```

7.

```
# mkdir –p /mnt/raid && mount /dev/md0 /mnt/raid
```

8.

# df

velicina je manja od sume particija zbog pariteta

9. Provjerite stanje RAID polja naredbom mdadm --detail <RAID uređaj>.

10.

# mdadm --add /dev/md0 /dev/sdb8 /dev/sdb9 /dev/sdb10

11.

# df

Kapacitet je nepromijenjen dok se ne provede resize2fs naredba

12.

# resize2fs /dev/md0

# df

13.

# touch /mnt/raid/test001

14.

# mdadm /dev/md0 --fail /dev/sdb7

# mdadm --detail /dev/md0

# mdadm /dev/md0 --remove /dev/sdb7

# mdadm --detail /dev/md0

15.

# parted /dev/sdb --align none

(parted) mkpart l 6000 7500

(parted) set 11 raid on

16.

# mdadm --add /dev/md0 /dev/sdb11

17.

Vidjeli smo kako se disk dodaje u polje

# df –h

# mdadm --detail /dev/md0

**Vježba 3.6.**

3.

# parted /dev/sdb --align none

(parted) mklabel msdos

(parted) mkpart p ext4 1 100

(parted) mkpart p ext4 100 600

(parted) mkpart p ext4 600 800

(parted) mkpart l ext4 2000 3000

4.

# pvcreate /dev/sdb{1,2,3}

5.

# pvs

6.

# vgcreate vg_001 /dev/sdb{1,3}

7.

```
# pvs
# vgs
# vgdisplay
```

8.

```
# lvcreate -L 250M vg_001
```

9.

```
# lvs
```

10.

```
# mkfs.ext4 /dev/mapper/vg_001-lvol0
```

11.

```
# mkdir /mnt/lvm && mount /dev/mapper/vg_001-lvol0 /mnt/lvm && df
```

12.

```
# lvextend -l +100%FREE /dev/mapper/vg_001-lvol0
```

13.

```
# resize2fs /dev/mapper/vg_001-lvol0
```

14.

```
# umount /dev/mapper/vg_001-lvol0
# resize2fs /dev/mapper/vg_001-lvol0 200M
```

15.

```
# resize2fs /dev/mapper/vg_001-lvol0 100M
# lvresize -L 100M /dev/mapper/vg_001-lvol0
# mount /dev/mapper/vg_001-lvol0 /mnt/lvm
```

**Vježba 4.5.**

3.

# ps -ef|egrep 'ssh|apache'

ili

# systemctl status apache2

# systemctl status ssh

4. Instalirajte monit naredbom:

# apt-get install monit -y

7.

# systemctl restart monit

8. Testirajte pristup do web-sučelja nove skupine spremišta.

# firefox https://127.0.0.1:2812/ &

U pregledniku odaberite Advanced  Add exception  Confirm security exception. Korisničko ime: admin, Lozinka: edu4it.

Što sve postoji u sučelju?

10.

# systemctl restart monit

13.

Javljaju se poruke o promjeni stasnja servisa.

**Vježba 4.6.**

3.

# mkdir –p /usr/log

5.

# perl /root/monit_edu.pl

6.

Skripta:

1 Počinje definicijom pravila i okoline.

2 Definicija potprograma za pribavljanje formatiranoga datuma.

3 Beskonačna petlja koja provjerava postojanje datoteka za zapisivanje i u ovisnosti o postojanju proces ssh zapisuje u nju i poduzima akcije.

8.

# tail –f /usr/log/l201.log

9.

# systemctl stop ssh

10.

Poruka o zaustavljenom servisu i njegovu pokretanju zapiše se više puta jer je provjera česta pa se više puta pokuša izvršiti tijekom jednoga pokretanja.

11.

Skripta je pokrenula ssh pa je uspješno provedena autentikacija i uspostavljena veza.

**Vježba 4.7.**

1.

# h2xs -AX -n l201

2.

# cp monit_edu.pl monit_edu/lib/monit_edu.pm

3.

# cd monit_edu

# perl Makefile.PL

# make

# make install

4.

Nema razlike u ponašanju, samo u pozivu.

**Vježba 4.8.**

3.

# mkdir –p /home/linux1/paketi

# cd /home/linux1/paketi

# apt-source openssh-server

4.

# apt-get build-dep openssh-server

5.

# cd openssh-<verzija>

6.

7. Izradite binarnu verziju paketa.

# dpkg-buildpackage -rfakeroot -uc –b

8.

Uz openssh paket provode se i paketi o kojima taj paket ovisi.

9.

#apt-get purge openssh-server -y

10.

# dpkg -i openssh-server_6.7p1-5+deb8u3_i386.deb

11.

# systemctl status ssh

# ssh l201@127.0.0.1

12.

Permitrootlogin je postavljen na no. Dakle, uspješno je izmijenjen izvorni paket.

***Bilješke:***