Konfiguracija Linuxove jezgre

Šušnjak, Lukas

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Lukas Šušnjak

Konfiguracija Linuxove jezgre

ZAVRŠNI RAD

Varaždin, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Lukas Šušnjak

Matični broj: 0016149103 7

Studij: Primjena informacijske tehnologije u poslovanju

Konfiguracija Linuxove jezgre

ZAVRŠNI RAD

Mentor:

Dr. sc. Luka Milić

Varaždin, lipanj 2024.

Lukas Šušnjak

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojega rada ter da se u njegovoj izradbi nisam koristio drugim izvorima osim onima koji su u njem navedeni. Za izradbu rada su rabljene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredaba u sustavu FOI-radovi

_____

Sažetak

Kroz ovaj rad prikazuje se proces stvaranja Linuxovih jezgara kroz proučavanje sustava za izgradnju jezgara i njegovih dijelova. U radu se općenito opisuje stvaranje Linuxovih jezgara kroz upoznavanje semantike Kconfig-sustava i jezika koji čini Kbuild-sustav. Isto tako, upoznaje se s specifičnim opcijama konfiguracije kroz proučavanje i usporedbu konfiguracija dviju znamenitih jezgara, čim se dobiva uvid u njihove primjene, ter se prikazuju opcije s većim utjecajem na krajnji sustav. Na kraju, novodobivenim uvidima, stvara se i izgrađuje vlastita konfiguracija.

Ključne riječi: Linux, jezgra, konfiguracija, Ubuntu, ArchLinux, Kbuild, Kconfig

Kazalo

1. Uvod

Kao Windows i MacOS Linux je jedan od najzastupljenijih operacijskih sustava danas, i rabi se na različnim uređajima od prijenosnika, osobnih računala, pokretnih uređaja, i za razliku od ostalih operacijskih sustava namijenjenih za krajnje korisnike, Linux se rabi i u poslužiteljima, ugradbenima i IoT-uređajima kao što su automobili, pametni hladnjaci, pametni kućni uređaji i više. Jedna od stvari koja odvaja Linux od ostalih operacijskih sustava je njegova iznimno visoka mogućnost prilagođivanja određenim potrebama korisnika. Središnji dio operacijskoga sustava koji određuje veze između sklopovskoga i programskoga dijela uređaja je jezgra operacijskoga sustava, i ona je ključna za porabu uređaja.

Linux i njegova jezgra su otvorenoga tipa (*open-source*) što znamenuje da ga korisnici mogu prilagođivati po svojim potrebama, i to potiče zanesenjake još od začetaka osobnih računala konfigurirati jezgre da bi postignuli najbolje brzine, najmanje potrošnje i smanjili veličinu jezgara za svoja računala. S tim u ovom radu će se uspoređivati jezgre dviju znamenitih Linuxovih distribucija, koje su namijenjene za različne svrhe i korisnike, i usporedba će se usredotočiti na bitnije postavke Linuxove jezgre, koje će se dalje rabiti za konfiguracije vlastite jezgre. Nakon donašanja odluke za koju porabu će se konfigurirati Linuxova jezgra proučit će se koje će opcije konfiguracije biti potrebno mijenjati da bi se postignuli željeni rezultati. Zatim će se jezgra izgraditi i instalirati kako bi bila porabljiva.

Motivacija za ovaj rad je bolje upoznavanje s Linuxovom jezgrom ter približavanje glavnih postavaka i njihovih utjecaja na sustav čitatelju rada. Poznavanje Linuxova operacijskoga sustava je iznimno cijenjeno u industriji s obzirom na njegovu raširenost, ter kroz proučavanje njegove jezgre i najvažnijih postavaka operacijski sustav se još bolje upoznaje i dobiva se bolje razumijevanje oruđa i programa s kojima se susreće kroz svakodnevnu porabu Linuxa.

1.1. Metode i tehnike rada

Kako bi bilo moguće usporediti konfiguracije jezgre i konfigurirati jezgru po željenim potrebama ključno je obratiti pozornost na nekoliko bitnih dijelova cijeloga postupka. Potrebno je odabrati distribucije čije konfiguracije jezgara će se uspoređivati, zatim je na svakoj distribuciji potrebno preuzeti jezgru i locirati konfiguracijske datoteke tih distribucija. Isto tako je bitno odabrati metodu, dotično sučelje kojim će se konfigurirati jezgra, i metodu izgradnje jezgre.

Za proučavanje i konfiguriranje Linuxove jezgre postoje različite metode, neke učinkovitije, neke sveobuhvatnije, a neke s drugim namjenama. Kad se Linuxova jezgra konfigurira i ugrađuje, opcije njezine konfiguracije se pohranjuju u datoteki pod nazivom *.config*, u koju su upisane sve konfiguracijske postavke i njihovo stanje u obličju „*CONFIG_naziv=stanje*“. Uređivanje te datoteke je jedan od sveobuhvatnijih načina za konfiguriranje Linuxove jezgre, ali ujedno i najkompliciraniji. Za usporedbu konfiguracija jezgara a i za samu konfiguraciju vlastite jezgre rabit će se Linuxova oruđa Kconfig i Kbuild. Kconfig nudi mogućnost konfiguriranja jezgre u menuconfig-sučelju. Navedena oruđa pružaju jednostavniji i pristupačniji način proučavanja Linuxove jezgre.

Sljedeća odluka koju je potrebno donijeti je na koji način će se uspoređivati konfiguracije jezgara i kako će se konfigurirati nova jezgra. Kako bi se mogle usporediti konfiguracije jezgara dviju distribucija, potrebno ih je prvo odabrati. Distribucije odabrane su ArchLinux i Ubuntu, s obzirom na to da su ciljane na poprilično različne publike, i konfigurirane za različne namjene. Arch je namijenjen za ispravljatelje (*debuggere*), razvijatelje i upravitelje (*administratore*), a Ubuntu namijenjen za širu javnost, svakodnevnoga korisnika, svakodnevne zadatke. Naravno obje distribucije se mogu rabiti u sve svrhe ali ipak Arch će na primjer biti bolji izbor za *debugging*. Usporedbom dviju konfiguracija jezgara dviju distribucija s različitim namjenama može se doći do zaključka koje opcije jezgre su bitnije i imaju veći utjecaj na ponašanje sustava.

Nakon ovoga potrebno je odlučiti s kojom svrhom, i s kojom ciljanom porabom sustava će inačica jezgre biti konfigurirana u ovom radu. Zamisao ove konfiguracije jezgre je imati jezgru bez nepotrebnih pogonitelja, opcija, čim će se smanjiti veličina jezgre, i ubrzati dizanje sustava. Ovakova jezgra rabit će se na malom kućnom poslužitelju medija.

2. Općenito o konfiguraciji jezgre i stvaranje znamenitih jezgara

Svaka Linuxova distribucija u središtu rabi Linuxovu jezgru, koju je 1991. razvio Linus Torvalds na varijaciju zamisli Unixova sustava. Danas se Linuxove jezgre rabe u svakom Linuxovu sustavu, od osobnih računala, do integriranih sustava. Specifičnost Linuxove jezgre je u tom što je ona modularna, mogu se dinamički uključivati i isključivati moduli, i iznimno je konfigurabilna, što je temelj ovoga rada. Svaka distribucija, bio to Ubuntu najviše ciljan za masovno tržište ili Arch koji je voljen među zanesenjacima, u središtu rabi Linuxovu jezgru. Osim slikovnoga sučelja, paketa i upravitelja paketima koji su inicijalno instalirani na distribucije, one se isto tako razlikuju u konfiguraciji same jezgre. Upravo ta jednostavnost konfiguracije jezgre omogućuje velik broj različnih distribucija s različnim namjenama. Svaka distribucija započinje s zamišlju namjene, nakon čega je potrebno konfigurirati jezgru. Razradit će se u nastavku postupak konfiguracije jezgre u svrhu stvaranja distribucijski specifične jezgre.

Za izgradnju jezgre za određenu distribuciju kao temelj uzimlje se osnovna jezgra, koju velik broj razvijatelja neprestano razvija, sukladno čemu postoje različne inačice osnovne jezgre. S obzirom na to što se distribucijom želi postignuti, razvijatelji distribucije mogu odlučiti kao temelj rabiti najnoviju jezgru, ili pak neku stariju jezgru. Distribucije najčešće imaju dvije vrste, *stable* ili *rolling release* (nazivi variraju od distribucije do distribucije, no navedeni su općeniti nazivi znani većini). Distribucija koja je stabilna rabit će posljednju stabilnu inačicu jezgre, i nju konfigurirati po parametrima i uvjetima distribucije, i distribucija će se osvježavati periodično svakih nekoliko mjeseci kad je to potrebno. Ako se razvije nova stabilna inačica jezgre, razvit će se nova inačica distribucije. Za razliku od stabilne distribucije, distribucija *rolling release* fokusirana je na *cutting edge*, i apsolutnu svježost, dakle za nju rabit će se najnovija inačica jezgre, i svakim će se novim izdanjem jezgre ta jezgra rabiti kao temelj distribucije. Nakon odabira osnovne jezgre potrebno je jezgru konfigurirati za potrebe distribucije i zatim ju izgraditi.

Dakle na primjeru distribucije Ubuntu, ako u terminalu pokrene se naredba *uname -a* dobiva se sljedeći ispis:

```
Linux lukas-Virtualbox 6.5.0-35-generic #35~22.04.1-Ubuntu SMP
PREEMPT_DYNAMIC Tue May 7 09:00:52 UTC 2 x86_84 x86_84 x86_84 GNU/Linux
```

Naravno, bitno je napomenuti da je naredba pokrenuta u virtualnom okružju, pa stoga vidi se ispis kao *linux-Virtualbox*. Iz ispisa ove naredbe vidi se da Ubuntu rabi jezgru „*6.5.0-35-generic*

#35~22.04.1-Ubuntu“, iz čega se može zaključiti da ta inačica Ubuntuova sustava kao osnovnu jezgru rabi jezgru inačice 6.5.0-35. Dakle kod stvaranja jezgre za ovu specifičnu verziju distribucije razvijatelji su konfigurirali jezgru 6.5.0-35.

Postoji veći broj načina konfiguracija jezgre, ali svi načini konfiguracije u srži se temelje na Linuxovu „Kernel Build Systemu“. Kbuild se može na najosnovnijoj razini podijeliti na:

1. Kconfig – dio za konfiguriranje opcija jezgre, kroz različne mete (*oldconfig*, *menuconfig*)
2. Kbuild odnosno Makefilei – dokumenti pisani u programskom jeziku makefile, koji opisuju kako će se pojedini dijelovi izvornoga koda jezgre graditi.

U nastavku će se opisati svaki od ovih dijelova zasebno, i bit će opisano kakovu ulogu oni imaju u izgradnji znamenitih jezgara, što će u primjeru biti Ubuntu, s obzirom na to da se u ovom radu radi s Ubuntuovom distribucijom.

2.1. Kconfig

Kconfig je skup kconfig-dokumenata koji se sastoje od definiranih opcija jezgre, dotično simbola. Većina opcija može postojati ili u jednom od dvaju stanja, dotično *boolean*, ili u jednom od triju stanja, dotično ona može biti *tristate*. Ako se radi o opciji tipa *boolean*, ona može biti uključena ili isključena, što znamenuje da će opcija biti ugrađena u jezgru ili ne će biti ugrađena u jezgru. *Boolean*-opcije mogu biti jednostavna obličja *y/n*, to jest, uključeno ili isključeno, ili pak mogu biti obličja isključeno ili uključeno s specifičnom vrijednošću. S druge strane sve opcije tipa *Tristate* mogu imati tri vrijednosti, uključeno, isključeno ili modul. Ako je opcija samo uključena ili isključena, učinak na jezgru je istovjetan kao i kod *boolean*-opcija, ali ako opcija ima stanje modul, ona nije izravno ugrađena u jezgru, dakle ne pokreće se kod dizanja sustava, ali ona je izgrađena i pripravna za ugradnju živoga sustava kad je to potrebno. Kad je sustav pokrenut i djelatno se rabi, ako se pokaže potreba za nekim pogoniteljima, oruđima, postavkama, skriptama, aplikacijama ili slično, odgovarajući modul može biti ugrađen u jezgru u tom trenutku, bilo to naredbom ili samo od sebe.

2.1.1. Opis konfiguracije jezgre

Kconfig je dakle središnji sustav koji se rabi za izgradnju jezgara. Različnost u postupku konfiguracije pojavljuje se kod odabira mete oruđa make. Najčešća meta za konfiguraciju je `make menuconfig` što nudi jednostavno konfiguriranje jezgre kroz slikovno sučelje ncurses, ali jezgra se može i konfigurirati tekstovno, na način da se uređuju i preinačuju datoteke kconfig. Kako bi se konfiguracija mogla jednostavno izvesti i proučiti, potrebno je poduzeti korake preuzimanja nove jezgre u Linuxov sustav, instaliranje potrebnih oruđa, te

preslikavanje trenutačne konfiguracije jezgre u kazala nove jezgre. Ovo je najjednostavniji način konfiguracije jezgre, ali kad se razvijaju nove distribucije postupak konfiguriranja je često ručan, ručnim uređivanjem datoteka. Nakon konfiguriranja jezgre razvijatelji zatim izgrađuju jezgru porabom datoteka makefile, isto tako preko oruđa make [1].

Kako bi se konfiguracija mogla započeti potrebno je preuzeti jezgru i pokrenuti Kconfig. Kako bi bilo moguće konfigurirati jezgru prvo je ključno preuzeti izvorni kod Linuxove jezgre na svaku distribuciju. Preuzimlje se posljednja verzija jezgre raspoloživa na *web*-stranici kernel.org u kazalo na lokaciji */usr/src*. Za preuzimanje izvornoga koda rabit će se oruđe wget. Nakon navigiranja u odgovarajuće kazalo, dotično */usr/src*, preuzimanje posljednje inačice 6.6 se obavlja naredbom:

```
wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.6.tar.xz
```

Nakon porabe prikazane naredbe u kazalo se preuzimlje komprimirana i arhivirana inačica Linuxove jezgre pod nazivom „*linux-6.6.tar.xz*“, ali izgled naziva dokumenta ovisi o inačici preuzete jezgre. Kako bi se izvorni kod jezgre mogao rabiti potrebno je datoteku dearhivirati i dekomprimirati, što se čini rabeći oruđe „*tar*“. Da bi se jezgra dearhivirala i dekomprimirala potrebno je pokrenuti sljedeću naredbu:

```
tar -xvf linux-6.6.tar.xz
```

U navedenoj naredbi vide se 3 glavna dijela: naziv naredbe, zastavice naredbe, i ciljana datoteka. Za dearhiviranje i dekomprimiranje rabeći „*tar*“-naredbu potrebne su sljedeće opcije: *x*, *v*, *f*. Slijedom njihovo znamenovanje je: dearhiviranje datoteke, ispis svih podataka i određivanje ciljanoga dokumenta. Nakon izvađanja navedenih naredaba dobiva se završna datoteka pod nazivom *linux-6.6* u kojoj se nahodi izvorni kod jezgre. [1]

Sljedeći koraci koje je potrebno izvesti kako bi se bilo u mogućnosti pokrenuti kconfig-sučelje i usporediti konfiguracije dviju distribucija su: lociranje i premještanje konfiguracijske datoteke distribucije u kazalo jezgre, te instaliranje nekoliko potrebnih oruđa koji su uvjet za pokretanje kconfig-sučelja. Ova dva koraka su ponešto drugačija na svakoj od distribucija ovisno o upravitelju paketa koji distribucija rabi, što je za Ubuntu *apt-get*, a za Arch je to *pacman*. Za pokretanje kconfig-sučelja potrebna su sljedeća oruđa: *ncurses*, *flex* i *bison* ter paket *build-essential*. Ncurses je oruđe koje pruža mogućnost izradbe sučelja koja sliče na tradicionalna slikovna sučelja unutar terminala, dakle to je oruđe koje je odgovorno za vizualni dio oruđa kconfig. Instaliranjem paketa build-essential glavno oruđe koje se instalira je *make*, ali isto tako uza nj instaliraju se i ostala oruđa koja se mogu pokazati korisnima kod kakove bilo izgradnje, ne samo jezgre.

Ncurses se preuzimlje i instalira na sljedeći način:

Ubuntu: `sudo apt-get install libncurses5-dev`

Arch: `sudo pacman -S ncurses`

Flex i bison se preuzimlju i instaliraju na sličan način.

Flex: Ubuntu: `sudo apt-get install flex`

Arch: `sudo pacman -S flex`

Bison: Ubuntu: `sudo apt-get install bison`

Arch: `sudo pacman -S bison`

Build-essential: Ubuntu: `sudo apt-get install build-essential`

Arch: `sudo pacman -S base-devel`

S instaliranim potrebnim oruđima za pokretanje menuconfig-sučelja, posljednji korak prije usporedbe je stvaranje preslike dokumenta konfiguracije jezgre distribucije koja se rabi. U većini slučajeva taj dokument na Arch-sustavu se nahodi na stazi */proc/config.gz*, osim ako je kod konfiguriranja jezgre isključena postavka *CONFIG_IKCONFIG*. Kako bi se datoteka premjestila u kazalo */usr/src/linux-6.6*, potrebno je rabiti oruđe zcat. Zcat je oruđe čijom porabom na komprimiranoj datoteki se ispisuje u terminal sve unutar komprimiranoga dokumenta bez dekomprimiranja toga dokumenta.

Kako bi se sastav toga dokumenta premjestio u novu datoteku unutar kazala */usr/src/linux-6.6*, potrebno se je pozicionirati u kazalo u koje je potrebno pohraniti novu datoteku konfiguracije i izvršiti sljedeću naredbu:

```
zcat /proc/config > .tempconf
```

Naredbom ozgora sav ispis koji bi se uobičajeno ispisao u prozor terminala preusmjeruje se u novu datoteku pod nazivom *.tempconf*, i ta se datoteka može poslije ukrcati u oruđe kconfig.

Za Ubuntu postupak se malo razlikuje jer se konfiguracija Linuxove jezgre u Ubuntu-sustavu nahodi u kazalu */boot*, u čistom tekstovnom formatu i najbolji je način za preslikavanje poraba cp-naredbe. Dakle, pozicionirani u kazalu */boot* pokreću naredbu:

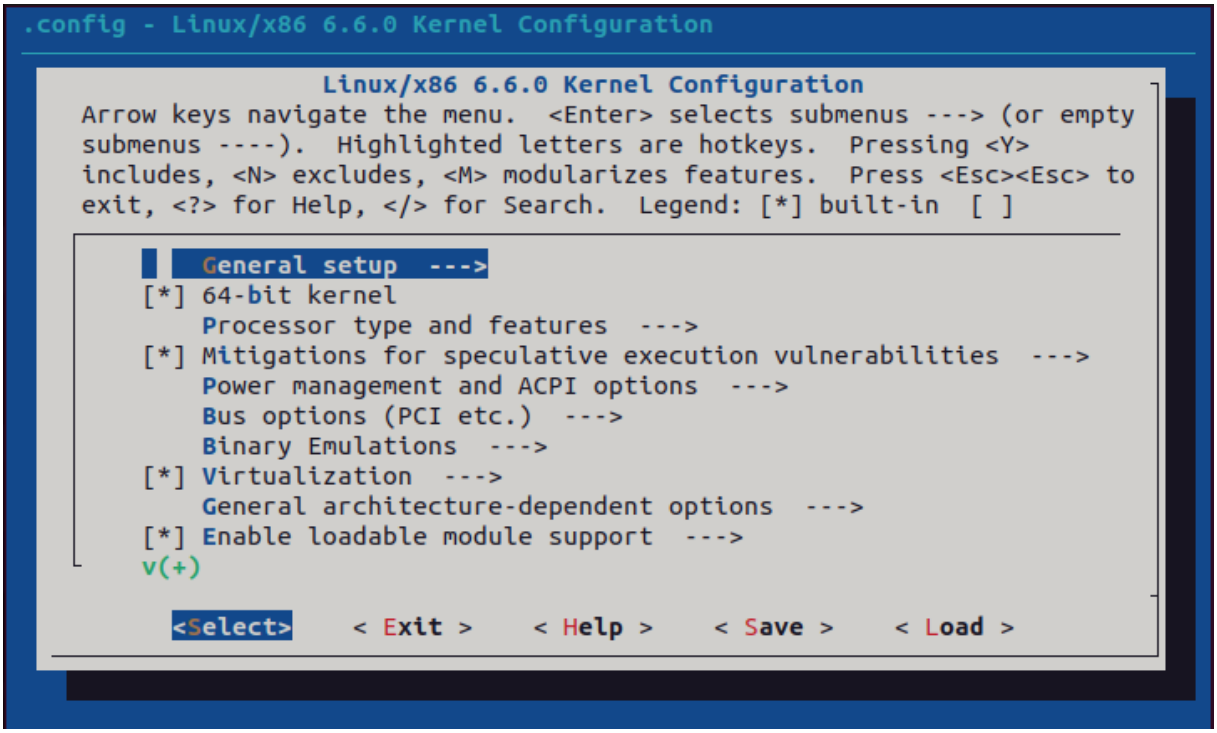
```
cp config-$(uname -r) /usr/src/linux-6.6/.tempconf
```

U naredbi ozgora dio koda *$(uname -r)* nakon pokretanja mijenja taj dio koda u trenutačnu inačicu jezgre koja se rabi na sustavu [2].

Nakon izvođenja navedenih naredaba moguće je pokrenuti konfiguraciju jezgre kroz jednu od mogućih meta. To može biti *make oldconfig*, za stvaranje konfiguracije temeljene na postojećoj

datoteki *.config*, *make defconfig* za stvaranje konfiguracije s razumijevanom Arch konfiguracijom, ili puno drugih. U ovom slučaju rabit će se *make menuconfig*, koji omogućuje konfiguriranje jezgre rabeći GUI-izbornik zasnivan na ncursesu.

U svakoj od distribucija naredba za pokretanje menuconfig-sučelja je jednaka, *sudo make menuconfig*. Nakon pokretanja naredbe prikaz će biti sličan ili isti (ovisno o inačici) onomu na sliki ozdola.



Slika 1. Prikaz sučelja menuconfiga

Kako bi bilo moguće vidjeti stanja svih postavaka distribucije koja je u porabi, potrebno je navigirati na *Load*, i upisati naziv konfiguracijske datoteke koja se je stvorila/premjestila u prijašnjem poglavlju, u ovom slučaju to je *.configTemp*. Nakon toga konfiguracija se može namjestiti po potrebama distribucije.

2.2. Kbuild i Makefiles

Nakon konfiguracije jezgre i spremanja konfiguracijske datoteke jezgru je potrebno izgraditi. Kbuild se sastoji od datoteka makefile koje uzimlju kao referenciju uređene datoteke kconfig, dotično datoteke kconfig u kojima su stanja opcija postavljena. U datoteka makefilema navedene su upute kako će se koja opcija kompilirati i izgraditi, kako će se izgraditi završna slika jezgre, i kako će biti izgrađeni moduli. Makefile rabi datoteku *.config* i iz nje iščitava stanja

opcija, po čem izgrađuje jezgru. Ne postoji samo jedna datoteka makefile, nego za gotovo svako kazalo postoji *Makefile* koji opisuje specifičnosti izgradnje slike jezgre i modula. Makefile-podsustav rabi 5 dijelova:

1. *Makefile* – misli se na vršni makefile-dokument koji je temelj za rekurzivnu gradnju jezgre.
2. *.config* – datoteka koja u čitljivu formatu prikazuje sve konfiguracijske opcije i njihovo odabrano stanje
3. *arch/$(ARCH)/Makefile* – makefile koji služi kao dodatak vršnomu makefileu te ima u sebi koje su specifične za različne arhitekture.
4. *scripts/Makefile.** – pravila za izvađanje svih makefile-dokumenata.
5. kbuild-makefilei – oko 500 dokumenata za svaki od modula [3]

Nakon što se preko neke od meta stvori konfiguracijska datoteka *.config* za jezgru, potrebno je jezgru izgraditi.

Najjednostavniji način za izvađanje ovoga koraka je unutar kazala izvornoga koda jezgre pokrenuti tri naredbe:

1. *make* – naredba za kompiliranje jezgre i modula ter izgradnju slike jezgre
2. *make modules_install* – naredba za instaliranje svih modula
3. *make install* – naredba za instaliranje jezgre [4]

grade jezgru na rekurzivan način što znamenuje da pokretanje make-naredbe će raditi rekurzivno, dotično, kad make-naredba izgradi sve što može unutar jednoga kazala, spušta se u potkazala i gradi se iz datoteka makefile u svakom od potkazala. Kako bi se jezgra pravilno izgradila, korisno je razumjeti i makefile-jezik, što je istumačeno u sljedećem poglavlju [3].

2.2.1. Makefile-jezik

Make je Linuxovo oruđe koje se rabi za kompiliranje i izgradnju aplikacija ter u ovom slučaju Linuxove jezgre. Make omogućuje jednostavnije praćenje jesu li već kompilirani i izgrađeni dijelovi koda o kojima je ovisan dio koda koji se trenutačno želi kompilirati odnosno izgraditi. Isto tako, oruđe make znatno pojednostavljuje postupak ponovnoga kompiliranja i ponovne izgradnje nekoga dijela koda nakon što je on promijenjen. Kako bi oruđe make bilo djelujuće i kako bi se moglo rabiti za navedene porabe, ključno je da postoji *makefile*.

Makefile je datoteka u kojoj se makeom, to jest makefile-jezikom opisuju zadatci koji će se izvršavati kada se pokrene oruđe make. U datoteki makefile se navode sve datoteke o kojima ovisi datoteka koja se želi kompilirati i izgraditi, naredbe za kompiliranje i izgradnju, ter

odabrane dodatne naredbe. U najosnovnijem dijelu Makefile se sastoji od mete i naputka. Najjednostavniji primjer make-koda izgleda ovako:

```
meta:

	naputak
```

Kako bi se datoteka makefile pokrenula, potrebno je u terminalu u kazalu u kojem se nahodi datoteka makefile izvršiti naredbu `make`. Nakon pokretanja izvršava se prva meta u datoteki makefile, što je najčešće *all*. U slučaju da kod u makefile-dokumentu izgleda kao navedeno ozgora, meta se u ovom slučaju nazivlje *phony* meta, što znamenuje da se ne misli na određenu datoteku, nego je samo naziv za navedeni naputak. *Phony* mete mogu djelovati samo ako u kazalu u kojem se nahodi Makefile ne postoji istoimena datoteka. U vidu Linuxove jezgre *phony* mete s kojima se najčešće susreće su *all*, *modules*, *modules_install* i *clean*. Naputak za metu *clean* u pravilu nije naveden pod metom *all*, za razliku od većine ostalih meta.

Ali se rabe za kompiliranje i izgradnju modula i slike jezgre, što nije moguće napraviti porabom isključivo *phony* meta, pa je potrebno da datoteka makefile ima u sebi i kod sljedećega izgleda:

```
meta.o : ovisnost1.o ovisnost2.o

	naputak
```

U kodu prikazanom ozgora razumijeva se sljedeće: *meta.o* predstavlja dokument koji se želi kompilirati i izgraditi, *ovisnost1.o* i *ovisnost2.o* predstavljaju datoteke i objekte o kojima *meta.o* ovisi kako bi bila pravilno kompilirana odnosno izgrađena, a *naputak* je skup uputa kojima se kompilira i izgrađuje *meta.o*. Kad se izvrši naredba `make` ako je *meta.o* prva meta u makefile-dokumentu ili ako je navedena u *phony* meti *all*, onda program prvo pregledava postoje li datoteke *ovisnost1.o* i *ovisnost2.o*, ter ako ne postoje, u datoteki makefile se pretražuju mete naziva *ovisnost1.o* i *ovisnost2.o* ter se one prve kompiliraju i izgrađuju. Nakon što je to izvršeno, i potvrđeno je da postoje sve datoteke o kojima ovisi *meta.o*, izvršava se naputak za tu metu. Filozofija makefile-jezika je poprilično jednostavna, i istumačeno je u glavnom dovoljno za izgradnju jednostavnih, djelujućih i primjenjivih datoteka makefile, ali kad se radi o izgradnji Linuxove jezgre, i programiranju novih modula za Linuxovu jezgru, žarište se stavlja na druge dijelove datoteka makefile [5] [6].

Kad se kompilira i gradi Linuxova jezgra, pozornost se obraća na module i ugrađene module. Po završetku postupka izgradnje jezgre porabom oruđa make, dobivaju se dva glavna dijela:

1. Slike Linuxove jezgre: nekomprimirana slika jezgre *vmlinux* i komprimirana slika *bzImage* koju rabi *bootloader*.

2. Datoteke modula jezgre za učitavanje: objektne datoteke modula [3] [6] [7] [8].

Nakon konfiguriranja jezgre dobiva se uređen oblik datoteke .config, koja je namijenjena da ju korisnici mogu čitati i uređivati, ali ona nije krajnji oblik vrijednosti opcija konfiguracije koju rabe kako bi kompilirale te izgradile jezgru. Kod izvršavanja make-naredbe stvara se još dodatnih datoteka, od kojih vrijedi spomenuti autoconf.h, koja ima u sebi popis svih konfiguracijskih opcija i njihova stanja u obliku #define, koju zatim makefile rabi za kompilaciju i izgradnju jezgre objekta modula. [9]

Unutar , što će se kompilirati i izgraditi, ter hoće li to biti ugrađeno u sliku jezgre ili izgrađeno kao modul za ukrcavanje, odlučuje se tako da se na temelju konfiguracijskih datoteka varijablama *obj-m* i *obj-y* dodjeljuju moduli, koji se zatim pozivlju u naputcima u obliku varijable. Istina, postupak je istumačen najjednostavnijim riječima, s obzirom na to da je pisanje datoteka makefile suradni rad više Linuxovih razvijatelja koji se je razvijao desetljećima, i u njima se i samim postupcima koji se odvijaju porabom oruđa make nahode dodatne složenosti, koje se u ovom radu ne će opisivati, nego će se tumačiti samo ono ključno za konfiguriranje i kompilaciju vlastite instancije jezgre, ter za dodavanje modula u jezgru.

Kao što je spomenuto, rabe prvotno varijable *obj-m* i *obj-y* za dodavanje modula u jezgru, kao module za ukrcavanje ili module koji će biti ukrcani u konačnu sliku jezgre, ali postoje dodatne varijable kao *always* koje pokazuju što će biti stalno ugrađeno u jezgru bez obzira na stanje opcije u konfiguracijskoj datoteki, ili pak *lib-y* koja pokazuje koji objekti će biti izgrađeni kao knjižnice. U srži *obj-m*, *obj-y*, *always* i *lib-y* su varijable tipa popisa, ali u dokumentaciji one se opisuju nazivom *ciljevi*, dakle na primjer, *obj-m* su ciljevi za module koji se učitavaju, a *obj-y* su ciljevi za ugrađene objekte [3].

U konkretnom primjeru objekti se dodaju u na sljedeći način:

- za ugrađene objekte ili module za ukrcavanje:

```
obj-$(CONFIG_opcija)    +=    opcija1.o
```

- za objekte ili module koji se grade iz više datoteka:

```
obj-$(CONFIG_opcija)    +=    opcija1.o
opcija1-y   :=    datoteka1.o datoteka2.o datoteka3.o
```

- za knjižnice:

```
lib.y       :=    objektBiblioteke.o
```
[3]

Valja napomenuti da se u navedenim primjerima ozgora operator pridruživanja razlikuje u nekim redcima koda. Makefile-jezik rabi nekoliko načina pridruživanja, a to su:

- = : ovo je operator pridruživanja u klasičnom smislu riječi, gdje se sve na desnoj strani operatora pridružuje varijabli s lijeve strane operatora;
- += : ovo je operator koji služi za proširivanje prije definirane varijable. Dakle ako se nešto želi pridružiti na varijablu koja je definirana operatorom =, u nasljednim redcima koda može se rabiti += za dodavanje toj varijabli;
- :=: ovaj operator nosi naziv „jednostavno proširena varijabla“ što efektivno znamenuje da, ako se varijabli pridruži neka druga varijabla koja je definirana prije varijable u kojoj se ona pozivlje i ponovno nakon nje, da će se rabiti instancija koja je definirana prije;
- ?= : jednostavan operator koji omogućuje pridruživanje varijabala samo ako nešto njoj nije prije pridruženo [10] [11].

```
lukas@server1:/usr/src/linux-6.9.4/init$ cat Makefile
# SPDX-License-Identifier: GPL-2.0
#
# Makefile for the linux kernel.
#

ccflags-y := -fno-function-sections -fno-data-sections

obj-y                          := main.o version.o mounts.o
ifneq ($(CONFIG_BLK_DEV_INITRD),y)
obj-y                          += noinitramfs.o
else
obj-$(CONFIG_BLK_DEV_INITRD)   += initramfs.o
endif
obj-$(CONFIG_GENERIC_CALIBRATE_DELAY) += calibrate.o

obj-y                          += init_task.o

mounts-y                        := do_mounts.o
mounts-$(CONFIG_BLK_DEV_RAM)    += do_mounts_rd.o
mounts-$(CONFIG_BLK_DEV_INITRD) += do_mounts_initrd.o
```

Slika 2. Primjer izgleda datoteke *init/Makefile*

Spomenuto je da oruđe make radi na rekurzivan način, dakle da se spušta u potkazala u kojima zatim isto tako izvršava Makefile za to kazalo, ali make se ne spušta u sva potkazala koja postoje u kazalu u kojem se nahodi vršni makefile, nego je potrebno posebno definirati u vršnoj datoteki makefile u koja potkazala se je potrebno spuštati. U Linuxovoj jezgri ovo je definirano u datoteki kbuild, isto tako rabeći obj-m, odnosno obj-m, na sljedeći način:

- Za potkazala u koja se je obvezatno spuštati:

```
obj-y =     poddirektorij/
```

- Za potkazala koja ovise o stanju opcije konfiguracije:

```
obj-$(CONFIG_POSTAVKA)        =      postavka/
```

Kad se radi o potkazalima, potkazalo se obilježava simbolom „/“, i zahvaljujući tomu simbolu oruđe make znade da se radi o potkazalu a ne o datoteki, što je standard koji vlada kroz cijeli Linuxov sustav. Za vrijeme izvršavanja make-naredbe sve datoteke *obj-y* su kompilirane i zatim spojene u jedan arhiv *built-in.a* koji nema simbole konfiguracijskih opcija, kako bi se poslije ta datoteka uz pomoć skripte *link-vmlinux.sh* u kazalu *scripts* spojila u jednu izvršnu datoteku *vmlinux*. Ostali moduli koji se dinamično ukrcavaju u jezgru za vrijeme njezine porabe kompiliraju se u datoteke s proširkom *.ko*, uz neke dodatne datoteke, ali *.ko* datoteke su one koje se ukrcavaju u jezgru [3] [12].

Na slici ispod prikazan je isječak sadržaja korijenske Makefile datoteke, pod nazivom Kbuild, koja specificira kazala u koja se spušta te u kojima se prolazi kroz Makefile datoteke kada se pokrene *make* naredba.

```
# Ordinary directory descending
# ------------------------------------------------------------------------

obj-y                   += init/
obj-y                   += usr/
obj-y                   += arch/$(SRCARCH)/
obj-y                   += $(ARCH_CORE)
obj-y                   += kernel/
obj-y                   += certs/
obj-y                   += mm/
obj-y                   += fs/
obj-y                   += ipc/
obj-y                   += security/
obj-y                   += crypto/
obj-$(CONFIG_BLOCK)     += block/
obj-$(CONFIG_IO_URING)  += io_uring/
obj-$(CONFIG_RUST)      += rust/
obj-y                   += $(ARCH_LIB)
obj-y                   += drivers/
obj-y                   += sound/
obj-$(CONFIG_SAMPLES)   += samples/
obj-$(CONFIG_NET)       += net/
obj-y                   += virt/
obj-y                   += $(ARCH_DRIVERS)
```

Slika 3. Primjer datoteke Makefilea

3. Specifične opcije konfiguracije jezgre

Kako bi se pobliže prikazale najučestalije konfiguracije Linuxove jezgre, usporedit će se dvije popularnije distribucije, s različnim namjenama. Ovom usporedbom ne dobiva se samo uvid u najučestalije opcije nego i grub uvid u to koja je namjena svake jezgre, kroz iščitavanje različnosti između dviju konfiguracija.

3.1. *General setup*

Redom se prikazuju opcije čija se stanja razlikuju između sustava Arch i Ubuntu, redom prikazivanja u menuconfig-sučelju, s njihovim CONFIG-nazivom, nakon kojega je u zagradama prikazano na kojem sustavu je opcija omogućena a slijedi opis opcije, i razlog za različnost između dviju distribucija.

1. *CONFIG_LOCALVERSION_AUTO* (Arch)

   Omogućuje da se na naziv inačice jezgre sam od sebe dodaje tekst ako je izvor jezgre *git-tree*. Ovo je omogućeno na Arch-sustavu i u glavnom korisno ispravljateljima i razvijateljima kojima je bitno po tanko poznavati sustav [13].

2. *CONFIG_USELIB* (Ubuntu)

   Opcija *uselib syscall* omogućuje veću unatražnu kompatibilnost s starijim C-knjižnicama. [13]

   Kod Arch-sustava ova opcija nije omogućena, iz čega se može zaključiti da je Arch-sustav više namijenjen korisnicima koji traže *bleeding-edge* oruđa, i isključivanjem se ove opcije smanjuje *overhead* na jezgri.

3. *Timers subsystem*

   a. Ubuntu: *CONFIG_NO_HZ_IDLE*

   b. Arch: *CONFIG_NO_HZ_FULL*

   Jedna od bitnijih opcija je odabir načina rada podsustava brojača (*Timers subsystem*). Na Ubuntu-sustavu rabi se *CONFIG_NO_HZ_IDLE*, dotično *tickless idle*, što znamenuje da se brojač isključuje kad je sustav *idle*, to jest ništa ne čini. Arch-sustav rabi *CONFIG_NO_HZ_FULL*, dotično *Full dynticks system*, što omogućuje još dodatno smanjivanje potrošnje. Ovo je vjerojatno zbog pretpostavke da će se Arch rabiti na jačim uređajima nego Ubuntu koji je namijenjen svakodnevnomu korisniku [14].

4. *CONFIG_BPF_PRELOAD* (Arch)

   Ova postavka je dio podsustava BPF (Berkeley Packet Filter), i omogućena je na Arch-sustavu. BPF je početno bio sučelje za nadzor i pročišćavanje primljenih i poslanih paketa na povezničkom sloju, ali djelovanje je znatno prošireno. Danas se tehnologija rabi za pisanje programa koji se mogu odvijati na razini jezgre, i služiti kao dodatci određenim dijelovima operacijskoga sustava. Ova postavka omogućuje ukrcavanje u sustav nekih BPF-programa koji su korisni ispravljateljima i razvijateljima [15].

5. *Preemption model*

   c. Ubuntu: *CONFIG_PREEMPT_VOLUNTARY*

   d. Arch: *CONFIG_PREEMPT*

   Ubuntu rabi *CONFIG_PREEMPT_VOLUNTARY* (*voluntary preemption*), što omogućuje jezgri preuzeti nadzor nad procesorom kad se izvršava neki proces samo onda kad to taj proces dopusti. Arch-sustav s druge strane rabi *Preemptible kernel*, što znamenuje da jezgra može preuzeti nadzor nad procesorom kad god to treba kako bi pružila procesor drugomu zadatku koji ima veći stupanj hitnosti. Ovo je još jedna od opcija koja pokazuje kako je Ubuntu sustav namijenjen svakodnevnim korisnicima, a Arch-sustav korisnicima koji rabe sustav za zahtjevnije zadatke i bitna im je responzivnost, ili se pak sustav rabi za neke potrebe u kojima je visoka responzivnost i izvedba bitna [16] [17].

6. *CONFIG_IRQ_TIME_ACCOUNTING* (Arch)

   Jedna od opcija koja je omogućena odmah na Arch-sustavu koja pokazuje da je Arch kao sustav namijenjen stručnim korisnicima i razvijateljima između ostaloga je *CPU Task time and stats accounting*. Ova opcija prati koliko traje koji prekidni zahtjev koji se pojavljuje kako bi korisnici uz pomoć *userspace*-programa mogli pratiti koliko traju IRQ-i kako bi dijagnosticirali moguće teškoće, ili pak poboljšali izvedbu sustava. [13]

7. *CONFIG_RCU_EXPERT* (Arch)*:*

   Opcija koja omogućuje da se čine naprjedne promjene na RCU-konfiguraciji, i pokazuje da je Arch kao sustav namijenjen razvijateljima, i korisnicima koji žele bolji uvid i bolji nadzor nad svojim sustavom. [18]

8. *CONFIG_IKCONFIG* (Arch*)*:

Ova postavka omogućuje da je konfiguracija jezgre vidljiva i raspoloživa u kazalu */proc*. Isto tako je jedna od postavaka koja pokazuje da je sustav namijenjen razvijateljima, ali nema izravan utjecaj na izvedbu ili potrošnju sustava [13].

9. *CONFIG_LOG_BUF_SHIFT*

    e. Ubuntu*:* 17

    f. Arch*:* 18

    Na Ubuntu-sustavu veličina međuspremnika jezgrenoga zapisnika postavljena je na *18*, a na Arch-sustavu postavljena je na *17*. *Kernel log buffer* zapisuje poruke jezgre koje se pojavljuju, i u slučaju da se zapisnik popuni najstarije poruke se zamjenjuju novima [13] [19].

    Pretpostavka izvedena iz surječja za ovu konfiguraciju je da Ubuntu rabi veću veličinu zbog veće jezgre koju Ubuntu ima naspram Arch-sustavu, s više članova i više odabranih opcija konfiguracije koje će imati potrebu pisati poruke u međuspremnik.

10. *CONFIG_PRINTK_INDEX* (Arch)

    Opcija koja omogućuje da se sve *printk*-poruke koje se zapisuju u jezgreni zapisnik indeksiraju, što dodatno pokazuje da je Arch-sustav namijenjen stručnim korisnicima, koji će možebiti imati potrebu pregledavati poruke pisane u jezgreni zapisnik, i indeksiranje im poruka omogućuje jednostavniju pretragu [20].

11. *CONFIG_PC104* (Ubuntu)

    Opcija koja pokazuje da je Ubuntu namijenjen široj javnosti koja će možebitno instalirati operacijski sustav na starije uređaje.

3.2. *Processor type and features*

1. *CONFIG_X86_EXTENDED_PLATFORM* (Ubuntu)

    Kod Ubuntu-sustava omogućena je opcija „*Support for extended (non-PC) x86 platforms*“ koja upozorava na to da je Ubuntu namijenjen za širi broj sustava, standardnih i nestandardnih [21].

2. *CONFIG_IOSF_MBI_DEBUG* (Ubuntu)

3. *CONFIG_GART_IOMMU* (Ubuntu)

    Opcija koja pruža dodatni pogonitelj za starije procesore, i iz toga se zaključuje da je Ubuntu namijenjen i za starije uređaje. [21]

4. *CONFIG_MAXSMP* (Ubuntu)
    a. *CONFIG_NR_CPUS (*Arch*)* - 320

    Postavka koja pokazuje kolik će biti najveći poduprti broj procesorskih jezgara u sustavu, i na Ubuntuu je postavljena na najveći moguć broj, a na Arch-sustavu na *320*. Postavljanje ove postavke na *320* procesorskih jezgara na Arch-sustavu Linuxovoj jezgri daje mogućnost boljega alociranja sredstava i može poboljšati dosljednost kod porabe većega broja procesorskih jezgara. U srži postavljanje vrijednosti na razuman broj procesorskih jezgara sustavu daje referentnu vrijednost po kojoj može djelotvornije alocirati sredstva [21].

5. *CONFIG_X86_MCELOG_LEGACY* (Ubuntu): Opcija omogućuje porabu staroga oruđa *mcelog daemon* koje prati i bilježi pogrješke koje se pojavljuju na sastavnicama računala [21] [22].

6. *CONFIG_X86_CPA_STATISTICS* (Arch): Opcija s nazivom „*Enable statistic for Change Page Attribute*“ koja je omogućena na Arch-distribuciji i omogućuje pregled podataka o CPA-operacijama. Pokazuje da je Arch namijenjen razvijateljima kojima je bitno imati uvid u stanja memorije [21].

7. *CONFIG_ARCH_MEMORY_PROBE* (Ubuntu): omogućuje sučelje za ispitivanje memorije u *userspaceu* [21].

8. *CONFIG_MTRR_SANITIZER_SPARE_REG_NR_DEFAULT*
    a. Ubuntu: 1
    b. Arch: 0

    Vrijednost ove postavke upućuje na broj pričuvnih MTRR-registara koje može rabiti MTRR-sanitizator u slučaju da ponestane registara. MTRR je Memory Type Range Register, dotično registar koji služi za opisivanje kako se u priručnu memoriju pohranjuju rasponi memorijskih adresa. Na Ubuntu-jezgri ta vrijednost je postavljena na *1* iz razloga što se potreba za pričuvnim registrom ovoga tipa može pokazati kod zahtjevnih zadataka i zadataka s zahtjevnim grafičkim uvjetima; s obzirom na to da je Ubuntu sveobuhvatniji sustav koji će se vjerojatnije rabiti za grafički zahtjevnije zadatke, postavljena je vrijednost na *1*. Na Arch-jezgri vrijednost je postavljena na *0* jer se pretpostavlja da se Arch-distribucija ne će rabiti za toliko zahtjevne svrhe, pa postavljanjem na *0* se smanjuje sveukupna složenost jezgre [23] [24].

9. *CONFIG_X86_KERNEL_IBT* (Arch)

Na Arch-sustavu omogućena je opcija „*Indirect branch tracking*“ koja omogućuje nadzor *control flowa* nekoga programa i sprječava napadača preuzeti nadzor nad tijekom programa izazivanjem neočekivana neizravnoga grananja. Po tom je ova opcija omogućena na Arch-sustavu iz sigurnosnih razloga [21] [25].

10. *TSX ENABLE MODE*

    a. Ubuntu: *CONFIG_X86_INTEL_TSX_MODE_OFF*

    b. Arch: *CONFIG_X86_INTEL_TSX_MODE_AUTO*

„*Transactional Synchronization Extensions*“ je postavka koja omogućuje *Transactional Memory* što je način usporednoga programiranja gdje se naredbe pisanja i čitanja odvijaju na nedjeljivi način, što sprječava dvije dretve izvesti isti zadatak na istom dijelu memorije. Tradicionalni način sprječavanja ovoga je porabom „*lock-based*“ sinkronizacije. Na Arch-sustavu TSX-postavka postavljena je kao *Auto* iz razloga što nema svaki procesor sposobnosti za TSX, nego samo neki Intelovi procesori. Isto tako ova mogućnost je i ukinuta na novim Intelovim procesorima zbog sigurnosnih teškoća. Iz navedenoga se može zaključiti da je na operacijskom sustavu Ubuntu ova postavka isključena kako bi se izbjegnulo iskorišćivanje ove opcije u zlobne svrhe. Isto tako ako TSX nije dobro ostvaren može dovesti i do pogoršavanja izvedbe umjesto poboljšavanja. Ova postavka je uključena na Arch-sustavu zato da se njom služe samo oni koji imaju dovoljno znanja da bi rabili TSX [21] [26].

11. *TIMER FREQUENCY*

    a. *Ubuntu*: *CONFIG_HZ_250*

    b. Arch: *CONFIG_HZ_300*

„*Timer Frequency*“ je vrijednost koja pokazuje koliko puta u sekundi se šalje prijekid na procesor kako bi sustav pratio vrijeme kako bi se izvršavali vremenski zakazani zadatci [27].

Učestalost brojača se proteže od 100 Hz sve do 1000 Hz. Nekim sustavima je potrebnija manja učestalost, a nekim sustavima veća učestalost. Na primjer određeni IoT-uređaji bolje rade s manjim učestalostima zato što su napajani baterijom pa su im niže učestalosti, reda 100 ili 250 Hz, bolji izbor. Sustavi kojima je bitno stvarno vrijeme, kao sustavi koji se bave audio-/videozadatcima, igrama ili trgovinom mogu se odlučiti za veću učestalost brojača. I Ubuntu i Arch u ovom slučaju rabe srednje vrijednosti učestalosti brojača, koje su dobra sredina za veći raspon zadataka. Moguće je da se je u Arch-sustavu

odlučila rabiti malo veća učestalost zbog toga jer je Arch namijenjen razvijateljima kojima je moguće potrebna malo veća učestalost.

3.3. *Power management and ACPI options*

1. *CONFIG_PM_WAKELOCKS* (Ubuntu)

   Uključivanje ove opcije na Ubuntuu omogućuje da se u „*userspaceu*“ mijenjaju izvori koji sustav bude iz stanja manje potrošnje [28].

   Ubuntu je namijenjen veliku broju korisnika, a i jedan je od najrabljenijih Linuxovih distribucija za poslužitelje, tako da velika prilagodljivost korisnicima omogućuje djelotvornije iskorišćivanje sustava. Isto tako, ova opcija je dobar način za manipuliranje potrošnjom.

2. *CONFIG_PM_ADVANCED_DEBUG* (Ubuntu)

   Postavka koja pruža mogućnosti upravljanja potrošnjom iz „*userspacea*“. [28]

3. *CONFIG_ACPI_DEBUGGER* (Ubuntu)

   Postavka koja omogućuje različna oruđa koja stvaraju ispis koristan za *debugging* teškoća s ACPI-podsustavom. ACPI, dotično „*Advanced Configuration and Power Interface*“ je podsustav koji stoji između sklopovlja i OS-a ter omogućuje bolje upravljanje potrošnjom ter manipuliranje potrošnje upravljanjem različnim uređajima koje sustav rabi [28] [29].

4. *CONFIG_ACPI_CUSTOM_METHOD* (Arch) = *m*

   Ova postavka omogućena je na Arch-sustavu kao modul jer izlaže sustav potencijalnim sigurnosnim ugrozama. To je iz razloga što ova opcija omogućuje preinaku jezgre na „živom“ sustavu. Korisnik može ručno napisati kod kojim mijenja ponašanje ACPI-podsustava i zahvaljujući ovoj opciji umetnuti taj kod u jezgru. Iz toga razloga ova opcija je omogućena na Arch-sustavu jer su korisnici Arch-sustava često razvijatelji i ispravljatelji koji znaju kako rabiti ove opcije bez izlaganja sustava ugrozi, za razliku od Ubuntu-sustava koji rabe između ostaloga i svakodnevni korisnici koji možebiti ne će imati dovoljno znanja i izložit će svoj sustav nepotrebnim ugrozama preko kojih se netko može okoristiti ovom slabošću na zloban način.

5. *CONFIG_ACPI_APEI_ERST_DEBUG* (Arch) = *m*

Opcija koja daje mogućnost sustavu pohranjivati podatke o različnim pogrješkama koje se pojavljuju u uređajima koje sustav rabi na disk, i tako daje način ispravljateljima proučavati što se događa sustavu kako bi ga mogli poboljšati ili ukloniti pogrješke [30].

3.4. *Enable Loadable Module support*

1. *CONFIG_MODULE_FORCE_LOAD* (Arch)

   Postavka koja omogućuje da se moduli jezgre ukrcaju pače i ako nije raspoloživ podatak o inačici modula koji se ukrcava [31].

   Uključivanje ove postavke može stvoriti različne teškoće na sustavu, ako se na primjer ukrca modul jezgre koji je stariji od onoga koji je potreban za izvršavanje neke radnje, on može ugroziti sustav i stvoriti nepredviđene pogrješke koje je potrebno rješavati. Na Arch-sustavu ova postavka je svejedno uključena jer se pretpostavlja da korisnik koji rabi Arch-sustav ima znanja ne izazvati ovakove pogrješke.

2. *CONFIG_MODULE_FORCE_UNLOAD* (Arch)

   Omogućivanje ove postavke je jednako pogibeljno kao i prijašnje navedene postavke. Ova postavka daje korisniku mogućnost iskrcati neki modul jezgre pače i ako se on trenutačno rabi. Ovo može uzrokovati katastrofalne kvarove, kao što su rušenje sustava ili kvarovi u jezgri [31].

3. *CONFIG_MODULE_UNLOAD_TAINT_TRACKING* (Arch)

   Ovo je značajka koja omogućuje praćenje svih iskrcanih modula jezgre koji su uzrokovali neki tip kvara. Značajka je korisna za ispravljatelje i razvijatelje, jer omogućuje bolje praćenje ponašanja sustava, i s obzirom na to da je to slučaj postavka je omogućena na Arch-sustavu [31].

4. *CONFIG_MODVERSIONS* (Ubuntu)

   Navedena postavka omogućuje porabu modula koji nisu kompilirani s jezgrom, na način da se na modul dodaju informacije koje čine modul kompatibilnim s jezgrom. S obzirom na to da se Ubuntu-sustav rabi za velik broj namjena, omogućivanje ove postavke proširuje kompatibilnost s različnim vrstama uređaja [31].

3.5. *Enable the block layer*

1. *CONFIG_BLK_DEV_THROTTLING_LOW* (Arch)

Ovo je značajka uz pomoć koje se pokušava smanjiti utjecaj prigušivanja blokovnih uređaja na *cgroupsu*, što je mehanizam za alociranje sredstava. Ovom postavkom se omogućuje da *cgroups* uvijek imaju najmanje potrebno propusnosti za učinkovit rad [32] [33].

2. *CONFIG_BLK_CGROUP_IOLATENCY* (Arch)

Mehanizam Linuxove jezgre kojim se pokušava održavati IO-kašnjenje na određenoj razini koja je konfigurirana. U slučaju da neki uređaj ima veću konfiguriranu razinu kašnjenja, na taj uređaj se primjenjuje *throttling* [32].

3.6. *Memory management options*

1. *CONFIG_CMA* (Arch)

   Neprekidna alokacija memorije je tehnika dodjeljivanja memorije u kojoj se dopušta procesima i podsustavima alokacija neprekidnih blokova memorije. Ovaj način alociranja memorije može skratiti vrijeme potrebno za pristup memoriji i omogućiti hitrije odvijanje procesa. Isto tako, porabom ove tehnike, smanjuje se vrijeme koje se troši na odvijanje sporednih nutarnjih procesa upravljanja memorijom (*overhead*). „Neneprekidna“ alokacija može biti bolji odabir kod računala gdje se rabi velika količina memorije i iskoristivost može biti bolja nego kod neprekidne alokacije [34] [35].

   Na Arch-sustavu ova postavka je vjerojatno omogućena jer je jednostavan način porabe memorije, koji je dobar za ispravljatelje i za različne raščlambe, isto tako ovaj način je djelotvorniji.

3.6.1. *Support for paging of Anonymous memory*

1. *CONFIG_ZSWAP_DEFAULT_ON* (Arch)

   Zswap je mehanizam koji daje mogućnost komprimirana *cachea* u kojem se komprimiraju stranice koje su u postupku „*swappanja*“. Ovakov mehanizam može poboljšati izvedbu sustava, i ubrzati rad s memorijom [34].

3.7. *Device Drivers*

1. *CONFIG_RAPIDIO* (Ubuntu)

   Postavka koja kod gradnje jezgre uključuje i potporu i pogonitelje za RapidIO-tehnologiju. RapidIO je arhitektura koja se rabi za packet switching, dotično organiziranje podataka u pakete i njihov prijenos među sastavnicama i mrežama. Ovo je tehnologija koja ima specifične namjene, kao u ugradbenim uređajima i telekomunikacijama. S obzirom na to da je Ubuntu-sustav namijenjen za velik broj namjena, ova postavka je omogućena na Ubuntu-sustavu jer postoji velika vjerojatnost da se Ubuntu-sustav rabi na sustavima koji to zahtijevaju [36].

2. *CONFIG_I3C* (Ubuntu)

   I3C je serijski komunikacijski protokol razvijen 2016. i nasljednikom je I2C-protokolu. Često se rabi u nekim ugradbenim uređajima, i pokretnim uređajima, kao glavni komunikacijski protokol između različnih senzora. Ubuntu je operacijski sustav koji se često rabi za ugradbene uređaje i potpora je za noviji protokol kao I3C dobra zamisao [37] [38].

3. *CONFIG_SPMI* (Ubuntu)

   System Power Management Interface je protokol koji služi za reguliranje napajanja sastavnica uređaja. Sastoji se od dvaju vodova, jednoga za takt, a drugoga za podatke. Kao i prijašnji protokoli rabe se u ugradbenim uređajima i pokretnim uređajima. [39]

4. *CONFIG_HSI* (Ubuntu)

   HSI je još jedan serijski protokol za prijenos podataka, ali ovaj je usredotočen na prijenos podataka na kratke razdaljine, u glavnom za prospajanje između najmanjih blokova (*die*) na čipu. Kao i prijašnji protokol često se rabi na pokretnim uređajima [40].

5. *CONFIG_GREYBUS* (Ubuntu)

   Greybus je protokol za prijenos podataka koji se rabi u modularnim uređajima, i ključnim je dijelom njih. Izvorno razvijen za projekt Google Ara; ali uz prkos neuspjehu projekta Ara, greybus ostaje kao mogućnost na Linuxovoj jezgri. Potpora je u Ubuntu-jezgri sama po sebi uključena [41].

6. *CONFIG_COMEDI* (Ubuntu)

   Postavka koja omogućuje potporu i Comedi-pogonitelje za uređaje za skupljanje podataka. Ovakovi uređaji mogu se rabiti u različnim IoT-uređajima gdje je potrebno

precizno skupljati podatke za raščlambe i druge svrhe. Ponovno je omogućena na Ubuntu-jezgri s obzirom da se Ubuntu rabi za IoT-uređaje [42].

7. *CONFIG_PECI* (Ubuntu)

PECI, dotično Platform Enviroment Control Interface, je jednožično sučelje za komunikaciju s vanjskim uređajima, kao na primjer uređajem za nadzor temperature procesora. Ovaj standard je specifičan za Intelove procesore [43] [44].

4. Vlastita konfiguracija

4.1. Cilj konfiguracije

Nakon usporedaba dviju konfiguracija jezgara dviju različnih distribucija Linux-sustava, i tumačenja gradnje jezgre, u ovom poglavlju kreće se s vlastitom konfiguracijom jezgre. Konfigurirat će se jezgra sustava Ubuntu, s obzirom na to da će se sustav rabiti na jednostavnom poslužitelju medija za potrebe 1 osobe. Postoji argument da je za poslužitelj medija poraba operacijskoga sustava Ubuntu Server prigodnija, ali računajući da to da se radi na malenu poslužitelju za jednu osobu, koja se lakše snalazi u slikovnom sučelju, nazočnost slikovnoga sučelja ne će imati znatne nedostatke, a nudi jednostavniji i hitriji rad s poslužiteljem kad je potrebno međudjelovanje s njim. Za velike korporativne poslužitelje, ili poslužitelje koji su pod velikom opterećenošću, nazočnost slikovnoga sučelja znamenuje nepotrebnu opterećenost na sustav, nepotrebno trošenje mjesta na disku i sustav se otvara različnim sigurnosnim ranjivostima. Kod konfiguracije jezgre za poslužitelj medija stavljat će se žarište na smanjivanje veličine slike jezgre, dakle, gradnju minimalne jezgre, koja je konfigurirana isključivo s specifikacijama i potrebama poslužitelja na prvom mjestu. Isto tako, kod opcija koje nude različne opcije s kompromisima, prioritizira se propusnost i hitrost, nasuprot na primjer responzivnosti, što se pojavljuje kod postavaka kao što je na primjer *preemption model*.

4.2. Specifikacije sustava

Kako bi se mogla učinkovito konfigurirati jezgra za poslužitelj medija koja će djelotvorno i pravodobno odrađivati svoje zadaće, ključno je razumjeti specifikacije poslužitelja na koji će se instalirati jezgra. Poslužitelj pokreće starije rabljeno kućno računalo koje je staro desetak godina, i na koje je instaliran operacijski sustav Ubuntu inačice „*Ubuntu 22.04.2 LTS x86_64*“, u središtu kojega se nahodi jezgra „*5.19.0-32-generic*“. Računalo pokreće Intelov procesor „Intel i5-4460“, koji radi s 4 jezgrama na 3.4 GHz. U sustavu je matična ploča „Gigabyte H81M-S2V“, i grafička kartica „NVIDIA GeForce GTX 750 Ti“, koja je istina vele nepotrebna za ove slučaje, s obzirom na to da se ne namišlja rabiti sklopovsko transkodiranje videa. Skladište podataka na poslužitelju događa se na tvrdom 2.5“-disku s 1 TB. U sustavu se nahodi 8 GB (2x4GB) DDR3-memorije od 1600 MT/s. Mreža poslužitelja medija konfigurirana je na način da se sve prenaša preko mreže Wi-Fi umjesto žične mreže, tako da poslužitelj rabi Wi-Fi USB-prilagodnik tp-link AC1300 Archer T3U plus.

Isto tako, za konfiguraciju jezgre kako bi sustav radio planirano, potrebno je razmotriti poslužitelj medija i s programske strane, dotično koji programi će se rabiti na sustavu. Za olakšano skladištenje podataka i datoteka ter za prijenos rečenih i povezivanje s drugim uređajima rabi se Plex. Na sustavu je isto tako pokrenut i VPN-servis, ter qBitTorrent pokrenut unutar Dockera.

4.3. Konfiguriranje

Prikaz konfiguracije jezgre će se odvijati kroz sljedeća potpoglavlja, gdje će u svakom biti prikazan jedan podizbornik unutar sučelja ncurses kconfig koje se pokreće naredbom `sudo make menuconfig` unutar direktorija u kojem se nahodi jezgra. U svakom potpoglavlju spomenut će se pojedina postavka na koju će se obratiti pozornost, i uključit će se, isključiti ili biti modificirana. Ne će biti prikazana svaka opcija, nego samo one koje će se razlikovati od početne konfiguracije. Uz svaku prikazanu postavku bit će dano tumačenje zašto i kako se ona mijenja. Ovo je osobito važno kod konfiguriranja jezgre jer ako se mijenjaju postavke o kojima ne postoji dovoljno znanja ili dobar razlog zašto se mijenja njihovo stanje, sustav se može susresti sa različnim pogrješkama, može se ponašati na neočekivane i neželjene načine, susresti s nemogućnošću u opće pokretanja sustava ili u najgorem slučaju neočekivanom prijekidu normalnoga rada sustava. Isto tako za svaku postavku za koju je to primjenjivo navest će se njezin utjecaj na jezgre. Kod nekih postavaka kad se one uključe imaju određen utjecaj na jezgre, dotično, uključivanje opcija često uzrokuje uključivanje mapa i objekata za izgradnju tih dijelova jezgre. Utjecaj na makefile prikazan je kao ispis naredbe *grep*, koja je rabljena unutar jezgre, kako bi se pronašlo koja opcija ima kakov utjecaj na . Rabljena je sljedeća naredba:

```
grep -r „$CONFIG_OPCIJA“ . –include=Makefile
```

U ovoj naredbi *-r* upozorava na to da se naredba izvodi rekurzivno, dotično da se pretražuje svako kazalo, potkazalo i datoteka iz onoga položaja u kojem se pokreće naredba. Znak točke (.) obilježava da se pretražuje i kazalo unutar kojega je naredba pokrenuta, a *–include=Makefile* znamenuje da će se pretraživati samo datoteke po nazivu *Makefile*.

4.3.1. *General Setup*

U podizborniku „*General Setup*“ nahode se u načelu, najbitnije opcije, dotično, opcije koje imaju najveći učinak na ponašanje i izvedbu sustava. Tu se nahode postavke bitne za cjelokupno ponašanje jezgre, i sve postavke koje se ne mogu pripisati pojedinim sastavnicama ili podsustavima.

Postavke kojima će se mijenjati stanje su sljedeće:

1. *Local version – append to kernel release* (*CONFIG_LOCALVERSION*)

   Navedena postavka nema nikakov izravan učinak na sustav, nego je isključivo postavka čija je svrha jednostavnije praćenje inačice jezgre koja se rabi. Ova postavka zahtijeva postavljanje niza znakova (*string*), koji će se nadodati na naziv i inačicu jezgre, koji se mogu vidjeti prigodom pokretanja naredbe kao što je *uname -r*. Zato će se uključiti i nadodati željeni *string*.

2. *Kernel Compression mode - LZO* (*CONFIG_KERNEL_LZO*)

   Odabir načina kompresije jezgre u nekim slučajevima može biti iznimno bitan, ali kod poslužitelja medija koji se rabi u kućanstvu kompresija jezgre ima manju ulogu. Kompresija jezgre u glavnom ima utjecaj na trajanje pokretanja sustava, u kojem se jezgra dekomprimira. Ovisno o tipu kompresije, pokretanje sustava može trajati dulje ako je jezgra vrlo komprimirana, ili kraće ako je jezgra manje komprimirana. Ovo ima najveću ulogu kod ugradbenih sustava, gdje je memorija ograničena, a sastavnice mogu iznimno varirati. S obzirom na to da se u ovom slučaju rabe starije sastavnice te sustav radi s dosta memorije i mjesta na disku, ter da se sustav rabi kao poslužitelj medija, na kojem se želi prioritizirati hitrost, odlučeno je da će se rabiti kompresija LZO, koja nudi nisku razinu kompresije i hitru dekompresiju [13].

   Utjecaj postavke na jezgre: postavljanje ove postavke na LZO-kompresiju uzrokuje pridruživanje vrijednosti varijabli koja određuje kompresiju jezgre. Ovo se događa za više varijabala, koje su specifične za određenu arhitekturu, ali za ovaj rad bitna je samo varijabla za x86-arhitekturu. Utjecaj na makefile izgleda ovako:

```
./arch/x86/boot/compressed/Makefile:suffix-$(CONFIG_KERNEL_LZO) := lzo
```

   Ovaj redak koda upućuje na to da se na varijablu *suffix-y* (*y* s obzirom na to da je uključena postavka *CONFIG_KERNEL_LZO*) pridružuje vrijednost *lzo*.

3. *Timer tick handling – Tickless Idle* (*CONFIG_HZ_PERIODIC*)

   Timer tick handling je jedna od znatnijih postavaka za poslužitelj medija, i općenito postavka koja ima popriličan utjecaj na sustav. Kod postavke *Timer Tick Handling* bira se način na koji će jezgra pratiti koji se procesi moraju i trebaju izvršiti, dotično, koliko često će se obavljati pregledi postoji li neki zadatak koji treba dobiti pozornost. S obzirom na to da se radi o poslužitelju medija koji na sebi ima pokrenute brojne usluge, sustav

može imati neke blagodati od porabe brojača, kako bi se možebitno povećala responzivnost i smanjilo kašnjenje. Isto tako jer se radi o poslužitelju koji rabi u glavnom jedna osoba, i nije često opterećen, procesor češće može ići u stanja kad ne čini ništa. Zato će se rabiti „*Tickless Idle*“, postavka koja omogućuje porabu brojača kad je procesor pod opterećenošću, i neporabu periodičnoga brojača kad je procesor u stanju mirovanja. Ovim se može postignuti neko, iako maleno, smanjivanje potrošnje energije.

4. *Preemption model – No Forced Preemption* (*CONFIG_PREEMPT_NONE*)

   Kod biranja *preemption modela* suprotstavljaju se responzivnost i propusnost. *Preemption model* u odnošaju je s tim kad i kako će se procesi i zadatci izvesti, dotično hoće li se privremeno zaustaviti odvijanje jednoga procesa kako bi se odvio proces veće bitnosti. Kod opcije „*No Forced preemption*“ operacijski sustav minimizira prijekide procesa koji se trenutačno odvija u svrhu odvijanja drugoga procesa, tim povećavajući propusnost sustava. Dakle ako to apsolutno nije potrebno, proces koji se odvija u jezgri dovršit će se do kraja, i ne će biti prekinut drugim procesom veće bitnosti [16].

5. *Core Scheduling for SMT* (*CONFIG_SCHED_CORE*)

   SMT dotično Simultaneous Multithreading (*Hyperthreading* u Intelovim procesorima) je nazočan u gotovo svim modernim procesorima, i povećava djelotvornost procesorskih jezgara, na način da jednu procesorsku jezgru dijeli na dva prividna dijela tako da se izvršavaju dvije dretve na jednoj procesorskoj jezgri. Core Scheduling je djelovanje koje dalje poboljšava učinkovitost SMT-a odlučivanjem koji procesi mogu dijeliti jezgru, i koji će se procesi odvijati na kojoj dretvi. Isključivanje ove postavke ima više nedostataka nego prednosti u odnosu na uključivanje ove opcije, što može poboljšati izvedbu sustava, stoga je u ovoj konfiguraciji uključena [45] [46].

6. *Kernel .config support* (*CONFIG_IKCONFIG*)

   IKCONFIG je konfiguracijska opcija koja efektivno nema utjecaj na ponašanje sustava ni njegovu izvedbu, nego jednostavno omogućuje korisniku sustava imati pristup datoteki konfiguracije žive jezgre, što s obzirom na to da postoji mogućnost da će se konfiguracija dalje proučavati i preinačivati vrijedi uključiti.

   Utjecaj opcije na jezgre:

   ```
   .kernel/Makefile:obj-$(CONFIG_IKCONFIG) += configs.o
   ```

   Navedeni ispisni tekst pokazuje da se u izgradnju jezgre, za kompilaciju djelovanja *CONFIG_IKCONFIG*, uključuje objekt *configs.o*. Iz nalaza naredbe isto tako može se iščitati da se ovo događa u datoteki makefile koja se nahodi na stazi *./kernel/Makefile*.

7. *Kernel Log Buffer Size* (*CONFIG_LOG_BUF_SHIFT*)

   Za vrijeme rada sustava jezgra obavlja brojne zadatke i kod većine zadataka postoje ispisne informacije koje mogu biti korisne upraviteljima sustava iz više razloga, najviše *debugginga*. Ispisni podatci mogu biti pogrješke, stanja sastavnica, dijagnostički podatci, podatci stvoreni kod dizanja sustava, statični podatci (kao obavijesti o sklopovlju i pogoniteljima), i ostali možebitno korisni podatci. Sve ove obavijesti se pohranjuju u međuspremnik dnevnika jezgre, to jest *kernel log buffer*. S obzirom na to da su ovi podatci privremeni i se mogu zapisati samo onoliko koliko je memorije dodijeljeno tomu dnevniku, nakon čega se stari podatci brišu te zapisuju novi, i skladište se u radnoj memoriji sustava, veličina ovoga dnevnika može imati utjecaj na izvedbu sustava. Što je više mjesta u radnoj memoriji alocirano međuspremniku za dnevnik jezgre, to postoji manje radne memorije za ostale procese. Poslužitelj medija koji je tema ovoga rada rabi razmjerno malo radne memorije, te se ne rabi za previše zadataka, nego najčešće samo par zadataka istodobno, pa će veličina dnevnika biti postavljena na nižu vrijednost, što će biti *16* [19].

4.3.2. *Processor type and features*

Postavke koje se nahode u izborniku *Processor type and features* su bitne za specificiranje procesora koji će sustav rabiti, i kako će se on ponašati u određenim situacijama. U ovom podizborniku odabrat će se samo one ključne opcije, kako bi se smanjila veličina završne jezgre; početno, u ovom podizborniku većina postavaka je uključena kako bi se omogućilo da se operacijski sustav može rabiti na svakom zastupljenijem procesoru, čim se izbjegava potreba da korisnik konfigurira jezgru, tim povećavajući broj mogućih korisnika Linuxova sustava. Ali s obzirom na to da se u ovom radu konfigurira jezgra, bilo bi mudro obratiti pozornost na isključivanje nepotrebnih opcija.

U nastavku navedene su sve postavke koje će biti isključene, a ako se koja postavka posebno uključuje to će biti napomenuto:

1. *Symmetric multi-processing support* (*CONFIG_SMP*)

   Postavka koja se rabi isključivo na sustavima s više od jednoga procesora, s tim će na ovoj instanciji jezgre biti isključena kako bi se smanjila veličina krajnje jezgre.

   Utjecaj na makefile: S obzirom na to da je postavka specifična za arhitekturu sustava, ponovno se obraća pozornost samo na utjecajne , koje su u odnošaju s x86-arhitekturom. S obzirom na to da se postavka u ovoj instanciji jezgre isključuje, utjecaj na makefile nije

znatan, ali kad bi postavka bila uključena, u kompilaciju jezgre bi se uključivao veći broj objekata kao što su: *smp.o*, *smpboot.o*, *setup_percpu.o*, *ipi.o*, *msr-smp.o*, *cahce-smp.o*.

2. *Support for extended x86 platforms* (*CONFIG_X86_EXTENDED_PLATFORM*)

   Postavka koja se rabi za malen broj sustava, za posebne potrebe, što poslužitelj medija u ovom radu nije, pa će se postavka isključiti.

3. *Intel Low Power Subsystem Support* (*CONFIG_x86_INTEL_LPSS*)
4. *AMD ACPI2Platform devices Support* (*CONFIG_X86_AMD_PLATFORM_DEVICE*)
5. *Enable IOSF sideband access through debugfs* (*CONFIG_IOSF_MBI_DEBUG*)
6. *Linux guest support* (*CONFIG_HYPERVISOR_GUEST*)

   Postavka ima utjecaj na , i njezinim se isključivanjem isključuju objekti *vmware.o*, *hypervisor.o* i *mshyperv.o*.

7. *Support Hygon processors* (*CONFIG_CPU_SUP_HYGON*)

   Isključivanjem postavke iz izgradnje jezgre isključuje se *hygon.o*.

8. *Support Centaur processors* (*CONFIG_CPU_SUP_CENTAUR*)

   Isključivanje opcije isključuje iz izgradnje jezgre cijelo kazalo *zhaoxin* i objekt *centaur.o*. Kod isključivanja ove postavke pojavljuje se prikladan primjer različnih načina rada makefile-jezika za izgradnju jezgre. Osim uključivanja objekata, odnosno datoteka C koje se kompiliraju, može se uključiti i cijelo kazalo kao što je slučaj ovdje. Ovo kazalo može imati dodatne datoteke makefile i programske datoteke. Ovo je u datoteki makefile prikazano na sljedeći način:

   ```
   obj-$(CONFIG_CPU_SUP_CENTAUR) += zhaoxin/
   ```

9. *Support Zhaoxin processors* (*CONFIG_CPU_SUP_ZHAOXIN*)

   Isključivanje ove opcije u datoteki makefile isto tako isključuje kazalo *zhaoxin*, ali ovaj put isključuje se i objekt *zhaoxin.o*.

10. *Old AMD GART IOMMU support* (*CONFIG_GART_IOMMU*)

    Isključivanje postavke za potporu AMD GART IOMMU isključuje objekte *amd_gart_64.o* i *aperture_64.o*.

11. *5-level page tables support* (*CONFIG_x86_5LEVEL*)

4.3.3. *Power management and ACPI options*

Poslužitelji zbog potrebe za niskim kašnjenjem, visokom responzivnošću i zbog velika kapaciteta korisnika veliku većinu vremena provode uključeni, ter je neisplativo i nepogodno rabiti ikakova stanja ACPI S poput hibernacije ili suspendiranja na RAM. Poslužitelj medija za kakova se konfigurira jezgra je razmjerno rijetko u porabi naspram više opterećenim korporativnim poslužiteljima, i u teoriji, S-stanja bi bila korisna i omogućila znatnu uštedu energije. Teškoća se prikazuje u tom da S-stanja kao takova zahtijevaju izvor „buđenja“, kao što je na primjer pritisak tipke na tipkovnici, što kod poslužitelja medija nije moguće. Ipak postoji i opcija porabe WoL-značajke, dotično „*Wakeup on LAN*“, gdje se slanjem takozvana „magičnoga paketa“ sustav budi, što je bez dodatnoga uređaja malene potrošnje kao na primjer Raspberry Pi iznimno teško izvedivo. U budućnosti porabe poslužitelja medija, ako se za to pokaže potreba, ovakov sustav se može ostvariti, ali do tad prepušten je alternativama, iako manje učinkovitima. Jedna od tih alternativa je „*Frequency Scaling*“ gdje se ovisno o zauzetosti procesora smanjuje njegov takt rada.

Poglavljem ozgora opisano je zašto se u izborniku za upravljanje potrošnjom i ACPI-opcijama isključuje kakova bilo mogućnost S-stanja, i uključuju se samo opcije za skaliranje takta procesora. Tako su u nastavku navedene sve opcije koje se uključuju, isključuju ili preinačuju:

1. *Suspend to RAM and standby* (*CONFIG_SUSPEND*)

   Iako je u odlomku ozgora navedeno da su opcije poput suspendiranja na radnu memoriju složene za ostvaraj, opcija suspendiranja na radnu memoriju ostat će uključena u slučaju da se u budućnosti pokaže potreba ostvaraja sustava koja će to omogućiti. S obzirom na to da isključivanje ove opcije ne će imati utjecaj na učinkovitost i izvedbu sustava, i ne će biti znatne uštede mjesta na disku, ne škodi ju ostaviti uključenu kako bi se u budućnosti možebitno smanjilo vrijeme uključivanja ove opcije i ponovne izgradnje jezgre ako se za nju pokaže potreba.

   U datoteka makefilema uključivanje ove postavke uključuje nekoliko objekata za različne arhitekture, ali ni jedna nije utjecajna x86-arhitektura. Jedino utjecajno uključivanje je u datoteki makefile u stazi *./kernel/power* gdje se uključuje objekt *suspend.o* u uobičajenu varijablu *obj-y*.

2. *Hibernation (aka 'suspend to disk')* (*CONFIG_HIBERNATION*)

   Za razliku od prijašnje opcije ova opcija će biti isključena, jer i ako se u poslužitelju medija ostvari mogućnost suspendiranja ili hibernacije, cijena hibernacija na disk je za sustav

kao poslužitelj medija prevelika, dotično, potrebno je previše vremena da bi se sustav ponovno pokrenuo iz ovoga stanja.

Isključivanje ove postavke u datoteka makefilema nosi promjene ovisne o arhitekturi sustava ali što se tiče x86-arhitekture uključuju se sljedeći objekti: *hibernate_$(BITS).o*, *hibernate_asm_$(BITS).o*, i *hibernate.o*. Isto tako unutar u kazalu *./kernel/power* uključuje se *hibernate.o*, *snapshot.o* i *swap.o*.

3. *Userspace snapshot device* (*CONFIG_HIBERNATION_SNAPSHOT_DEV*)

   Opcija se isključuje s obzirom na to da se rabi samo kad je u porabi hibernacija. Unutar ona isključuje objekt *user.o*.

4.3.4. *Device Drivers*

Podizbornik *device drivers* je jedan od podizbornika koji u jezgri Ubuntu-sustava ima uključeno puno zališnih postavaka kako bi se povećala suradljivost s što više drugih sustava i uređaja bez potrebe da korisnici koji će rabiti druge sustave i uređaje sami konfiguriraju jezgru. S obzirom na to isključivanjem puno opcija iz ovoga podizbornika može se nešto smanjiti veličina gotove jezgre zbog čega zauzvrat može se skratiti vrijeme dekomprimiranja jezgre. Za utjecaj ovih opcija na ne će se navoditi staze svih datoteka makefile, nego samo isključeni objekti, osim ako je potrebno dodatno tumačenje. U nastavku su navedene sve opcije koje će se isključiti:

1. *EISA support* (*CONFIG_EISA*)

   Opcija unutar datoteka makefile isključuje kazalo *eisa/*, i objekte *eisa-bus.o*, *pci_eisa.o*, *virtual_root.o*, *eisa.o* (jednom u *.drivers/parisc/Makefile*, i jednom u *./arch/x86/kernel/Makefile*; objekt *eisa.o* koji se uključuje u *Makefileu* za x86-arhitekturu, za druge arhitekture taj se objekt razlikuje), *eisa_enumerator.o*, *eisa_eeprom.o*, *aic7770.o* i *aic7770_osm.o*.

2. *GNSS receiver support* (*CONFIG_GNSS*)

   U datoteka makefilema isključuju se objekti *gnss.o* i *ice_gnss.o* ter kazalo *gnss/*.

3. *Macintosh device drivers* (*CONFIG_MACINTOSH_DRIVERS*)

4. *I3C support* (*CONFIG_I3C*)

   U datoteka makefilema isključuje se objekt *i3c.o* i kazalo *master/*.

5. *Eckelmann SIOX Support* (*CONFIG_SIOX*)

   U datoteka makefilema isključuje se kazalo *siox/* i objekt *siox-core.o*.

6. *MCB support* (*CONFIG_MCB*)

   U datoteka makefilema isključivanjem ove opcije isključuje se kazalo *mcb/* i objekt *mcb.o*.

7. *IndustryPack bus support* (*CONFIG_IPACK_BUS*)

   U datoteka makefilema isključuje se kazalo *ipack/* i objekt *ipack.o*.

8. *SoundWire support* (*CONFIG_SOUNDWIRE*)

   U datoteka makefilema isključuje se kazalo *soundwire/* i objekt *soundwire-bus.o*.

9. *Platform support for Mellanox hardware* (*CONFIG_MELLANOX_PLATFORM*)

   U datoteka makefilema isključivanjem ove opcije isključuje se samo kazalo *mellanox/*.

10. *Platform support for Chrome hardware* (*CONFIG_CHROME_PLATFORMS*)

    Kao i za prijašnju opciju u datoteka makefilema isključuje se samo kazalo *chrome/*.

11. *Data acquisition support (comedi)* (*CONFIG_COMEDI*)

    Isključivanjem ove postavke u Datoteka makefilema isključuju se comedi/, kcomedilib/ i drivers/ (unutar datoteke ./drivers/comedi/Makefile) mape te objekt comedi.o.

12. *Greybus support* (*CONFIG_ GREYBUS*)

    Isključivanjem ove opcije u datotekama *./drivers/Makefile* i *.drivers/staging/Makefile* isključuje se kazalo *greybus/*.

13. *InfiniBand support* (*CONFIG_INFINIBAND*)

    Isključivanjem ove opcije u datoteka makefilema isključuje se kazalo *infiniband/* te objekt *libiscsi.o*.

14. *Sony MemoryStick card support* (*CONFIG_MEMSTICK*)

    Isključivanje ove opcije u datoteka makefilema isključuje kazalo *memstick/*.

15. *Universal Flash Storage Controller* (*CONFIG_SCSI_UFSHCD*)

    Isključivanje ove opcije u datoteka makefilema isključuje kazala *core/* i *host/*.

16. *Multiple devices driver support (RAID and LVM)* (*CONFIG_MD*)

    Ovo je jedna od opcija na koju će se obratiti pozornost da ostane uključena. Bez ove opcije nije moguće postaviti RAID-tehnologiju pohrane podataka, što iako se trenutačno ne rabi u poslužitelju medija, s obzirom na to da se radi o poslužitelju, na koji će se stavljati sve više i više podataka, koji možebitno mogu postati i bitni, postoji mogućnost da će se ova tehnologija ostvariti.

Isto tako, uključivanje ove opcije u datoteka makefilema uključuje kazalo *md/*, i u datoteki makefile na kazalima *./drivers/net/* i *./drivers/md/* uključuje poveći broj objekata ovisno o metodi RAID-a koji će se rabiti, koji se ovdje ne će navoditi.

4.3.5. *File systems*

File systems je još jedan od izbornika u koji je uračunano puno zalihosti, s obzirom na to da ni jedan datotečni sustav nije većinski prevladavajuć u sustavima, nego različni sustavi zahtijevaju drugačije datotečne sustave, bilo to iz tehničkih razloga ili zbog osobnih preferencija korisnika. U ovom podizborniku će se iz toga razloga uključiti samo oni datotečni sustavi koji će se rabiti, a zališni će se isključiti. Poslužitelj medija za koji se jezgra gradi će osnovno rabiti *ext4 filesystem*, ali ostavit će se uključene i sve ostale opcije koje bi mogle biti rabljene u slučajima drugačije formatiranih uklonjivih medija. U nastavku se navode sve opcije koje će biti isključene.

1. *Reiserfs support (deprecated)* (*CONFIG_REISERFS_FS*)

Isključivanjem ove opcije u datoteka makefilema se isključuje kazalo *reiserfs/*.

2. *JFS filesystem support* (*CONFIG_JFS_FS*)

Isključivanjem ove opcije u datoteka makefilema isključuje se kazalo *jfs/*.

3. *GFS2 file system support* (*CONFIG_GFS2_FS*)

Kao i kod prijašnjih u datoteka makefilema isključivanjem ove opcije isključuje se mapa *gfs2/*.

4. *OCFS2 file system support* (*CONFIG_OCFS2_FS*)

Isključivanjem ove opcije u datoteka makefilema isključuje se kazalo *ocfs2/*.

5. *NILFS2 file system support* (*CONFIG_NILFS2_FS*)

Isključivanjem ove opcije u datoteka makefilema isključuje se kazalo *nilfs2/*.

6. *F2FS filesystem support* (*CONFIG_F2FS_FS*)

Isključivanjem ove opcije u datoteka makefilema isključuje se kazalo *f2fs/*.

7. *File system based Direct Access (DAX) support* (*CONFIG_FS_DAX*)

Isklјčivanjem ove opcije u datoteka makefilema isključuju se objekti *dax.o* i *xfs_notify_failure.o*.

8. *Overlay filesystem support* (*CONFIG_OVERLAY_FS*)

Isključivanjem ove opcije u datoteka makefilema isključuje se kazalo *overlayfs/*.

4.4. Konfiguracija u brojevima

Za potrebe razumijevanja, korisno je navesti koliko ukupno objekata, kazala i opcija uključuje, isključuje ili u slučaju opcija i modificira.

Nova konfiguracija razlikuje se od bazne na sljedeći način:

- Uključeno: 5 opcija, 2 kazala i minimalno 2 objekta;
- Isključeno: 36 opcija, 25 kazala i minimalno 39 objekata;
- Modificirano: 4 opcije.

5. Proces izgradnje jezgre

U prijašnjim poglavljima istumačeni su koraci konfiguracije i izgradnje jezgre zbog boljega razumijevanja, ali u ovom poglavlju svi ti koraci će biti stavljeni u surječje kako bi se precizno prikazao postupak konfiguracije i izgradnje jezgre za ovaj rad. Dobro je napomenuti da postupak prikazan u nastavku nije sveopće primjenjiv preko svih distribucija i sustava nego je specificiran i prilagođen za sustav koji se obrađuje u radu, iako može poslužiti kao predložak za druge konfiguracije.

Nakon pohrane konfiguracije koja je namještena unutar ncurces-sučelja pod nazivom *.config* potrebno je jezgru izgraditi. Kako bi se jezgra izgradila potrebno je utvrditi jesu li instalirana sva potrebna oruđa za izgradnju jezgre. Kako bi se ovo izvelo, u terminalu se pokreću sljedeće naredbe:

```
sudo apt-get install libelf-dev libssl-dev
```

Paket *libelf-dev* je paket koji ima knjižnice za rukovanje datoteka ELF i njihov razvoj, a *libssl-dev* je paket koji ima oruđa za rad s SSL-protokolom. [47] [48]

Oba paketa su potrebna za izgradnje inačice jezgre 6.9.4, i potrebno ih je instalirati kako bi se jezgra uspješno izgradila. U slučaju da jedan od ovih dvaju paketa nedostaje pojavljuje se pogrješka koja upozorava na nepostojanje zaglavlja i knjižnica. Na drugim sustavima i inačicama jezgre isto tako je moguće da će se pojaviti pogrješka koja upozorava da nedostaje koji drugi paket; u takovim situacijama iščitava se iz pogrješke što nedostaje, i traži se odgovarajući paket za instalaciju kako bi se jezgra mogla izgraditi.

Nakon instaliranja potrebnih paketa moguće je pokrenuti naredbu make, koja u slučaju ovoga sustava izgleda kao *make -jx*. Broj „*x*“ u ovoj naredbi daje oruđu make uputu koliko procesorskih jezgara može rabiti za izgradnju slike jezgre kako bi se postupak hitrije odvio. Dakle make-naredba rekurzivno prohodi kroz ter kompilira sve opcije i izgrađuje sliku jezgre. Nakon pokretanja naredbe dođe do poruke pogrješke koja glasi:

```
make[1]: *** [/usr/src/linux-6.9.4/Makefile:1919: .] Error 2

make: *** [Makefile:240: __sub-make] Error 2
```

Promatrajući samo ozgora navedene poruke pogrješke teško je doći do zaključaka što je pošlo po krivu, a, s obzirom na to da se je kod izvodio neko vrijeme nakon pokretanja naredbe, isto tako je nepraktično pregledavati sve ispisne retke kako bi bilo vidljivo gdje je pošlo po krivu. Kako bi se dijagnosticiralo gdje se pojavljuje problem, zbog načina na koji make-jezik radi, moguće je samo ponovno pokrenuti naredbu make, i izgrađivat će se samo dijelovi koji se još

nisu, tako da bi se u teoriji pokretanjem naredbe make hitro trebala vidljivo prikazati specifičnija pogrješka. Nakon ponovnoga pokretanja odmah na početku ispisnoga teksta naredbe pokazuje se sljedeća pogrješka:

```
make[3]: *** No rule to make target ‘debian/canonical-certs.pem’,
        needed by 'certs/x509_certificate_list'. Stop.
```

Iz navedenoga može se zaključiti da do pogrješke dođe kod popisa x509-potvrdnica. Kako bi se ovaj problem riješio dovoljno je pokrenuti sljedeće naredbe:

```
scripts/config –disable SYSTEM_TRUSTED_KEYS

scripts/config –disable SYSTEM_REVOCATION_KEYS
```
[49]

Ove naredbe isključuju popise koji imaju pouzdane i nepouzdane ključeve. Koliko je znano, ovo je teškoća koja se pojavljuje kod izgradnje jezgara 5.9 i dalje, ali isključivanje ovih popisa ključeva u slučaju ovoga sustava nema negativnih nuspojava.

U nekim situacijama ako paket nije prije instaliran potrebno je isto tako instalirati lzop za tip konfiguracije koja rabi LZO-kompresiju jezgre, kako bi se izgradnja i kompresija mogla izvršiti. Kako bi se paket instalirao potrebno je pokrenuti sljedeću naredbu:

```
sudo apt-get install lzop
```

Nakon što su sve navedene naredbe izvršene, i svi potrebni paketi instalirani, može se započeti s postupkom izgradnje i instalacije. Kao što je navedeno, kako bi se izgradila slika jezgre, potrebno je pokrenuti naredbu make s zastavicom broja procesora koji se žele rabiti. U kontekstu ovoga sustava naredba izgleda ovako:

```
sudo make -j2
```

Kad je izvršavanje naredbe make dovršeno, pripravna je slika jezgre, dotično datoteke *bzImage* i *vmlinux* (u načelu se stvara još dodatnih datoteka, ali *bzImage* i *vmlinux* su najbitniji za potrebe ovoga rada).

Sljedeći korak u postupku izgradnje jezgre jest izgradnja modula, što se izvršava sljedećom naredbom:

```
sudo make modules_install
```

Nakon izvršavanja naredbe navedene ozdola, sljedeći korak je instalacija same jezgre u sustav, što se izvršava naredbom:

```
sudo make install
```

Kada se naredba završi izvršavati, u sustavu je instalirana novokonfigurirana jezgra, što je u slučaju jezgre ovoga rada jezgra 6.9.

Sljedeći korak instalacije često se odvija kao dio naredbe *make install*, ali za dodatnu sigurnost kako bi se izbjegao *kernel panic*, korisno je pokrenuti sljedeće naredbe:

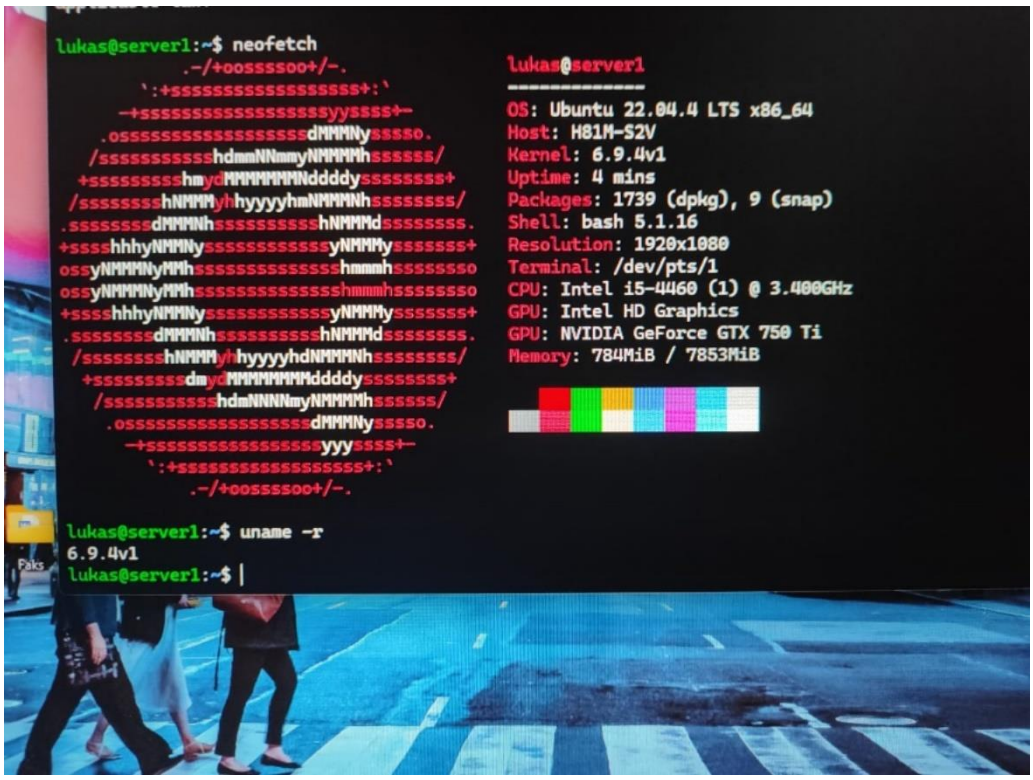
```
sudo update-initramfs -c -k 6.9.4v1

sudo update-grub
```

[1]

Naredba update-initramfs služi za osvježavanje i stvaranje *initramfsa*, dotično privremenoga datotečnoga sustava *ram*, koji se rabi kod pokretanja sustava, i služi za montiranje datotečnoga sustava, ukrcavanje ključnih jezgrenih modula, i ostalih zadataka koji su potrebni za uspješno pokretanje Linuxova sustava. Za pokretanje naredbe za osvježavanje *initramfsa* rabljene su opcije "*-c*" i "*-k*", koje služe za stvaranje novoga initramfs-sustava, dotično za specifikaciju verzije jezgre koja se rabi u sustavu,. Sljedeća rabljena naredba je update-grub, koja služi za osvježavanje gruba, odnosno *bootloadera*. Kad su sve naredbe izvršene, sigurno je ponovno pokrenuti sustav naredbom *reboot* [1].

Nakon ponovnoga pokretanja sustava dobro je potvrditi instalaciju nove jezgre, pokretanjem naredbe *uname -r* koja pokazuje inačicu i naziv jezgre koja je instalirana. U slučaju da se prikaže inačica jezgre koja je instalirana i tekst koji je upisan pod opcijom *CONFIG_LOCALVERSION*, nova konfiguracija jezgre je uspješno instalirana. Ovaj korak prikazan je na fotografiji ispod.



Slika 4. Prikaz instalirane konfiguracije na stvarnu sustavu

Naravno kad je nova jezgra instalirana korisno je vidjeti utjecaj konfiguracije na jezgru. Rabeći sljedeću naredbu može se promotriti veličina slike jezgre:

```
ls -lha /boot | grep $(uname -r)
```

Pokretanje navedene naredbe ispisuje sve što se nahodi u kazalu *boot* i imade u svojem nazivu ono što ispisuje *uname -r*. Ovu naredbu je potrebno pokrenuti prije instalacije i nakon instalacije nove jezgre. Nakon pokretanja na operacijskom sustavu prije instalacije nove jezgre veličina datoteke *vmlinuz* prikazana je kao 16 MB, a nakon instalacije nove jezgre prikazuje se 14 MB. Iako se ovo ne čini kao znatno smanjivanje, dobro je napomenuti da se konfiguraciji jezgre nije pristupalo s stavom apsolutne smanjenosti, nego se je konfiguraciji pristupalo s stavom isključivanja nepotrebnoga. Isto tako još jedan od dodatnih razloga zašto se smanjivanje ne čini tako znatnim je to što se je za kompresiju jezgre rabio algoritam LZO koji ima manji omjer kompresije nego što ima osnovna opcija na Ubuntu što je ZSTD.

6. Zaključak

Linuxova jezgra je iznimno složena i nalaz je neprekidnoga razvoja već više od 30 godina, kojemu su prinašali ljudi cijeloga svijeta. Zbog takova dugogodišnjega i suradnoga razvoja i njezine naravi *open-source* Linux jezgra je krajnje modularna i primjenjiva u veliku broju tržišta, od pokretnih uređaja, poslužitelja, osobnih računala, do integriranih i IoT-uređaja. Zahvaljujući toj modularnosti, postoji mogućnost za stotine Linux distribucija, s različnim namjenama i poslanstvima, ter svatko može i stvoriti vlastitu distribuciju ili izvod nekih od prijašnjih distribucija.

Tako je u ovom radu općenito opisano stvaranje Linuxovih jezgara, od preuzimanja jezgre, konfiguriranja jezgre do instalacije jezgre. Isto tako je opisana teorija i semantika Kconfiga i Kbuilda, ter kako zajedno rade i preko kojih oruđa se njima upravlja. Nakon toga, specifično su prikazane opcije konfiguracije jezgre na način da su se usporedile dvije popularne konfiguracije, Ubuntu i Arch, čim su se pobliže prikazale važne opcije i njihove primjene i svrhe.

S novim razumijevanjem opcija konfiguracije i sustava za izgradnju konfigurirana je i izgrađena vlastita jezgra, s svrhom za porabu u malenu kućnom poslužitelju medija. Kroz rad zaključuje se kako je konfiguracija jezgre kroz godine se pretvorila u iznimno jednostavan postupak s malo potrebnih koraka, i mogućnost se otvara svakomu tko želi naučiti ili već ima razumijevanja. Isto tako, prikazuje se i koliko podataka o sustavu promatranje konfiguracije znamenite jezgre može reći o završnom sustavu, i kako zbog spomenute modularnosti samo konfiguriranje sveopće jezgre može dovesti do dvaju iznimno različitih sustava, kao što je Ubuntu s širokom primjenom ili s druge strane Arch kao sustav za zanesenjake.

7. Literatura

[1] G. Jevtić, „How to Build Linux Kernel From Scratch {Step-By-Step Guide}“, phoenixNAP, 12. studenog 2020. [Na Internetuu]. Dostupno: https://phoenixnap.com/kb/build-linux-kernel. [Pristupljeno: 8. studenog 2023.].

[2] H. Vaghela, "Obtain Kernel Config from Currently Running Linux System," Baeldung, 22. studenog 2022. [Na Internetuu]. Dostupno: https://www.baeldung.com/linux/kernel-config. [Pristupljeno: 8. studenog 2023.].

[3] M. E. Chastain, "Linux Kernel Makefiles," kernel.org. [Na Internetu]. Dostupno: https://www.kernel.org/doc/Documentation/kbuild/makefiles.txt. [Pristupljeno: 1. prosinca 2023].

[4] L. Reynolds, "Linux kernel configuration," linuxconfig.org, 7. veljače 2022. [Na Internetu]. Dostupno: https://linuxconfig.org/in-depth-howto-on-linux-kernel-configuration. [Pristupljeno: 9. studenog 2023].

[5] A. Y. Ogun, "What is Makefile and make? How do we use it?," Medium, 1. svibnja 2022. [Na Internetu]. Dostupno: https://medium.com/@ayogun/what-is-makefile-and-make-how-do-we-use-it-3828f2ee8cb. [Pristupljeno: 2. prosinca 2023].

[6] S. Patil, "What is a Makefile and how does it work?," opensource.com, 22. kolovozaa 2018. [Na Internetu]. Dostupno: https://opensource.com/article/18/8/what-how-makefile. [Pristupljeno: 2. prosinca 2023].

[7] Baeldung, „Differences Between vmlinux, vmlinuz, vmlinux.bin, zimage, and bzImage“, Baeldung, 11. srpnja 2023. [Na Internetu]. Dostupno: https://www.baeldung.com/linux/kernel-images.

[8] J. Martine, "Kbuild: the Linux Kernel Build System," Linux Journal, 26. prosinca 2012. [Na Internetu]. Dostupno: https://www.linuxjournal.com/content/kbuild-linux-kernel-build-system. [Pristupljeno: 1. prosinca 2023].

[9] C. Jin, "Exploring the Linux kernel: The secrets of Kconfig/kbuild," opensource.com, 11. listopadaa 2018. [Na Internetu]. Dostupno: https://opensource.com/article/18/10/kbuild-and-kconfig. [Pristupljeno: 1. prosinca 2023].

[10] "6.5 Setting Variables," gnu.org. [Na Internetu]. Dostupno: https://www.gnu.org/software/make/manual/html_node/Setting.html#Setting. [Pristupljeno: 6. prosinca 2023].

[11] N. B, "What is the difference between := and += in make file?," stackoverflow, 19. travnja 2012. [Na Internetu]. Dostupno: https://stackoverflow.com/questions/10227598/what-is-the-difference-between-and-in-make-file. [Pristupljeno: 6. prosinca 2023].

[12] "Introduction to Linux Kernel Modules and Kbuild," The Linux Foundation, 2018. [Na Internetu]. Dostupno: https://cm.e-ale.org/2018/modules-kbuild/kernel-modules-and-kbuild.pdf. [Pristupljeno: 1. prosinca 2023].

[13] Linux Developeri, „linux/init/Kconfig“, GitHub, 2023. [Na Internetu]. Dostupno: https://github.com/torvalds/linux/blob/master/init/Kconfig [Pristupljeno: 20. siječnja 2024].

[14] mreff555, “Timer Tick handling”, LinuxQuestions.org, 4. srpnja 2013. [Na Internetuu]. Dostupno: https://www.linuxquestions.org/questions/linux-kernel-70/timer-tick-handling-4175468487/ [Pristupljeno: 20. siječnja 2024.].

[15] "eBPF Documentation," eBPF. [Na Internetu]. Dostupno: https://ebpf.io/what-is-ebpf/. [Pristupljeno: 28. siječnja 2024].

[16] Linux Developeri, „linux/kernel/Kconfig.preempt“, GitHub, 2023. [Na Internetu] Dostupno: https://github.com/torvalds/linux/blob/a38297e3/kernel/Kconfig.preempt [Pristupljeno: 28. siječnja 2024.].

[17] Madhurihammad, “Difference between Preemptive and Non-Preemptive Kernel in OS”, GeeksForGeeks, 2. Svibnja 2023. [Na Na Internetuu] Dostupno: https://www.geeksforgeeks.org/difference-between-preemptive-and-non-preemptive-kernel-in-os/ [Pristupljeno: 30. Siječnja 2024]

[18] Linux Developeri, “linux/kernel/rcu/Kconfig”, GitHub, 2023. [Na Internetu] Dostupno: https://github.com/torvalds/linux/blob/a38297e3/kernel/rcu/Kconfig [Pristupljeno: 2. veljače 2024]

[19] Burak Gökmen, „How to Add Messages to the Kernel Buffer“, Baeldung, 18. Ožujka 2023. [Na Internetu] Dostupno: https://www.baeldung.com/linux/kernel-buffer-add-messages [Pristupljeno: 2. veljače 2024]

[20] “Message logging with printk”, The Linux Kernel. [Na Internetu] Dostupno: https://docs.kernel.org/core-api/printk-basics.html [Pristupljeno: 2. valjače 2024]

[21] Linux Developeri, „linux/arch/x86/Kconfig“, GitHub, 2023. [Na Internetu] Dostupno: https://github.com/torvalds/linux/blob/a38297e3/arch/x86/Kconfig [Pristupljeno: 5. valjače 2024]

[22] “code README”, mcelog. [Na Internetu] Dostupno: https://mcelog.org/README.html [Pristupljeno: 5. Veljače 2024.]

[23] tomreyn, "How to fix mtrr_cleanup: can not find optimal value," GitHub, 9. srpnja 202. [Na Internetu]. Dostupno: https://github.com/tomreyn/linux_mtrr_size_fix. [Pristupljeno: 5. veljače 2024].

[24] Gentoo Authors, „MTRR and PAT“, gentoo linux, 23. srpnja 2018. [Na Internetu] Dostupno: https://wiki.gentoo.org/wiki/MTRR_and_PAT [Pristupljeno: 5. veljače 2024].

[25] J. Corbet, "Indirect branch tracking for Intel CPUs," LWN.net, 31. ožujka 2022. [Na Internetu]. Dostupno: https://lwn.net/Articles/889475/. [Pristupljeno: 6. veljače 2024].

[26] „Transactional Memory“, IBM, 5. ožujka 2021. [Na Internetu]. Dostupno: https://www.ibm.com/docs/en/xffbg/121.141?topic=fortran-transactional-memory [Pristupljeno: 6. veljače 2024.]

[27] Linux Developeri, „linux/kernel/Kconfig.hz“, GitHub, 2023. [Na Internetu] Dostupno: https://github.com/torvalds/linux/blob/a38297e3/kernel/Kconfig.hz [Pristupljeno: 6. veljače 2024]

[28] Linux Developeri, „linux/kernel/power/Kconfig“, GitHub, 2024. [Na Internetu] Dostupno: https://github.com/torvalds/linux/blob/a38297e3/kernel/power/Kconfig [Pristupljeno: 8. veljače 2024]

[29] Frank Bodammer i sur. “SUSE LINUX – Administration Guide Chapter 9. Power Management / 9.3. ACPI“, Micro Focus, (bez. dat.), [Na Internetu] Dostupno: https://www.novell.com/de-de/documentation/suse91/suselinux-adminguide/html/ch09s03.html [Pristupljeno: 8. Veljače 2024]

[30] Linux Developeri, „linux/drivers/acpi/apei/Kconfig“, GitHub, 2019. [Na Internetu] Dostupno: https://github.com/torvalds/linux/blob/a38297e3/drivers/acpi/apei/Kconfig [Pristupljeno: 8. veljače 2024]

[31] Linux Developeri, „linux/kernel/module/Kconfig“, GitHub, 2023. [Na Internetu] Dostupno: https://github.com/torvalds/linux/blob/a38297e3/kernel/module/Kconfig [Pristupljeno: 10. veljače 2024]

[32] Linux Developeri, „linux/kernel/module/Kconfig“, GitHub, 2023. [Na Internetu] Dostupno: https://github.com/torvalds/linux/blob/a38297e3/block/Kconfig [Pristupljeno: 10. veljače 2024]

[33] Milan Navrátil i sur. “Chapter 1. Introduction to Control Groups (Cgroups)”, Red Hat, 9. Prosinca 2020. [Na Internetu] Dostupno: https://access.redhat.com/documentation/en-us%20/red_hat_enterprise_linux/6/html/resource_management_guide/ch01 [Pristupljeno: 10 veljače 2023.]

[34] Linux Developeri, „linux/mm/Kconfig“, GitHub, 2023. [Na Internetu] Dostupno: https://github.com/torvalds/linux/blob/a38297e3/mm/Kconfig [Pristupljeno: 13. veljače 2024]

[35] A. Chakraborty, "Contiguous and Non-Contiguous memory allocation in operating system," tutorialspoint, 6. travnja 2023. [Na Internetu]. Dostupno: https://www.tutorialspoint.com/contigious-and-non-contigious-memory-allocation-in-operating-system#:~:text=It%20requires%20less%20overhead%20and,memory%20block%20without%20any%20interruption. [Pristupljeno: 13. veljače 2024].

[36] “RapidIO Interconnect Architecture for Networking”, NXP. [Na Internetu] Dostupno: https://www.nxp.com/docs/en/fact-sheet/RAPIDIOFS.pdf [Pristupljeno: 14. Veljače 2024]

[37] “I3C and I3C Basic Frequently Asked Questions”, mipi alliance, 2024. [Na Internetu] Dostupno: https://www.mipi.org/resources/i3c-frequently-asked-questions [Pristupljeno: 14. Veljače 2024]

[38] Brendan Murphy, “I2C vs I3C: What are the Differences?”, 4 Svibnja 2024. [Na Internetu] Dostupno: https://www.totalphase.com/blog/2022/05/i2c-vs-i3c-what-are-the-differences/ [Pristupljeno 14. Veljače 2024]

[39] “MIPI SPMI System Power Management Interface”, mipi alliance, 2012. [Na Internetu] Dostupno:https://2384176.fs1.hubspotusercontent-na1.net/hubfs/2384176/MIPI_SPM_Interface_Overview_r01.pdf [Pristupljeno: 14. Veljače 2024]

[40] Linux Developeri, „High Speed Synchronous Serial Interface (HSI)“, The Linux Kernel, (bez dat.). [Na Internetu] Dostupno: https://www.kernel.org/doc/html/v4.14/driver-api/hsi.html [Pristupljeno: 14. Veljače 2024]

[41] Linux Developeri, “linux/drivers/greybus/Kconfig”, GitHub, 2023 [Na Internetu] Dostupno: https://github.com/torvalds/linux/blob/a38297e3/drivers/greybus/Kconfig [Pristupljeno: 14 veljače 2024]

[42] Linux Developeri, “linux/drivers/comedi/Kconfig”, GitHub, 2023 [Na Internetu] Dostupno: https://github.com/torvalds/linux/blob/a38297e3/drivers/comedi/Kconfig [Pristupljeno: 14 veljače 2024]

[43] Linux Developeri, “peci”, The Linux Kernel, (bez dat.). [Na Internetu] Dostupno: https://docs.kernel.org/peci/peci.html [Pristupljeno: 14 veljače 2024]

[44] Linux Developeri, “Kernel driver peci-cputemp”, The Linux Kernel, (bez dat.) [Na Internetu] Dostupno: https://docs.kernel.org/hwmon/peci-cputemp.html [Pristupljeno: 14 veljače 2024]

[45] “Simultaneous multithreading”, IBM, 24. Ožujka 2023. [Na Internetu] Dostupno: https://www.ibm.com/docs/en/aix/7.3?topic=concepts-simultaneous-multithreading [Pristupljeno: 18 ožujka 2024]

[46] Linux Developeri, “Core Scheduling”, The Linux Kernel, (bez dat.) [Na Internetu] Dostupno: https://docs.kernel.org/admin-guide/hw-vuln/core-scheduling.html [Pristupljeno: 18 ožujka 2024]

[47] "Package: libelf-dev (0.176-1.1ubuntu0.1 and others) [security]," Ubuntu. [Na Internetu]. Dostupno: https://packages.ubuntu.com/focal/libelf-dev. [Pristupljeno: 19. svibnja 2024].

[48] "Package: libssl-devel," Cygwin. [Na Internetu]. Dostupno: https://www.cygwin.com/packages/summary/libssl-devel.html. [Pristupljeno: 19. svibnja 2024].

[49] Shehryar Ahmed, „Compiling kernel 5.11.11 and later“, StackExchange, 6 travnja 2021. [Na Internetu] Dostupno: https://askubuntu.com/questions/1329538/compiling-kernel-5-11-11-and-later [Pristupljeno: 20 svibnja]

8. Popis slika