

Modularizacija SquashFS Linux datotečnog sistema

by

Dajan Bračković

Submitted to the Odsjek za računarstvo i informatiku
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

ELEKTROTEHNICKI FAKULTET U SARAJEVU

June 2020

© Dajan Bračković, MMXX. All rights reserved.

The author hereby grants to ETF permission to reproduce and to
distribute publicly paper and electronic copies of this thesis document
in whole or in part in any medium now known or hereafter created.

Author

Odsjek za računarstvo i informatiku

May 18, 2020

Certified by

Samir Ribić

dr. sc.

Thesis Supervisor

Accepted by

Professor dr. sc. Jasmin Velagić

Dean of the Faculty of Electrical Engineering Sarajevo

Contents

Abstract	5
Uvod	6
Live distribucija?	6
Postupak kreiranja live USB	7
Upotrebe live distribucija	8
Datotečni sistemi	9
SquashFS datotečni sistem	9
Slax	10
Primjer aktiviranja/deaktiviranja Slax modula	10
NimbleX	11
Nedostaci Ubuntu-a	11
Zasto modifikovati instalacionu iso datoteku?	11
Sistemske zahtjevi	13
Priprema radnog okruženja	14
SquashFS paket	14
Proces kreiranja zasebnih modula	15
Modul NodeJS	15
Kreiranje direktorija potrebnih za rad	16
unsquashfs filesystem.squashfs datoteke	16
Konfiguracija paketa za chroot okruženje	17

Instalacija nodeJS paketa unutar chroot okruženja	18
Generisanje filesystem.squashfs direktorija	19
Generisanje Ubuntu .iso image sa nodeJS modulom	19
Pokretanje iso image-a pomoću kvm biblioteke	20
Rezultat Modul NodeJS	21
Modul MongoDB	23
Kreiranje direktorija potrebnih za rad	23
unsquashfs filesystem.squashfs datoteke	23
Konfiguracija paketa za chroot okruženje	24
Instalacija mongoDB paketa unutar chroot okruženja	25
Generisanje filesystem.squashfs direktorija	27
Generisanje Ubuntu .iso image sa mongoDB modulom	28
Pokretanje iso image-a pomoću kvm biblioteke	29
Rezultat Modul MongoDB	29
Modul Java	29
Kreiranje direktorija potrebnih za rad	30
unsquashfs filesystem.squashfs datoteke	30
Konfiguracija paketa za chroot okruženje	31
Instalacija Java paketa unutar chroot okruženja	32
Generisanje filesystem.squashfs direktorija	34
Generisanje Ubuntu .iso image sa Java modulom	34
Pokretanje iso image-a pomoću kvm biblioteke	35
Rezultat Modul Java	36
Modul Chrome	37
Kreiranje direktorija potrebnih za rad	37
unsquashfs filesystem.squashfs datoteke	38
Konfiguracija paketa za chroot okruženje	38
Instalacija Google Chrome paketa unutar chroot okruženja	39
Generisanje filesystem.squashfs direktorija	40
Generisanje Ubuntu .iso image sa Google Chrome modulom	41

Pokretanje iso image-a pomoću kvm biblioteke	42
Rezultat Modul Chrome	42

Abstract

This work is describing the process of modularization of Linux SquashFS filesystem. Modularization is performed by manual customization of packages in the live system, then extracting that customized system as a separate module.

This thesis will show how to create 3 separate modules out of the same base image, which will be the ubuntu-18.04.4-desktop-amd64.iso.

We will be using the SquashFS tools to make the modifications inside the ubuntu-18.04.4-desktop-amd64.iso image.

Uvod

Live distribucija?

Live distribucije omogućavaju korisniku da pokrene operativni sistem sa nekog eksternog uređaja kao što su CD/DVD, USB, HDD ili SSD (HDD i SSD se češće koriste kao eksterni diskovi na koje je u potpunosti instaliran operativni sistem), te da se operativni sistem učita u cjelosti u RAM memoriju. To omogućava upotrebu operativnog sistema bez potrebe da se isti instalira na interni disk unutar računara.

Najčešće se ipak koriste live CD/DVD ili USB distribucije.

Na ovaj način možemo pokrenuti instancu operativnog sistema na računar koji već ima prethodno instaliran operativni sistem, te nakon upotrebe ugasiti računar i izvaditi uređaj sa kog je pokrenut "živi" operativni sistem. Prilikom sljedećeg paljenja računara, biće pokrenut originalno instalirani operativni sistem.

Može se reći da su CD/DVD live distribucije sve manje i manje popularne zbog rasta upotrebe USB uređaja za pokretanje live distribucija.

Za razliku od live CD/DVD instalacija, podaci na USB uređaju mogu biti modificirani i dodatni podaci mogu biti upisani na uređaj. Korisnik može sa sobom u džepu nositi kompletan operativni sistem, aplikacije, konfiguracione datoteke i mnogo ličnih podataka.

Sa aspekta sigurnosti USB live distribucije imaju prednosti i mane. Svakako da je USB mnogo manji i stoga je lakše ga prenijeti i sakriti na neku sigurnu lokaciju pri čemu se onemogućuje drugima da pristupe vašim podacima. Međutim svakako da ga je lakše izgubiti, pa su šifriranje podataka i rezervna kopija mnogo važniji nego u slučaju tradicionalnog desktop operativnog sistema.

Treba imati na umu da će pokretanje live instalacija na starijim mašinama koje nemaju podršku za USB 2.0 protokol, biti jako sporo.

Neke mašine ne dozvoljavaju pokretanje tj. "bootanje" sa USB uređaja, te je potrebno "enableati" tu opciju unutar BOOT "managera" u BIOS-u.

Postupak kreiranja live USB

Postupak kreiranja live USB distribucije je sljedeći:

1. Uključiti USB uređaj u vaš računar
2. Preuzimanje instalacione datoteke, npr.

fedora 32 <https://getfedora.org/en/workstation/download/>

ili npr. ubuntu 20.04 <https://ubuntu.com/download/desktop>

3. Otvoriti Linux terminal, tj. command line interface.
4. Izvršiti komandu:

```
1 lsblk
```

Izlaz ove komande bi trebao ispisati sadržaj sličan sljedećem:

```
NAME                MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
loop0                7:0      0   93.9M  1 loop /snap/core/9066
loop1                7:1      0  160.7M  1 loop /snap/midori/451
loop2                7:2      0    97M   1 loop /snap/core/9289
sda                  8:0      0  111.8G  0 disk
├─sda1                8:1      0   243M  0 part /boot
├─sda2                8:2      0     1K  0 part
├─sda5                8:5      0  111.6G  0 part
│   └─sda5_crypt      254:0    0  111.6G  0 crypt
│       ├─debianjoda--vg-root
│       │   254:1      0    28G  0 lvm  /
│       └─debianjoda--vg-swap_1
│           254:2      0    7.9G  0 lvm  [SWAP]
└─debianjoda--vg-home
    254:3      0   75.7G  0 lvm  /home
sdb                  8:16     1   14.4G  0 disk
├─sdb1                8:17     1    3.4G  0 part /media/dejanqa/Ubuntu-Studio 20.04 LTS
├─sdb2                8:18     1    3.9M  0 part
├─sdb3                8:19     1   11.1G  0 part /media/dejanqa/writable
sr0                  11:0     1  1024M  0 rom
```

5. U ovom konkretnom slučaju uključen je USB uređaj i to se vidi na slici iznad.

Potrebno je izvršiti sljedeće komande:

```
1 sudo umount /dev/sdb1
```

```
2 sudo umount /dev/sdb3
```

Komandama iznad smo "dismountali" particije na USB uređaju koje imaju MOUNT-POINT.

6. Nakon ovog slijedi komanda:

```
1 sudo dd bs=4M if=path/home/user/downloads/ubuntu-20.04-amd64.iso of=/dev/sdb conv  
=fdatasync status=progress
```

Ovu komandu treba pustiti da se izvrši u potpunosti i podrazumijeva brisanje cijelog sadržaja na USB uređaju, pa korisnik treba biti svjestan te činjenice.

Nakon što se komanda dd završi uspješno dobili smo naš "live USB" :D.

7. Nakon ovog možemo pokrenuti ponovo računar (restart) te ući u boot meni i odabrati "bootanje" sa našeg uključenog USB uređaja na koji je instaliran iso image. Na većini računara za ulazak u boot meni je potrebno pritisnuti neku od F1->F12 tipki te odabrati opciju da računar boot-a sa USB-a.

8. Kada sistem boota treba odabrati opciju Try Ubuntu without installing, i nakon toga je live distribucija spremna za korištenje.

Upotrebe live distribucija

Live distribucije imaju razne upotrebe:

1. Za instalaciju operativnih sistema na HDD/SSD
2. Za popravku i spašavanje podataka sa računara
3. Za kompjutersku forenziku
4. Za izlistavanje i testiranje hardvera
5. Za sigurnosno testiranje mreže
6. Za promjenu, krađu, probijanje passworda
7. Za skeniranje i odstranjivanje virusa i malwarea
8. Za internet bankarstvo, kao namjenski konfigurisana platforma sa pojačanim sigurnosnim aspektima.

Datotečni sistemi

Datotečni sistemi (file systems ili filesystems) su skupina metoda i podatkovnih struktura koje operativni sistem koristi radi vođenja evidencije o podacima na disku ili particiji diska. Datotečni sistem određuje način na koji su organizovani podaci na disku.

Prije nego što disk ili particija diska mogu biti upotrebljeni kao datotečni sistem, datotečni sistem mora biti inicijaliziran na disk/particiju i strukture za čuvanje podataka trebaju biti upisane u disk. Ovaj proces se zove kreiranje datotečnog sistema. Prije svega treba napomenuti da su Linux datotečni sistemi su slični UNIX datotečnim sistemima. Većina UNIX datotečnih sistema ima sličnu strukturu i uređenost. Glavni koncepti su "superblock", "inode", "data block". "directory block" i "direction block".

"Superblock" sadrži informacije o datotečnom sistemu uopćenito, npr. veličina datotečnog sistema. "Inode" sadrži sve informacije o pojedinačnoj datoteci, izuzev njenog naziva. Naziv se smješta u direktorij zajedno sa rednim brojem "inodea".

Različiti operativni sistemi nemaju podršku za iste datotečne sisteme. Windows podržava NTFS, dok linux uglavnom EXT2, EXT3, EXT4. Mac s druge strane HFS+. FAT32 je stariji datotečni sistem razvijen od Microsoft-a, ali ga podržavaju i Linux i Windows operativni sistemi pa je nekad koristan za USB diskove koji se dijele između Windows-a i Linux-a.

SquashFS datotečni sistem

Squasfs je kompresovani "read-only" file system. Squasfs kompresuje datoteke, inodeove i direktorije. Podržava blokove veličina između 4KiB i 1MiB.

SquashFS datotečni sistem je napisan od strane Phillip Lougher-a i Robert Lougher-a. Nekoliko algoritama za kompresiju su podržani kao gzip, LZMA i LZO.

Squashfs se koristi u Live CD verzijama Arch Linux-a, Debian-a, Fedora-e, Gentoo Linux-a, HoleOS-a, Salix-a, Ubuntu-a, Clonezilla-e, "embedded/ugradbenim" distribucijama kao što su "OpenWrt" i DD-WRT ruter firmware. Koriste ga i Chromecast i Android Nugat. AppImage koristi squasfs za generisanje "appimages".

Slax

Slax je LiveCD Linux distribucija koju aktivno razvija Tomáš Matějček. Na oficijelnoj stranici Slax operativnog sistema stoji opis "Vaš džepni operativni sistem".

Jedna od prednosti Slax operativnog sistema je jednostavnost modifikacije. Paketi mogu biti dodani i uklonjeni koristeći Slax module, koji su kompresovani "read-only" squashfs datotečni sistemi kompresovani pomoću LZMA algoritma.

Paketi mogu biti instalirani upotrebom apt paket menadžera, zatim se spašava modul sa .sb ekstenzijom, te se generiše slax.iso sa tim novokreiranim modulom.sb pomoću genslaxiso komande.

Primjer aktiviranja/deaktiviranja Slax modula

Ispod su izlistane komande unutar Slax distribucije za instaliranje, aktiviranje, deaktiviranje modula sa firefox-esr(firefox-extended-support-release) paketom.

```
1 apt install firefox-esr
2 savechanges firefox.sb
3 genslaxiso slax.iso firefox.sb
4 slax activate firefox.sb
5 slax deactivate module.sb
```

NimbleX

NimbleX je Linux distribucija optimizirana da se pokrene sa CD-a, USB-a ili direktno preko mreže. Karakteristika NimbleX-a je brzo boot-anje, i mala količina prostora na disku koju zauzima, s obzirom da je distribuiran sa KDE desktop okruženjem. NimbleX je jedinstven po svojoj web stranici koja dozvoljava korisnicima da generišu boot-abilne distribucije modifikovane putem browsera. NimbleX boot-a u otprilike duplo kraćem vremenskom roku od Ubuntu ili recimo Fedora distribucije.

NimbleX je bio na strani kritika zbog nedostatka instaliranih sigurnosnih softverskih paketa.

Nedostaci Ubuntu-a

Ubuntu je baziran na SquashFS koji ima mogućnost podjele u više datoteka, ali se ipak ne može podijeliti na ovaj način jer instalacija aplikacija utiče na paketni menadžer. Dakle, moduli moraju biti napravljeni tako da njihova instalacija i de/instalacija ažurira popis paketa. Onda može da slijedi tehnika pravljenja pojedinih modula. Ja sam predložio da moduli idu redom Core, pa X.ORG, pa neki ne-grafički (kakve ste uradili), neko desktop okruženje i veća grafička aplikacija (tu ste uradili Chrome). Kako je vaš rad odlagan godinama, u međuvremenu je problem riješen, jer je nova verzija Slax Linuxa bazirana na Debianu, mada je nisam probao. Ali ipak ćemo rad dovesti do kraja.

Zasto modifikovati instalacionu iso datoteku?

Zasto uopce mijenjati instalacioni iso image operativnog sistema? Postoji nekoliko razloga:

1. Da bismo napravili svoju distribuciju mijenjajući postojeću iso datoteku
2. Da bismo predstavili određenu aplikaciju
3. Radi lokalizacije na određeni jezik

4. Da bismo uklonili odredjene softverske pakete
5. S ciljem dodavanja novih softverskih paketa
6. U svrhu azuriranja softverkih paketa
7. Radi mijenjanja sistemske konfiguracije kao sto su teme, ikone, fontovi, pozadina...

Najlaksi nacin modifikacije iso image-a baziranih na Ubuntu distribuciji je koriscenjem "Ubuntu Customization Kit" alata. Medjutim ovaj rad ce obuhvatiti drukciju princip, manualni.

Svaki od modula koji su kreirani su bazirani na istom base image-u, ubuntu-18.04.4-desktop-amd64.iso. Modifikacijom istog dobit cemo tri modula:

1. Modul NodeJS - ubuntu-with-nodejs-18.04-amd64.iso
2. Modul MongoDB - ubuntu-with-mognodb-18.04-amd64.iso
3. Modul Java - ubuntu-with-java-18.04-amd64.iso

Sistemske zahtjevi

Da biste se uputili u ovaj zadatak postoji prije svega nekoliko hardverskih minimuma koje vaša radna mašina treba da ispunjava:

1. Najmanje 20GB slobodnog prostora na disku, mada poželjno bi bilo i više od 20GB, pogotovo ukoliko pravite veći broj različitih modula.
2. Najmanje 2048MB RAM memorije i 4GB alocirane swap memorije.
3. Linux kernel sa squashfs podrškom.
4. QEMU/KVM || VirtualBox || VMWare - bilo koji od ova 3 alata za testiranje kreiranih modula.
5. genisoimage - paket za generisanje novog iso image-a

Priprema radnog okruzenja

Instalirati squashfs-tools i genisoimage:

```
1 sudo apt-get install squashfs-tools genisoimage
```

SquashFS paket

Paket squashfs-tools implementira 2 funkcije koje se koriste u ovom radu a koje pruža SquashFS (<http://tldp.org/HOWTO/SquashFS-HOWTO/whatis.html>). Radi se o funkcijama **mksquashfs** i **unsquashfs**. Prva od navedenih koristi se za kreiranje squashfs datoteke, dok se druga funkcija koristi za raspakivanje kompresovane squashfs datoteke.

SquashFS je moguće instalirati kao dodatak na linux jezgro. Prema tome moguće ga je instalirati na različite linux distribucije. Za Debian distribuciju njegov naziv je squashfs-tools.

Proces kreiranja zasebnih modula

Proces kreiranja zasebnih modula za svaki od modula je sličan, s tim da će se razlikovati nazivi direktorija, .iso datoteka i komande za instaliranje modula. Neki paketi se instaliraju direktno jednom komandom, dok je za neke druge potrebno vršiti dodatne konfiguracije.

Modul NodeJS

NodeJS je kros-platforma "otvorenog koda" za izvršavanje JavaScript koda izvan okruženja web pretraživača. Nešto slično kao što su u Java svijetu JVM(JavaVirtualMachine) i JRE(JavaRuntimeEnvironment) i JDK(JavaDevelopmentKit) zajedno. Može se reći da NodeJS objedinjuje sve te stvari u jedan koncept, ali za JavaScript programski jezik.

NodeJS ima "event-driven" baziranu arhitekturu koja omogućava asinhroni input/output. To ga čini pogodnim za razvijanje web aplikacija koje imaju mnošto input/output operacija, kao i za razvoj "real-time" web aplikacija i browser igrice.

Prirodno NodeJS podržava samo JavaScript programski kod. Međutim podržava i programske jezike koji se daju kompajlirati u JS. To su CoffeeScript, Dart, TypeScript, ClojureScript.

NodeJS se najčešće koristi za razvijanje WebServer-a. Jedna od bitnih razlika između NodeJS-a i PHP-a je ta što kod PHP-a većina funkcija blokira izvršavanje naredne funkcije dok se ne izvrši u potpunosti, dok su funkcije u NodeJS-u neblokirajuće te se mogu izvršavati paralelno, te na osnovu "callback" funkcije signaliziraju uspješno izvršenje ili error.

NodeJS oficijelno podržavaju i Linux i MacOS i Windows OS-i.

NodeJS je napravljen na osnovu Google V8 JavaScript engine-a, koji prvenstveno napravljen za upotrebu u Google Chrome i Chromium pretraživačima.

Kreiranje direktorija potrebnih za rad

NodeJS Modul će biti kreiran od istog baznog modula kao i svi ostali moduli. To je ubuntu-18.04.4-desktop-amd64.iso datoteka:

```
1 mkdir ~/squashfs/livedtmp
2 mkdir ~/squashfs/livedtmp/isoimgs
3 mv ~/Downloads/ubuntu-18.04.4-desktop-amd64.iso ~/squashfs/livedtmp/isoimgs
4 cd ~/squashfs/livedtmp
```

Napraviti mnt direktorij unutar livedtmp direktorija u koji će biti mount-an ubuntu-18.04.4-desktop-amd64.iso image:

```
1 mkdir mnt
2 sudo mount -o loop ./isoimgs/ubuntu-18.04.4-desktop-amd64.iso mnt
```

Napraviti direktorij extract-cd u kojeg ćemo kopirati mnt direktorij izostavljajući filesystem.squashfs datoteku unutar /casper direktorija:

```
1 mkdir extract-cd
2 sudo rsync --exclude=/casper/filesystem.squashfs -a mnt/ extract-cd
```

Napraviti direktorij za modul nodejs i kopirati u njega extract-cd direktorij:

```
1 mkdir modul-nodejs
2 sudo rsync -a extract-cd/ modul-nodejs
```

unsquashfs filesystem.squashs datoteke

U ovom trenutku ćemo upotrijebiti unsquashfs funkciju iz squashfs-tools paketa. Te ćemo kopirati raspakovani squashfs-root direktorij u edit direktorij. Ovaj edit direktorij ćemo kasnije koristiti da unutar njega instaliramo nodejs pakete:

```
1 sudo unsquashfs mnt/casper/filesystem.squashfs
2 sudo mv squashfs—root/ edit
```

Konfiguracija paketa za chroot okruženje

Da bi imali mrežnu konekciju unutar edit direktorija jedno rjesenje je kopirati /run direktorij unutar edit direktorija. Najbolje manuelno popuniti resolv.conf unutar edit direktorija:

```
1 sudo gedit edit/etc/resolv.conf
```

Te unijeti sljedeći sadržaj i spasiti promjene: (*nameserver 1.1.1.1*
nameserver 8.8.8.8).

Isto vazi i za etc/hosts datoteku:

```
1 sudo gedit edit/etc/hosts
```

Kopirati sadržaj iz /etc/hosts datoteke na sistemu domaćinu unutar edit/etc/hosts datoteke:

```
127.0.0.1          localhost
127.0.1.1          debianjoda.joda.net      debianjoda
```

Namjestiti edit/dev direktorij kopirajući /dev/ direktorij sa hosta, zatim chroot u edit direktorij. Obaviti mount instrukcije navedene ispod. Ukoliko korisnik odluči da obrisu edit direktorij iz nekog razloga, bilo bi potrebno uraditi unmount edit direktorija da sistem domaćin ne bi postao neupotrebljiv:

```
1 sudo mount --bind /dev/ edit/dev
2 sudo chroot edit
3 mount -t proc none /proc
4 mount -t sysfs none /sys
5 mount -t devpts none /dev/pts
```

Također potrebno je izvršiti sljedeće komande da bi se izbjegli problemi sa lokalizacijom:

```
1 export HOME=/root
2 export LC_ALL=C
```

Instalacija nodeJS paketa unutar chroot okruženja

Za ispis svih instaliranih paketa:

```
1 dpkg-query -W --showformat='${Installed-Size}\t${Package}\n' | sort -nr | less
```

Instalacija nodejs paketa:

```
1 apt-get update
2 apt-get install curl
3 curl -sL https://deb.nodesource.com/setup_13.x | sudo -E bash -
4 apt-get install -y nodejs
```

Nakon završetka instalacije izvršiti unutar chroot:

```
1 apt-get clean
2 rm -rf /tmp/* ~/.bash_history
3 rm -rf /tmp/* ~/.bashrc
4 rm /var/lib/dbus/machine-id
5 rm /sbin/initctl
6 dpkg-divert --rename --remove /sbin/initctl
7 umount /proc || umount -lf /proc
8 umount /sys
9 umount /dev/pts
10 umount /dev
11 exit
```

Ponovno generisati filesystem.manifest:

```
1 chmod +w extract-cd/casper/filesystem.manifest
2 sudo su
3 chroot edit dpkg-query -W --showformat='${Package} ${Version}\n' > extract-cd/
  casper/filesystem.manifest
```

```
4 exit
5 sudo cp extract—cd/casper/filesystem.manifest extract—cd/casper/filesystem.manifest—
  desktop
6 sudo sed -i '/ubiquity/d' extract—cd/casper/filesystem.manifest—desktop
7 sudo sed -i '/casper/d' extract—cd/casper/filesystem.manifest—desktop
```

Generisanje filesystem.squashfs direktorija

Sada ćemo upotrijebiti drugu funkciju iz squashfs-tools, a to je mksquashfs. S tom funkcijom ćemo kompresovati edit direktorij u novu filesystem.squashfs datoteku. U kodu ispod je potrebno izvršiti komandu iz linije 1 i jednu od preostale 3, pri čemu prva (komanda na liniji 2) daje najslabiju kompresiju, ali je najbrža. Druga komanda se duže izvršava ali je veći procenat kompresije u odnosu na prvu komandu. Dok je kod treće komande procenat kompresije najveći, a vrijeme izvršenja najduže:

```
1 sudo rm extract—cd/casper/filesystem.squashfs
2 sudo mksquashfs edit extract—cd/casper/filesystem.squashfs —nozma
3 sudo mksquashfs edit extract—cd/casper/filesystem.squashfs —b 1048576
4 sudo mksquashfs edit extract—cd/casper/filesystem.squashfs —comp xz —e edit/boot
```

Naredni korak je da azuriramo filesystem.size datoteku:

```
1 sudo su
2 printf $(du -sx --block-size=1 edit | cut -f1) > extract—cd/casper/filesystem.size
3 exit
```

Generisanje Ubuntu .iso image sa nodeJS modulom

Nakon toga upisati naziv image-a unutar README.diskdefines. Upisati 'Ubuntu with NodeJS 18.04.4 LTS "Bionic Beaver" - Release amd64' u polje DISKNAME:

```
1 cd extract—cd
2 sudo rm md5sum.txt
```

```
3 find -type f -print0 | sudo xargs -0 md5sum | grep -v isolinux/boot.cat | sudo tee md5sum  
    .txt
```

Azurirati md5sum.txt datoteku:

```
1 sudo gedit extract-cd/README.diskdefines
```

Napokon mozemo napraviti iso image koji ce da sadrzi NodeJS modul. Za ovu operaciju koristimo funkciju genisoimage. Neke linux distribucije nude mkisofs funkciju. Tako da ukoliko ne radi jedna trebala bi druga:

```
1 sudo genisoimage -D -r -V "$IMAGE_NAME" -cache-inodes -J -l -b isolinux/isolinux.  
    bin -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table -o ../  
    ubuntu-with-nodejs-18.04-amd64.iso .
```

Pokretanje iso image-a pomoću kvm biblioteke

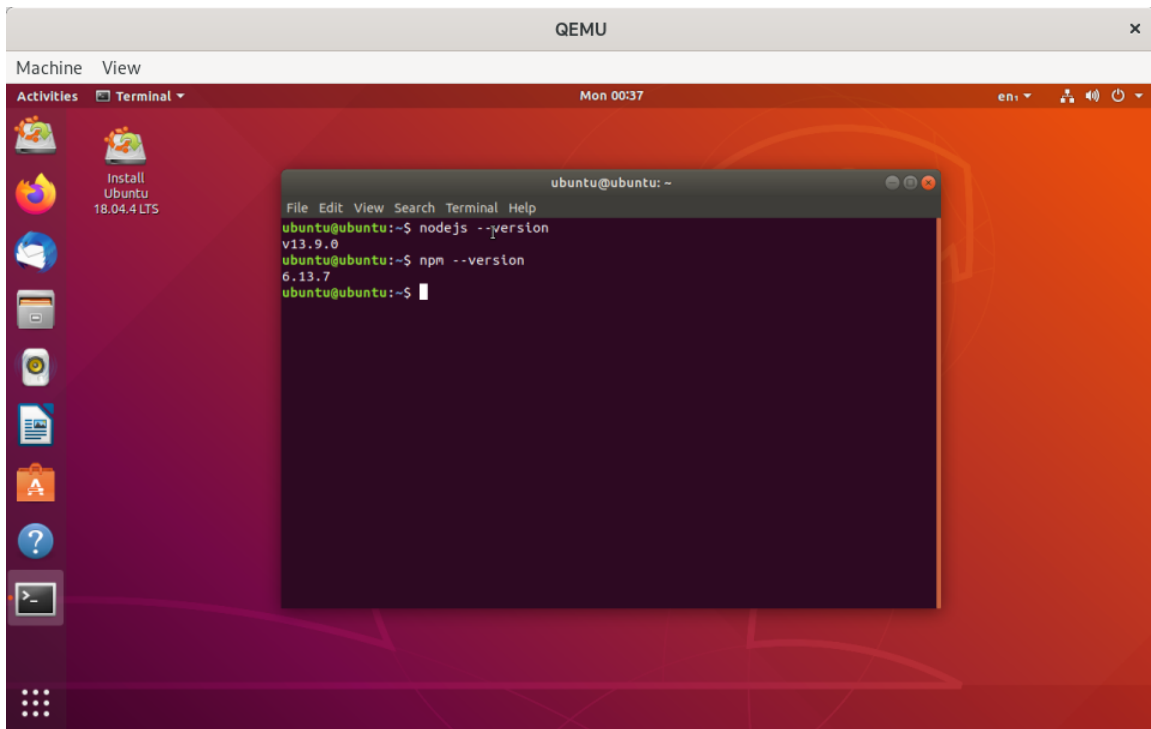
Sada cemo napraviti virtuelni hard disk pomocu qemu-img komande da bismo pokrenuli na njemu nas novi modul NodeJS Ubuntu.

```
1 cd ~  
2 qemu-img create ubuntunodejs.img 5G
```

Pokrenucemo modul pomocu KVM-a:

```
1 sudo kvm -hda ubuntunodejs.img -cdrom ~/zavrzni/livecdtmp/ubuntu-with-nodejs  
    -18.04-amd64.iso -boot d -m 2048
```

Rezultat Modul NodeJS



Unutar ove live instalacije mozemo upotrijebiti nodeJS biblioteku te kreirati jednostavnu web aplikaciju.

Prateci uputstvo na linku:

<https://docs.microsoft.com/en-us/azure/app-service/app-service-web-get-started-nodejs>

unutar nase live distribucije sa preinstaliranim NodeJS bibliotekama izvorsimo sljedece komande koristeći Terminal:

-
- ```
1 git clone https://github.com/Azure-Samples/nodejs-docs-hello-world
2 cd nodejs-docs-hello-world
3 npm start
```
- 

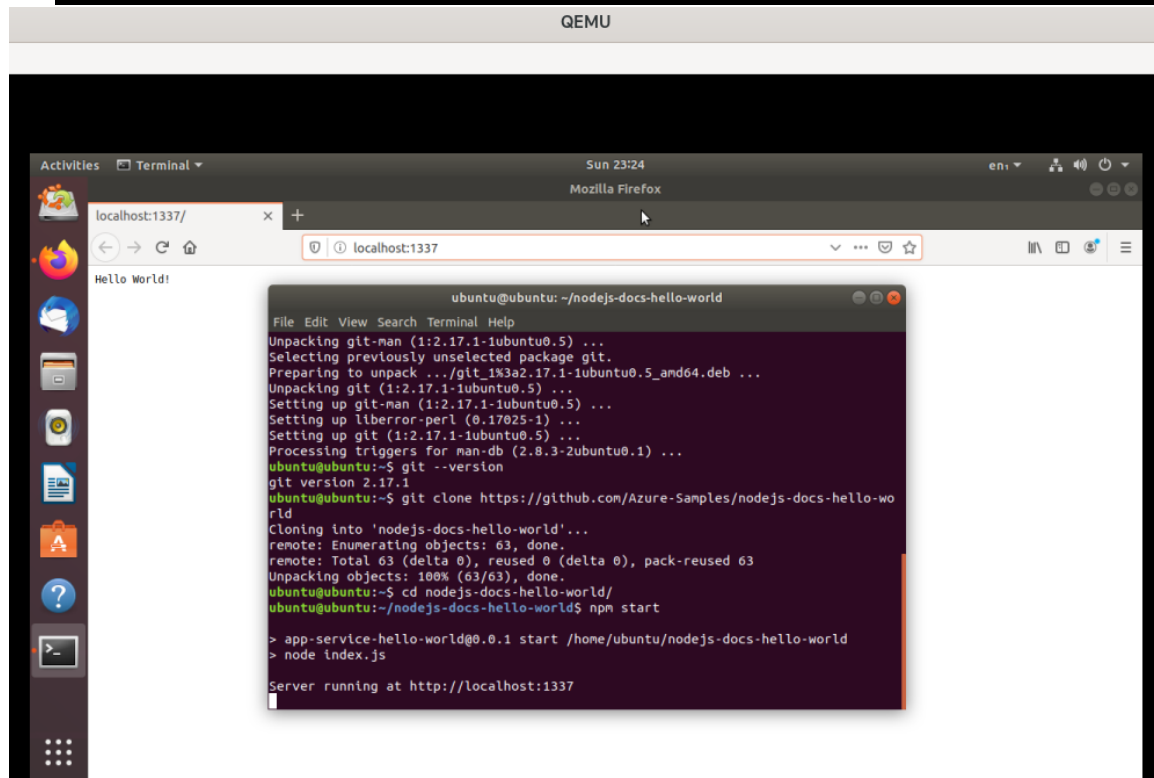
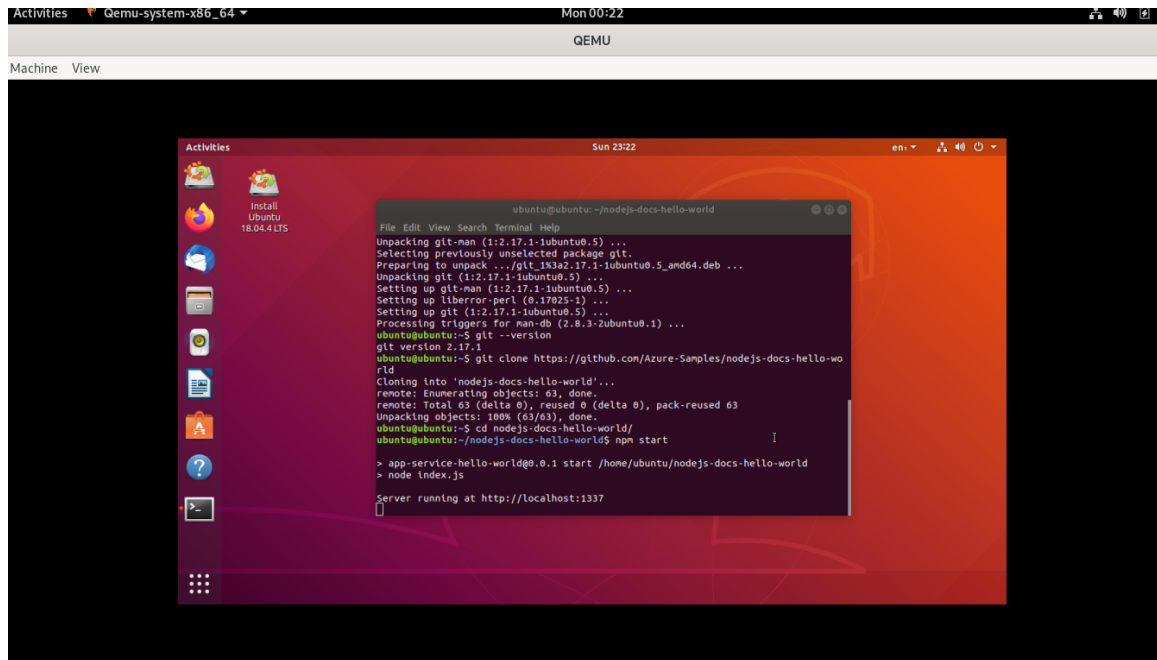
Ukoliko git program nije instaliran potrebno je instalirati git koristeći komandu:

- 
- ```
1 sudo apt install git
```
-

Nakon toga NodeJS bi trebao pokrenuti server kojeg možemo provjeriti u web pregledniku na URL-u:

-
- ```
1 http://localhost:1337
```
-

Za potrebe rada nije rađena modifikacija ove web aplikacije, ali moguće je iskoristiti aplikaciju kao bazu za nadograđivanje po želji. HTTP web server se kreira unutar index.js datoteke te bi početna modifikacija bila svakako nadogradnja ove datoteke sa dodatnim funkcionalnostima.



# Modul MongoDB

MongoDB je nerelaciona baza podataka napisana u C++ programskom jeziku. Koristi JSON format za spremanje podataka.

To je čini pogodnom za povezivanje sa NodeJS bibliotekama, čiji smo modul već napravili u prethodnom paragrafu.

Postupak kreiranja ovog modula će biti gotovo identičan postupku kreiranja NodeJS modula, izuzev dijela u kojem se vrši instaliranje novih paketa unutar raspakovanog squashfs datotečnog sistema.

## Kreiranje direktorija potrebnih za rad

---

```
1 cd ~/squashfs/livecdtmp
2 sudo mount -o loop ./isoimgs/ubuntu-18.04.4-desktop-amd64.iso mnt
3 mkdir extract-mongodb-cd
4 sudo rsync --exclude=/casper/filesystem.squashfs -a mnt/ extract-mongodb-cd
5 mkdir modul-mongodb
6 sudo rsync -a extract-mongodb-cd/ modul-mongodb
```

---

## unsquashfs filesystem.squashfs datoteke

Zatim slijedi korak u kojem se opet raspakuje filesystem.squashfs direktorij. Ova operacija može potrajati par minuta tako da je ne treba prekidati:

---

```
1 sudo unsquashfs mnt/casper/filesystem.squashfs
```

---

Te prekopiramo sadržaj novonastalog squashfs-root direktorija u edit-mongodb direktorij:

---

```
1 sudo mv squashfs-root/ edit-mongodb
```

---

## Konfiguracija paketa za chroot okruženje

Da bi imali mrežnu konekciju unutar edit-mongodb direktorija jedno rjesenje je kopirati /run direktorij unutar edit-mongodb direktorija. Najbolje manuelno popuniti resolv.conf unutar edit direktorija:

---

```
1 sudo gedit edit-mongodb/etc/resolv.conf
```

---

Te unijeti sljedeći sadržaj i spasiti promjene: (*nameserver 1.1.1.1*  
*nameserver 8.8.8.8*).

Isto važi i za etc/hosts datoteku:

---

```
1 sudo gedit edit-mongodb/etc/hosts
```

---

Kopirati sadržaj iz /etc/hosts datoteke na sistemu domaćinu unutar edit-mongodb/etc/hosts datoteke:

```
127.0.0.1 localhost
127.0.1.1 debianjoda.joda.net debianjoda
```

Namjestiti edit-mongodb/dev direktorij kopirajući /dev/ direktorij sa hosta, zatim chroot u edit-mongodb direktorij. Obaviti mount instrukcije navedene ispod. Ukoliko korisnik odluči da obrisu edit-mongodb direktorij iz nekog razloga, bilo bi potrebno uraditi unmount edit-mongodb direktorija da sistem domaćin ne bi postao neupotrebljiv:

---

```
1 sudo mount --bind /dev/ edit-mongodb/dev
2 sudo chroot edit-mongodb
3 mount -t proc none /proc
4 mount -t sysfs none /sys
5 mount -t devpts none /dev/pts
```

---

Neophodno je podesiti sistemske varijable pomoću sljedeće komande da bi se izbjegli problemi sa lokalizacijom:

---

```
1 export HOME=/root
2 export LC_ALL=C
```

---



## Instalacija mongoDB paketa unutar chroot okruženja

Za ispis svih instaliranih paketa:

---

```
1 dpkg-query -W --showformat='${Installed-Size}\t${Package}\n' | sort -nr | less
```

---

Naime da bismo mogli pokrenuti mongoDB, neophodno je instalirati libcurl4 i openssl pakete:

---

```
1 sudo apt-get install libcurl4 openssl
```

---

Preuzimanje mongodb paketa sa interneta:

---

```
1 wget https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-ubuntu1804-4.2.5.tgz
```

---

Ekstrakcija paketa:

---

```
1 tar -zxvf mongodb-linux-x86_64-ubuntu1804-4.2.5.tgz
```

---

Da bismo izbjegli potrebu da postavimo putanju u PATH sistemsku varijablu, kopiraćemo mongodb bin direktorij u /usr/local/bin/ direktorij:

---

```
1 sudo cp mongodb-linux-x86_64-ubuntu1804-4.2.5/bin/* /usr/local/bin/
```

---

Konfiguracija mongodb paketa:

Prvo napravimo direktorij u koji će mongodb spremati podatke:

---

```
1 sudo mkdir -p /var/lib/mongo
```

---

Također potrebno je napraviti direktorij u koji će se spremati logovi:

---

```
1 sudo mkdir -p /var/log/mongodb
```

---

Potrebno je ažurirati privilegije pristupa na novokreirane direktorije:

---

```
1 chown 'whoami' /var/lib/mongo
2 chown 'whoami' /var/log/mongodb
```

---

Sada možemo pokrenuti mongod proces:

---

```
1 mongod --dbpath /var/lib/mongo --logpath /var/log/mongodb/mongod.log --fork
```

---

Provjera instalacije:

---

```
1 mongo --version
```

---

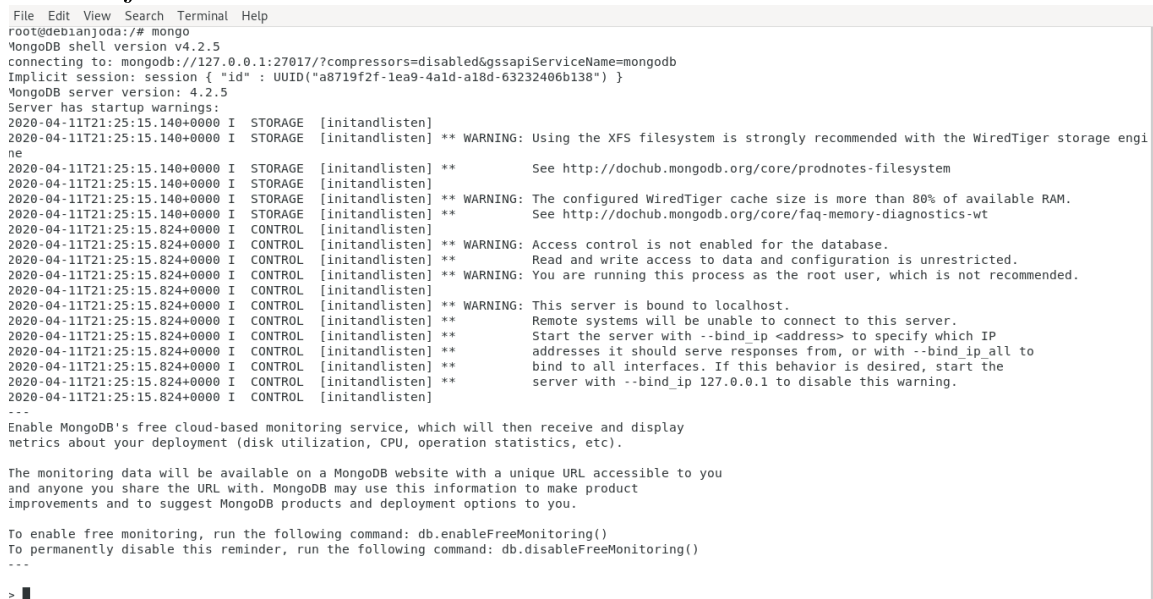
Rezultat komande bi trebao potvrditi uspješno instaliran mongodb:

---

```
1 MongoDB shell version v4.2.5
2 git version: 2261279b51ea13df08ae708ff278f0679c59dc32
3 OpenSSL version: OpenSSL 1.1.1 11 Sep 2018
4 allocator: tcmalloc
5 modules: none
6 build environment:
7 distmod: ubuntu1804
8 distarch: x86_64
9 target_arch: x86_64
```

---

Na slici ispod je prikazan mongo pokrenut u konzoli unutar chroot edit-mongodb direktorija:



```
File Edit View Search Terminal Help
root@debianjoda:/# mongo
MongoDB shell version v4.2.5
connecting to: mongod://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongod
Implicit session: session { "id" : UUID("a8719f2f-1ea9-4a1d-a18d-63232406b138") }
MongoDB server version: 4.2.5
Server has startup warnings:
2020-04-11T21:25:15.140+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2020-04-11T21:25:15.140+0000 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2020-04-11T21:25:15.140+0000 I STORAGE [initandlisten] ** WARNING: The configured WiredTiger cache size is more than 80% of available RAM.
2020-04-11T21:25:15.140+0000 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/faq-memory-diagnostics-wt
2020-04-11T21:25:15.824+0000 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-04-11T21:25:15.824+0000 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2020-04-11T21:25:15.824+0000 I CONTROL [initandlisten] ** WARNING: You are running this process as the root user, which is not recommended.
2020-04-11T21:25:15.824+0000 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2020-04-11T21:25:15.824+0000 I CONTROL [initandlisten] ** Remote systems will be unable to connect to this server.
2020-04-11T21:25:15.824+0000 I CONTROL [initandlisten] ** Start the server with --bind ip <address> to specify which IP
2020-04-11T21:25:15.824+0000 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --bind ip all to
2020-04-11T21:25:15.824+0000 I CONTROL [initandlisten] ** bind to all interfaces. If this behavior is desired, start the
2020-04-11T21:25:15.824+0000 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.
2020-04-11T21:25:15.824+0000 I CONTROL [initandlisten]

Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()

```

Bilo bi poželjno mongoDB pokrenuti povezujući je sa drugom IP adresom jer je po defaultu povezana na localhost tj. 127.0.0.1 te može primati zahtjeve samo od aplikacija koje su na toj mašini na kojoj je instaliran mongo.

---

```
1 Start the server with --bind_ip <address> to specify which IP addresses it should serve
 responses from, or with --bind_ip_all to bind to all interfaces.
```

---

Nakon završetka instalacije izvršiti unutar chroot "čišćenje":

---

```
1 apt-get clean
2 rm -rf /tmp/* ~/.bash_history
3 rm -rf /tmp/* ~/.bashrc
4 rm /var/lib/dbus/machine-id
5 rm /sbin/initctl
6 dpkg-divert --rename --remove /sbin/initctl
7 umount /proc || umount -lf /proc
8 umount /sys
9 umount /dev/pts
10 umount /dev
11 exit
```

---

Ponovno generisati filesystem.manifest:

---

```
1 sudo chmod +w extract-mongodb-cd/casper/filesystem.manifest
2 sudo su
3 chroot edit-mongodb dpkg-query -W --showformat='${Package} ${Version}\n' >
 extract-mongodb-cd/casper/filesystem.manifest
4 exit
5 sudo cp extract-mongodb-cd/casper/filesystem.manifest extract-mongodb-cd/casper/
 filesystem.manifest-desktop
6 sudo sed -i '/ubiquity/d' extract-mongodb-cd/casper/filesystem.manifest-desktop
7 sudo sed -i '/casper/d' extract-mongodb-cd/casper/filesystem.manifest-desktop
```

---

## Generisanje filesystem.squashfs direktorija

Sada ćemo upotrijebiti drugu funkciju iz squashfs-tools, a to je mksquashfs. S tom funkcijom ćemo kompresovati edit-mongodb direktorij u novu filesystem.squashfs datoteku. U kodu ispod je potrebno izvršiti komandu iz linije 1 i jednu od preostale 3, pri čemu prva (komanda na liniji 2) daje najslabiju kompresiju, ali je najbrža. Druga komanda se duže izvršava ali je veći procenat kompresije u odnosu na prvu komandu. Dok je kod treće komande procenat kompresije najveći, a vrijeme izvršenja najduže:

---

```
1 sudo rm extract-mongodb-cd/casper/filesystem.squashfs
2 sudo mksquashfs edit-mongodb extract-mongodb-cd/casper/filesystem.squashfs -no-lzma
3 sudo mksquashfs edit-mongodb extract-mongodb-cd/casper/filesystem.squashfs -b
 1048576
4 sudo mksquashfs edit-mongodb extract-mongodb-cd/casper/filesystem.squashfs -comp xz
 -e edit-mongodb/boot
```

---

Naredni korak je da azuriramo filesystem.size datoteku:

---

```
1 sudo su
2 printf $(du -sx --block-size=1 edit-mongodb | cut -f1) > extract-mongodb-cd/casper
 /filesystem.size
3 exit
```

---

Nakon toga upisati naziv image-a unutar README.diskdefines. Upisati 'Ubuntu with MONGODB 18.04.4 LTS "Bionic Beaver" - Release amd64' u polje DISKNAME:

---

```
1 sudo gedit extract-mongodb-cd/README.diskdefines
```

---

## Generisanje Ubuntu .iso image sa mongoDB modulom

Ažurirati md5sum.txt datoteku:

---

```
1 cd extract-mongodb-cd
2 sudo rm md5sum.txt
3 find -type f -print0 | sudo xargs -0 md5sum | grep -v isolinux/boot.cat | sudo tee md5sum
 .txt
```

---

Napokon možemo napraviti iso image koji će da sadrži MongoDB modul. Za ovu operaciju koristimo funkciju genisoimage. Neke linux distribucije nude mkisofs funkciju. Tako da ukoliko ne radi genisoimage trebala bi raditi funkcija mkisofs:

---

```
1 sudo genisoimage -D -r -V "$IMAGE_NAME" -cache-inodes -J -l -b isolinux/isolinux.
 bin -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table -o ../
 ubuntu-with-mongodb-18.04-amd64.iso .
```

---

## Pokretanje iso image-a pomoću kvm biblioteke

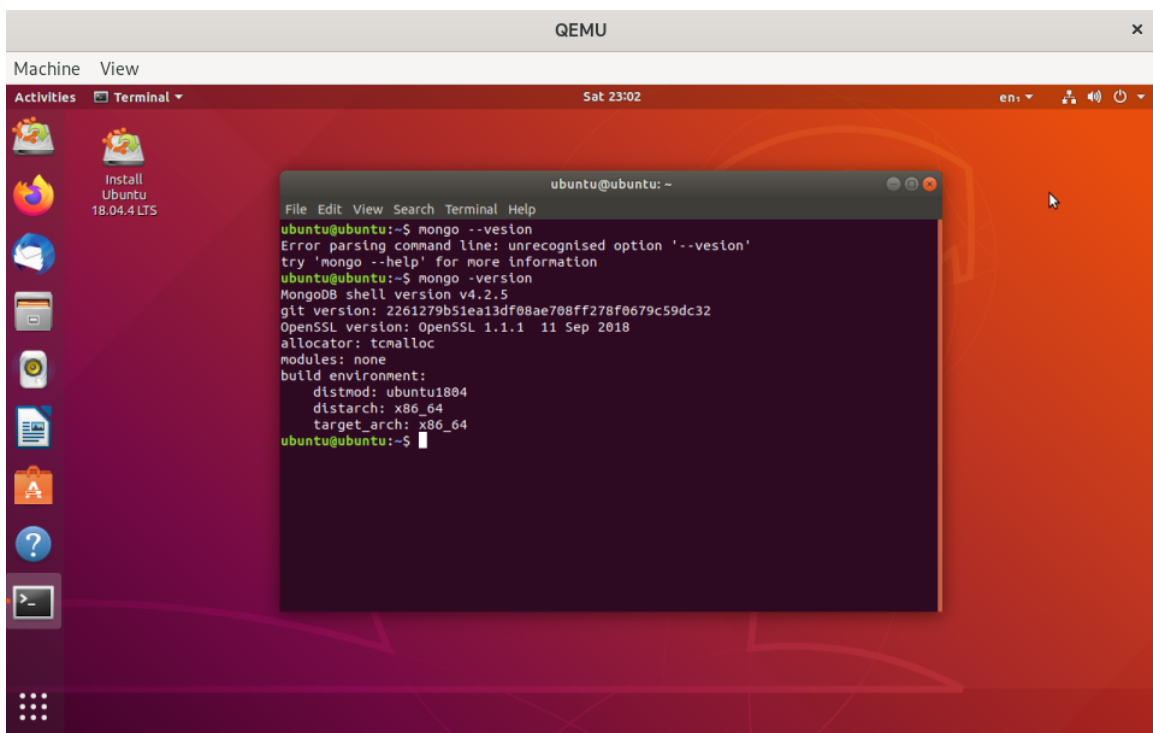
Sada ćemo napraviti virtuelni hard disk pomoću qemu-img komande da bismo pokrenuli na njemu naš novi modul mongoDB Ubuntu.

- ```
1 cd ~  
2 qemu-img create ubuntumongodb.img 5G
```

Pokrenućemo modul pomoću KVM-a:

- ```
1 sudo kvm -hda ubuntumongodb.img -cdrom ~/zavrzni/livedtmp/ubuntu-with-mongodb
-18.04-amd64.iso -boot d -m 2048
```

## Rezultat Modul MongoDB



# Modul Java

Java je objektno-orijentisani programski jezik opće namjene. Aplikacije napisane u Java programskom jeziku se prije izvršenja kompajliraju u Java bytecode te se izvršavaju u Java virtuelnoj mašini, tako da jedan java program može da se pokrene na bilo kom računaru koji ima instaliran Java Virtual Machine.

Java je napravljena od strane Sun Microsystems kompanije te objavljena u maju 1995. godine. Posljednja verzija Java-e je 14, dok je posljednja LTS verzija 11. Postoji nekoliko različitih platformi za koje se distribuira Java, tako da iz toga se mogu definisati 4 grupe Java distribucija:

1. Java Card - za kartice
2. Java Platform, Micro Edition (ME) - za računare sa ograničenim resursima
3. Java Platform, Standard Edition (SE) - za takozvane radne stanice ili "workstations"
4. Java Platform, Enterprise Edition (EE) - za velike distribuirane poslovne sisteme i internet okruženja

Slijedi postupak kreiranja Java modula.

## Kreiranje direktorija potrebnih za rad

---

```
1 cd ~/squashfs/livecdtmp
2 sudo mount -o loop ./isoimgs/ubuntu-18.04.4-desktop-amd64.iso mnt
3 mkdir extract-java-cd
4 sudo rsync --exclude=/casper/filesystem.squashfs -a mnt/ extract-java-cd
5 mkdir modul-java
6 sudo rsync -a extract-java-cd/ modul-java
```

---

## unsquashfs filesystem.squashfs datoteke

Zatim slijedi korak u kojem se opet raspakuje filesystem.squashfs direktorij i kopiramo ga u edit-java direktorij. Ovaj put ćemo edit direktorij imenovati edit-java da

ne izgubimo prethodni sadržaj edit direktorija.

---

```
1 sudo unsquashfs mnt/casper/filesystem.squashfs
2 sudo mv squashfs-root/ edit-java
```

---

Na slici ispod se vidi output unsquashfs komande:

```
dejanqa@debianjoda:~/zavrzni/livedttmp$ sudo unsquashfs mnt/casper/filesystem.squashfs
Parallel unsquashfs: Using 4 processors
141272 inodes (160595 blocks) to write

[=====] 160595/160595 100%

created 113789 files
created 17309 directories
created 27449 symlinks
created 7 devices
created 0 fifos
```

## Konfiguracija paketa za chroot okruženje

Da bi imali mrežnu konekciju unutar edit-java direktorija jedno rješenje je kopirati /run direktorij unutar edit-java direktorija. Najbolje manuelno popuniti resolv.conf unutar edit-java direktorija:

---

```
1 sudo gedit edit-java/etc/resolv.conf
```

---

Te unijeti sljedeći sadržaj i spasiti promjene: (*nameserver 1.1.1.1*  
*nameserver 8.8.8.8*).

Isto važi i za etc/hosts datoteku:

---

```
1 sudo gedit edit-java/etc/hosts
```

---

Kopirati sadržaj iz /etc/hosts datoteke na sistemu domacinu unutar edit-java/etc/hosts datoteke:

```
127.0.0.1 localhost
127.0.1.1 debianjoda.joda.net debianjoda
```

Namjestiti edit-java/dev direktorij kopirajući /dev/ direktorij sa hosta, zatim chroot u edit-java direktorij. Obaviti mount instrukcije navedene ispod. Ukoliko korisnik odluči da obriše edit-java direktorij iz nekog razloga, bilo bi potrebno uraditi unmount edit-java direktorija da sistem domaćin ne bi postao neupotrebljiv:

---

```
1 sudo mount --bind /dev/ edit-java/dev
2 sudo chroot edit-java
3 mount -t proc none /proc
4 mount -t sysfs none /sys
5 mount -t devpts none /dev/pts
```

---

Takodjer potrebno je izvršiti sljedeće komande da bi se izbjegli problemi sa lokalizacijom:

---

```
1 export HOME=/root
2 export LC_ALL=C
```

---

## Instalacija Java paketa unutar chroot okruženja

Za ispis svih instaliranih paketa:

---

```
1 dpkg-query -W --showformat='${Installed-Size}\t${Package}\n' | sort -nr | less
```

---

Instalacija java paketa:

---

```
1 apt update
2 apt install default-jdk
```

---

Provjera verzije java instalacije:

---

```
1 java -version
```

---

Rezultat prethodne komande bi trebao biti:

---

```
1 openjdk version "11.0.6" 2020-01-14
2 OpenJDK Runtime Environment (build 11.0.6+10-post-Ubuntu-1ubuntu118.04.1)
3 OpenJDK 64-Bit Server VM (build 11.0.6+10-post-Ubuntu-1ubuntu118.04.1, mixed mode
 , sharing)
```

---

Sada možemo instalirati Eclipse, koji je jedan od najpoznatijih JAVA IDE (Integrated Development Environment). Prvo ćemo preuzeti eclipse.tgz sa interneta pomoću wget komande, a zatim instalirati:



---

```
1 wget http://ftp.jaist.ac.jp/pub/eclipse/technology/epp/downloads/release/2019-03/R/eclipse
 -java-2019-03-R-linux-gtk-x86_64.tar.gz
2 tar -zxvf eclipse-java-2019-*--R-linux-gtk-x86_64.tar.gz -C /usr/
3 ln -s /usr/eclipse/eclipse /usr/bin/eclipse
4 nano /usr/share/applications/eclipse.desktop
```

---

Nakon posljednje komande unijeti sljedeći sadržaj:

---

```
1 [Desktop Entry]
2 Encoding=UTF-8
3 Name=Eclipse IDE
4 Comment=Eclipse IDE
5 Exec=/usr/bin/eclipse
6 Icon=/usr/eclipse/icon.xpm
7 Terminal=false
8 Type=Application
9 StartupNotify=false
```

---

Sada možemo pokrenuti eclipse međjutim to ćemo kasnije uraditi kada pokrenemo iso file u qemu-kvm. Nakon završetka instalacije izvršiti unutar chroot:

---

```
1 apt clean
2 rm -rf /tmp/* ~/.bash_history
3 rm -rf /tmp/* ~/.bashrc
4 rm /var/lib/dbus/machine-id
5 rm /sbin/initctl
6 dpkg-divert --rename --remove /sbin/initctl
7 umount /proc || umount -lf /proc
8 umount /sys
9 umount /dev/pts
10 umount /dev
11 exit
```

---

Ponovno generisati filesystem.manifest:

---

```
1 sudo chmod +w extract-java-cd/casper/filesystem.manifest
2 sudo su
3 chroot edit-java dpkg-query -W --showformat='${Package} ${Version}\n' > extract-
 java-cd/casper/filesystem.manifest
4 exit
5 sudo cp extract-java-cd/casper/filesystem.manifest extract-java-cd/casper/filesystem.
 manifest-desktop
6 sudo sed -i '/ubiquity/d' extract-java-cd/casper/filesystem.manifest-desktop
7 sudo sed -i '/casper/d' extract-java-cd/casper/filesystem.manifest-desktop
```

---

## Generisanje filesystem.squashfs direktorija

Sada ćemo upotrijebiti drugu funkciju iz squashfs-tools, a to je mksquashfs. S tom funkcijom ćemo kompresovati edit-java direktorij u novu filesystem.squashfs datoteku, baš kao i u prethodna dva slučaja sa MongoDB i NodeJS modulima. U kodu ispod je potrebno izvršiti komandu iz linije 1 i jednu od preostale 3, pri čemu prva (komanda na liniji 2) daje najslabiju kompresiju, ali je najbrža. Druga komanda se duže izvršava ali je veći procenat kompresije u odnosu na prvu komandu. Dok je kod treće komande procenat kompresije najveći, a vrijeme izvršenja najduže:

---

```
1 sudo rm extract-java-cd/casper/filesystem.squashfs
2 sudo mksquashfs edit-java extract-java-cd/casper/filesystem.squashfs -nolzma
3 sudo mksquashfs edit-java extract-java-cd/casper/filesystem.squashfs -b 1048576
4 sudo mksquashfs edit-java extract-java-cd/casper/filesystem.squashfs -comp xz -e edit/
 boot
```

---

Naredni korak je da ažuriramo filesystem.size datoteku:

---

```
1 sudo su
2 printf $(du -sx --block-size=1 edit-java | cut -f1) > extract-java-cd/casper/filesystem
 .size
3 exit
```

---

Nakon toga upisati naziv image-a unutar README.diskdefines. Upisati 'Ubuntu with Java 18.04.4 LTS "Bionic Beaver" - Release amd64' u polje DISKNAME:

---

```
1 sudo gedit extract-java-cd/README.diskdefines
```

---

## Generisanje Ubuntu .iso image sa Java modulom

Ažurirati md5sum.txt datoteku:

---

```
1 cd extract-java-cd
2 sudo rm md5sum.txt
3 find -type f -print0 | sudo xargs -0 md5sum | grep -v isolinux/boot.cat | sudo tee md5sum
.txt
```

---

Napokon možemo napraviti iso image koji će da sadrži Java modul. Za ovu operaciju koristimo funkciju genisoimage. Neke linux distribucije nude mkisofs funkciju. Tako da ukoliko ne radi genisoimage trebala bi raditi funkcija mkisofs:

---

```
1 sudo genisoimage -D -r -V "$IMAGE_NAME" -cache-inodes -J -l -b isolinux/isolinux.
bin -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table -o ../
ubuntu-with-java-18.04-amd64.iso .
```

---

## Pokretanje iso image-a pomoću kvm biblioteke

Sada ćemo napraviti virtuelni hard disk pomocu qemu-img komande da bismo pokrenuli na njemu naš novi modul Java Ubuntu.

---

```
1 cd ~
2 qemu-img create ubuntujava.img 5G
```

---

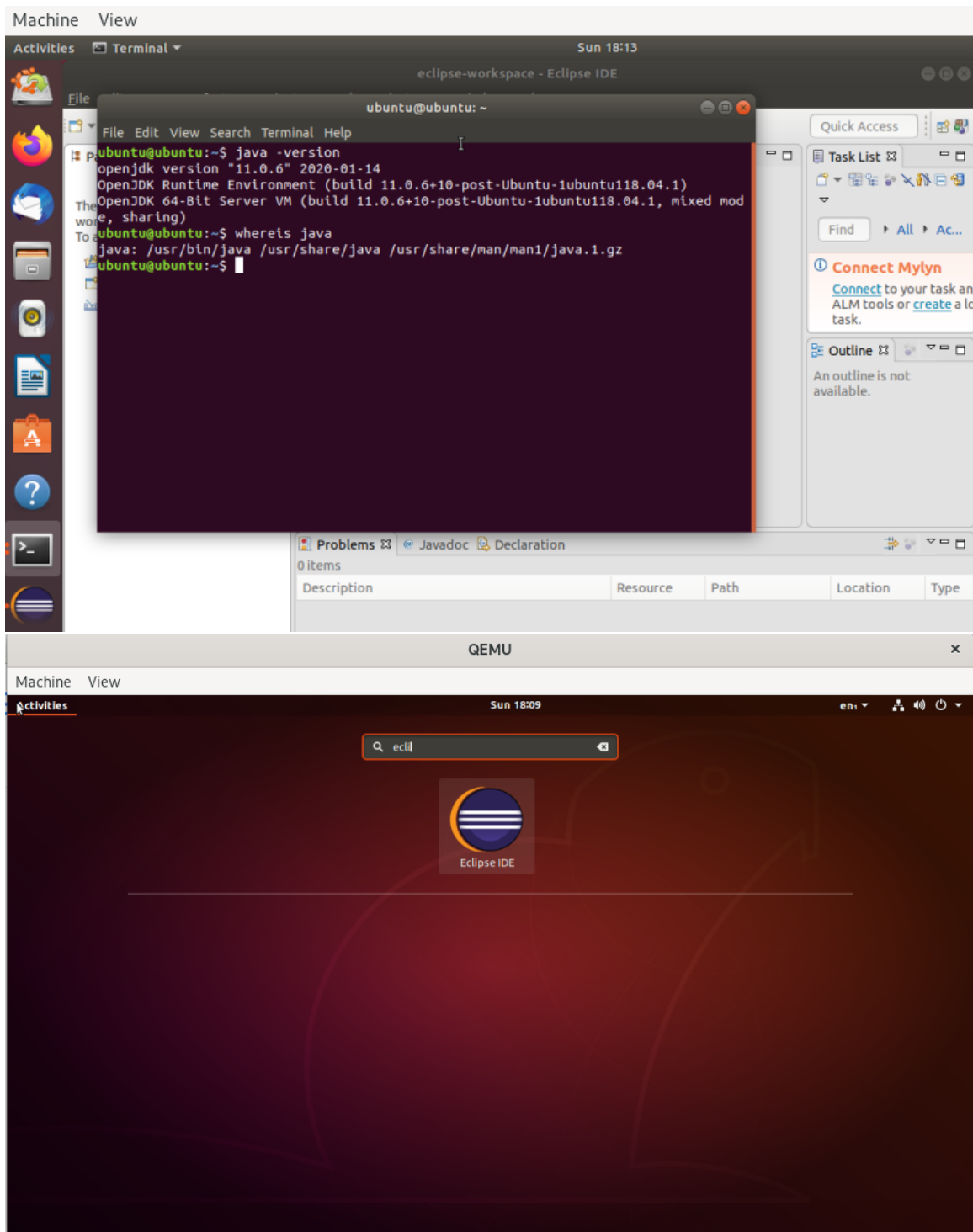
Pokrenućemo modul pomoću KVM-a:

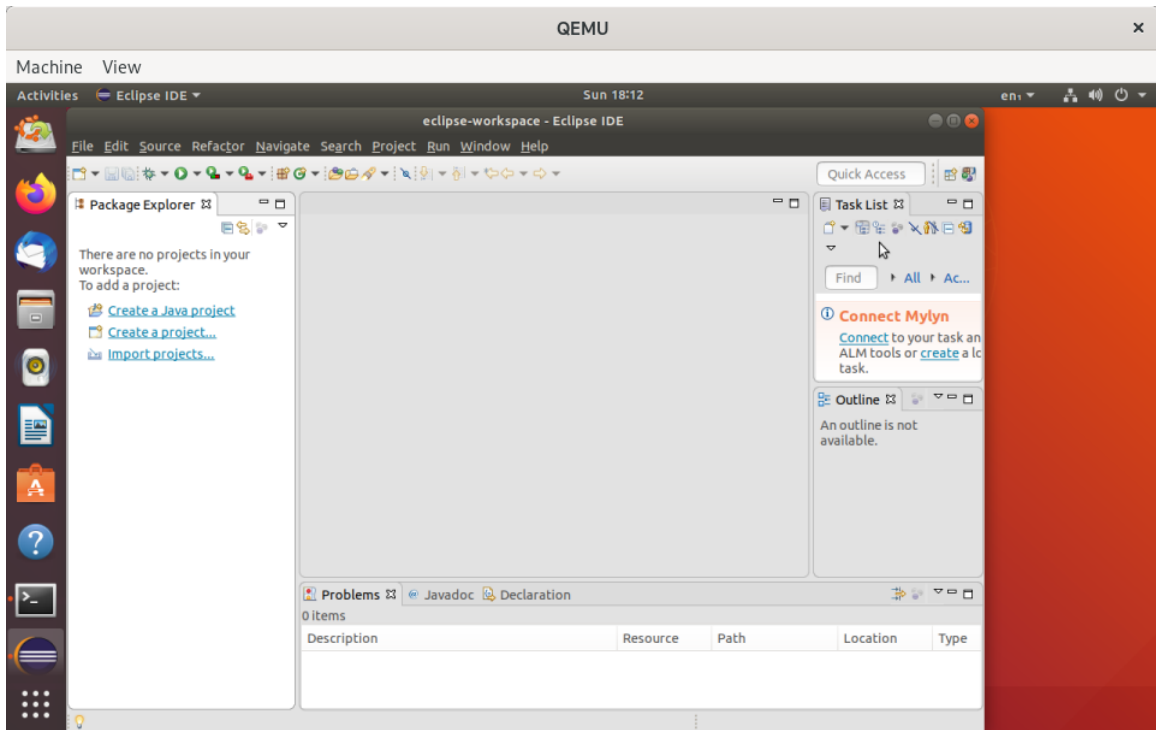
---

```
1 sudo kvm -hda ubuntujava.img -cdrom ~/zavrsni/livecdtmp/ubuntu-with-java-18.04-
amd64.iso -boot d -m 2048
```

---

## Rezultat Modul Java





## Modul Chrome

Kao treći primjer modularizacije squashfs datotečnog sistema, kreiran je modul Chrome. Kao što mu ime kaže, riječ je o modulu sa instaliranim Chrome pretraživačem paketa. Većinom koraci su identični kao u prethodna 2 slučaja, izuzev koraka instaliranja dodatnih paketa unutar modula.

### Kreiranje direktorija potrebnih za rad

---

```
1 cd ~/squashfs/livedtmp
2 sudo mount -o loop ./isoimgs/ubuntu-18.04.4-desktop-amd64.iso mnt
3 mkdir extract-chrome-cd
4 sudo rsync --exclude=/casper/filesystem.squashfs -a mnt/ extract-chrome-cd
5 mkdir modul-chrome
6 sudo rsync -a extract-chrome-cd/ modul-chrome
```

---

### unsquashfs filesystem.squashfs datoteke

Zatim slijedi korak u kojem se opet raspakuje filesystem.squashfs direktorij i kopiramo ga u edit-chrome direktorij. Ovaj put ćemo edit direktorij imenovati edit-chrome da ne izgubimo prethodni sadržaj edit direktorija.

---

```
1 sudo unsquashfs mnt/casper/filesystem.squashfs
2 sudo mv squashfs-root/ edit-chrome
```

---

### Konfiguracija paketa za chroot okruženje

Da bi imali mrežnu konekciju unutar edit-chrome direktorija jedno rješenje je kopirati /run direktorij unutar edit-chrome direktorija. Najbolje manuelno popuniti resolv.conf unutar edit-chrome direktorija

*nameserver 1.1.1.1*

*nameserver 8.8.8.8.*

Isto važi i za `etc/hosts` datoteku. Najbolje je provjeriti nakon izvršenih komandi da li je upisan sadržaj u `resolv.conf` i `hosts` datoteke, te ukoliko nije dopuniti nedostatke:

---

```
1 sudo cp /etc/resolv.conf edit-chrome/etc/
2 sudo mount -o bind /run/ edit-chrome/run
```

---

Kopirati i `hosts` direktorij/:

---

```
1 sudo cp /etc/hosts edit-chrome/etc/
```

---

Namjestiti `edit-chrome/dev` direktorij kopirajući `/dev/` direktorij sa hosta, zatim `chroot` u `edit-chrome` direktorij. Obaviti `mount` instrukcije navedene ispod. Ukoliko korisnik odluči da obriše `edit-chrome` direktorij iz nekog razloga, bilo bi potrebno uraditi `umount` `edit-chrome` direktorija da sistem domaćin ne bi postao neupotrebljiv:

---

```
1 sudo mount --bind /dev/ edit-chrome/dev
2 sudo chroot edit-chrome
3 mount -t proc none /proc
4 mount -t sysfs none /sys
5 mount -t devpts none /dev/pts
```

---

Također potrebno je izvršiti sljedeće komande da bi se izbjegli problemi sa lokalizacijom:

---

```
1 export HOME=/root
2 export LC_ALL=C
```

---

## Instalacija Google Chrome paketa unutar chroot okruženja

Za ispis svih instaliranih paketa:

---

```
1 dpkg-query -W --showformat='${Installed-Size}\t\${Package}\n' | sort -nr | less
```

---

Instalacija `google-chrome` paketa:

---

```
1 sudo nano /etc/apt/sources.list.d/google-chrome.list
```

---

Te upisati u ovu datoteku sljedeći sadržaj

---

```
1 deb [arch=amd64] http://dl.google.com/linux/chrome/deb/ stable main
```

---

Zatim spasiti datoteku unutar nano editora sa **CTRL+O**, **ENTER** za potvrdu i **CTRL+X** za izlaz iz nano editora.

Sljedeća komanda preuzima Google javni ključ da bismo mogli instalirati google-chrome. Zatim komandom apt-key dodajemo ključ u prsten javnih ključeva da bi apt mogao potvrditi integritet Google Chrome paketa.

---

```
1 wget https://dl.google.com/linux/linux_signing_key.pub
```

```
2 sudo apt-key add linux_signing_key.pub
```

---

Sada izvršimo ažuriranje liste paketa i instaliramo google-chrome-stable paket:

---

```
1 apt update
```

```
2 apt install google-chrome-stable
```

---

Provjera instalacije:

---

```
1 google-chrome-stable --version
```

---

Nakon završetka instalacije izvršiti unutar chroot:

---

```
1 apt-get clean
```

```
2 rm -rf /tmp/* ~/.bash_history
```

```
3 rm -rf /tmp/* ~/.bashrc
```

```
4 rm /var/lib/dbus/machine-id
```

```
5 rm /sbin/initctl
```

```
6 dpkg-divert --rename --remove /sbin/initctl
```

```
7 umount /proc || umount -lf /proc
```

```
8 umount /sys
```

```
9 umount /dev/pts
```

```
10 exit
```

---



Ponovno generisati filesystem.manifest:

---

```
1 sudo chmod +w extract-chrome-cd/casper/filesystem.manifest
2 sudo su
3 chroot edit-chrome dpkg-query -W --showformat='${Package} ${Version}\n' > extract
 -chrome-cd/casper/filesystem.manifest
4 exit
5 sudo cp extract-chrome-cd/casper/filesystem.manifest extract-chrome-cd/casper/
 filesystem.manifest-desktop
6 sudo sed -i '/ubiquity/d' extract-chrome-cd/casper/filesystem.manifest-desktop
7 sudo sed -i '/casper/d' extract-chrome-cd/casper/filesystem.manifest-desktop
8 sudo umount edit/dev
```

---

## Generisanje filesystem.squashfs direktorija

Sada cemo upotrijebiti drugu funkciju iz squashfs-tools, a to je mksquashfs. S tom funkcijom cemo kompresovati edit-chrome direktorij u novu filesystem.squashfs datoteku. U kodu ispod je potrebno izvršiti komandu iz linije 1 i jednu od preostale 3, pri čemu prva (komanda na liniji 2) daje najslabiju kompresiju, ali je najbrza. Druga komanda se duže izvršava ali je veći procenat kompresije u odnosu na prvu komandu. Dok je kod treće komande procenat kompresije najveći, a vrijeme izvršenja najduže:

---

```
1 sudo rm extract-chrome-cd/casper/filesystem.squashfs
2 sudo mksquashfs edit-chrome extract-chrome-cd/casper/filesystem.squashfs -nozma
3 sudo mksquashfs edit-chrome extract-chrome-cd/casper/filesystem.squashfs -b 1048576
4 sudo mksquashfs edit-chrome extract-chrome-cd/casper/filesystem.squashfs -comp xz -e
 edit/boot
```

---

Naredni korak je da ažuriramo filesystem.size datoteku:

---

```
1 sudo su
2 printf $(du -sx --block-size=1 edit-mysql | cut -f1) > extract-chrome-cd/casper/
 filesystem.size
3 exit
```

---

Nakon toga upisati naziv image-a unutar README.diskdefines. Upisati 'Ubuntu with Google Chrome 18.04.4 LTS "Bionic Beaver" - Release amd64' u polje DISKNAME:

```
1 sudo gedit extract-chrome-cd/README.diskdefines
```

---

## Generisanje Ubuntu .iso image sa Google Chrome modulom

Ažurirati md5sum.txt datoteku:

```
1 cd extract-chrome-cd
2 sudo rm md5sum.txt
3 find -type f -print0 | sudo xargs -0 md5sum | grep -v isolinux/boot.cat | sudo tee md5sum
.txt
```

---

Napokon možemo napraviti iso image koji će da sadrži Google Chrome modul. Za ovu operaciju koristimo funkciju genisoimage. Neke linux distribucije nude mkisofs funkciju. Tako da ukoliko ne radi genisoimage trebala bi raditi funkcija mkisofs:

```
1 sudo genisoimage -D -r -V "$IMAGE_NAME" -cache-inodes -J -l -b isolinux/isolinux.
bin -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table -o ../
ubuntu-with-chrome-18.04-amd64.iso .
```

---

## Pokretanje iso image-a pomoću kvm biblioteke

Sada ćemo napraviti virtuelni hard disk pomocu qemu-img komande da bismo pokrenuli na njemu naš novi modul Google Chrome Ubuntu.

```
1 cd ~
2 qemu-img create ubuntuchrome.img 5G
```

---

Pokrenucemo modul pomoću KVM-a:

```
1 sudo kvm -hda ubuntuchrome.img -cdrom ~/zavrzni/livecdtmp/ubuntu-with-chrome
-18.04-amd64.iso -boot d -m 2048
```

---

## Rezultat Modul Chrome

Ovje će biti tekst + slike vezano za GoogleChrom Modul.