Univerza
v Ljubljani

Fakulteta za
gradbeništvo in
geodezijo

Jamova 2
1000 Ljubljana, Slovenija
telefon (01) 47 68 500
faks (01) 42 50 681
fgg@fgg.uni-lj.si

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE TEHNIČNO UPRAVLJANJE NEPREMIČNIN

Kandidat/-ka:

! ⌈9b a ! bD ! CL0

L½± ! W ! bW9 twh{¢hw{YL I ! b ! [L½ { th a h2Wh t¸¢ I hb {YwL t¢ ± [Lb ¦ · h{

Diplomska naloga št.: 34/TUN

D9h{ t ! ¢L ! [! b ! [¸{L{ L a t [9 a 9b¢ ! ¢L hb ² L¢ I t¸¢ I hb {/wL t¢{ Lb [Lb ¦ ·

Graduation thesis No.: 34/TUN

Mentor:
ØŠÖ LJNŜŘ Y I·∃ {I·Y2 5N2ōyŜ

Predsednik komisije:
izr. prof. dr. Tomaž Ambrožič

Član komisije:
viš. pred. mag. Mojca

Ljubljana, 18. 09. 2014

»TA STRAN JE NAMENOMA PRAZNA«

STRAN ZA POPRAVKE

Stran z napako	Vrstica z napako	Namesto	Naj bo

IZJAVE

Podpisani Alen Mangafić izjavljam, da sem avtor diplomske naloge z naslovom »Izvajanje prostorskih analiz s pomočjo Python skript v Linux OS«.

Izjavljam, da je elektronska različica v vsem enaka tiskani različici.

Izjavljam, da dovoljujem objavo elektronske različice v digitalnem repozitoriju.

Ljubljana, 12. 9. 2014 Alen Mangafić

BIBLIOGRAFSKO-DOKUMENTACIJSKA STRAN Z IZVLEČKOM

UDK: **004:528(043.2)**
Avtor: **Alen Mangafić**
Mentor: **viš. pred. mag. Samo Drobne**
Naslov: **Izvajanje prostorskih analiz s pomočjo Python skript v Linux OS**
Tip dokumenta: **Diplomska naloga – visokošolski študij**
Obseg in oprema: **45 str., 30sl.**
Ključne besede: **prostorske analize, Python, skripte, Linux, Windows, odprta koda**

Izvleček

Strojna in programska oprema sta temeljni orodji za izvajanje različnih del na vseh področjih inženirstva. Tehnologije, ki jih lahko srečamo na trgu, danes še posebej ponujajo veliko več kot so včasih. Ko govorimo o komercialni tehnologiji GIS in inženirski strojni opremi, govorimo o zelo dragih izdelkih, ki ponujajo odličen zagon in podporo vsakemu podjetju, ki jih zna uporabljati. Poleg komercialnih izdelkov obstaja širok spekter odprtokodnih rešitev, ki ponujajo enako dobre usluge in zadostujejo strokovni uporabi. Enako kot za programska orodja velja za proste in odprtokodne operacijske sisteme; primer so sistemi iz družine Linux. V diplomski nalogi smo preizkusili izvajanje prostorskih analiz v popolnoma odprtokodnem okolju, z vidika algoritemske vsebine, ki smo jo izvajali v programskem jeziku Python. Le-ta je danes standardni skriptni jezik za GIS-namene in za izdelavo vtičnikov v programih, kot so ArcGIS (ESRI, Arcpy) in QGIS. Enotnost programskega jezika ponuja povezavo med različnimi programskimi orodji, kot tudi med različnimi sistemskimi platformami. Zato je izmenjava skript in vtičnikov popolnoma skladna tudi v primeru, ko se pogovarjamo o prehodu in uporabi enake skripte med različnimi operacijskimi sistemi ali med različnimi GIS-orodji. Enostavnost jezika inženirjem ponuja osredotočenje na reševanje inženirskih problemov, brez prekomernega ukvarjanja s samim postopkom programiranja. Le-tega zahtevajo bolj kompleksni programski jeziki, kot sta C++ in Java. V delu smo opisali postopek od začetne namestitve operacijskega sistema Linux Mint do namestitve razvojnih okolij Python ter njihove uporabe. Izbrane skripte za izvedbo prostorskih analiz smo preizkusili v različnih operacijskih sistemih (v Linux Mint 17 Qiana ter v operacijskem sistemu Windows 8.1), rezultate pa smo ovrednotili kvalitativno in kvantitativno.

BIBLIOGRAPHIC-DOCUMENTALISTIC INFORMATION

UDC:	**004:528(043.2)**
Author:	**Alen Mangafić**
Supervisor:	**Sen. Lect. Samo Drobne, MSc**
Title:	**Geospatial analysis implementation with Python scripts in Linux OS**
Document type:	**Graduation thesis – higher education professional study**
Notes:	**45 p., 30 fig.**
Keywords:	**spatial analysis, Python, scripts, Linux, Windows, open source**

Abstract

Hardware and software represent basic tools for performance of various tasks in all fields of engineering. This is particularly true today, as available technologies offer considerably more than they used to in the past. The GIS technology and engineering software namely, represent particularly expensive products, which offer an excellent start-up and support for any company knowledgeable in their use. In addition to commercial products, there is also a wide spectrum of open source solutions, which offer same quality of service and meet the professional standards of use. The situation is similar when it comes to free and open source operating systems, the Linux family systems for example. Performance of spatial analysis in a fully open source system is tested in the thesis from the point of view of algorithm content, which was carried out in Python programming language. Python has become the standard scripting language for the requirements of GIS and for designing plug-ins in programmes such as ArcGIS (ESRI, Arcpy) and QGIS. Programming language uniformity provides the link among various programming tools as well as among various system platforms. The exchange of scripts and plug-ins is thus completely compatible, including in the cases of transit and use of the same script among different operating systems and among various GIS tools. The simplicity of language enables the engineers to focus on solutions of engineering tasks without having to spend too much time on the process of programming. Focus on programming is however required by more complex programming languages such as C++ and Java. The process from the initial installation the Linux Mint operating system until installation of Python development environments and their use is described in the thesis. The thesis provides tests of selected scripts for performing spatial analysis in various operating systems (in Linux Mint 17 Qiana and in Windows 8.1 operating systems) and gives qualitative and quantitative evaluation of the results.

ZAHVALA

Še posebej se zahvaljujem mentorju, viš. pred. mag. Samu Drobnetu, za vso pomoč in nasvete, katere sem dobil ob pisanju diplomske naloge. Zahvaljujem se tudi asist. mag. Oskarju Sterletu, za kakovostno tutorstvo med študijem.
Iz srca se zahvaljujem mami, Jadranu, Toniju, Bočaju, Jalli in Gromki, ker so mi stali ob strani med študijem.

Hvala!

Kazalo

KAZALO SLIK

1 UVOD

Prostorske analize so sestavni del geo-informacijske stroke. Njihovo izvajanje poteka s pomočjo različnih programskih orodij, med katerimi jih je večina dragih, a na srečo obstajajo tudi cenejše, celo zastonjske rešitve.

Včasih se zgodi, da v istih orodjih nimamo dostopa do posebne želene funkcije, katera bi olajšala morebitne repetitivne analize. K razširitvi funkcionalnosti nam pomaga razširitev z izgradnjo vtičnikov in uporabo skript v programskih jezikih, katere orodje podpira. Hiter razvoj odprtokodnih skupnosti je pripeljal do hitrega razvoja programskih orodij in mreženja uporabnikov. Odprtokodnost omogoča boljšo prilagodljivost, ki pa je lahko različna od sistema do sistema. Za lažje razumevanje in uporabo odprtokodnih sistemov je ključno poznavanje njihovih lastnosti. Zato smo v drugem poglavju tega diplomskega dela opisali koncept operacijskega sistema ter skozi kratek zgodovinski pregled prišli do operacijskih sistemov Linux.

Opisali smo vlogo in izgled najbolj popularnih Linux distribucij, nato pa se osredotočili na njihovo uporabo. Opisali smo namestitev, pogosto uporabljana programska orodja v inženirstvu ter, v tretjem poglavju, nadaljevali podrobneje z opisom programskega jezika Python. Nato smo – po opisu sintakse in semantike jezika – opisali še nekaj primerov programskih skript. V četrtem poglavju smo se ukvarjali le z izbranimi skriptami za izvajanje prostorskih analiz. Opisali smo njihovo strukturo, prikazali vhodne in izhodne datoteke ter primerjali njihovo funkcionalnost na različnih platformah.

2 LINUX

2.1 Operacijski sistemi

"Operacijski sistem je programska oprema, katera upravlja strojno opremo računalnika" (Silberschatz in sod., 2009, str. 3). Če bi računalnike primerjali s človeškim telesom, bi vlogo operacijskega sistema (OS) lahko primerjali s fiziološko funkcijo možganov oz. centralnim nadzorom telesa in koordinacijo njegovih procesov.
"Operacijski sistem je sestavljen iz dveh delov: jedra (angl. *kernel*), ki skrbi za nadzor procesorja, pomnilnika, procesov in naprav v računalniku, ter uporabniškega vmesnika, ki skrbi za interakcijo med uporabnikom in računalnikom" (MaFiRa-Wiki, 2014).
Vedenje in način komunikacije med tema dvema deloma se razlikujeta od sistema do sistema – tudi zaradi fizičnih lastnosti jeder in strojne opreme. Strukturno lahko jedra operacijskih sistemov delimo na monolitična, mikrojedra ter hibridna. Glede na komercialno uporabo razlikujemo prosta in plačljiva jedra, glede na razvojno filozofijo pa jih delimo na odprto- in zaprtokodna.

2.2 Kratka zgodovina OS

V moderni zgodovini človeštva se je v obdobju med obema vojnama in po njima zgodil podoben vzorec razvoja. Med vojnama 20. stoletja je prihajalo do prekomernih kapitalskih investicij v vojaški tehnološki razvoj – zaradi tega se je v manj kot 40 letih zgodil velik tehnološki razkorak med generacijama. Poleg inovacij bojnih sredstev je prišlo tudi do pomembnih odkritij na področjih navigacije in daljinskega zaznavanja (Gee-H, RADAR, hidrofon), medicine (prenosna radiologija, uporaba penicilina), materialov, energetike (nuklearna energija) in računalništva za namene šifriranja (Enigma, Lorentz; Wikipedia, 2014).
Po letu 1945 je prišlo do številnih novih odkritij, kar je sprožilo velik izziv ostalemu akademskemu svetu. Prvi računalniki niso vsebovali operacijskih sistemov. Delovali so s pomočjo programov, ki so morali, poleg svojih algoritmov, že v kodi samega programa definirati vse potrebne sistemske namestitve ter način komunikacije programa in strojne opreme. Veliko programerjev je zato opravljalo enako delo. Da bi se izognili nepotrebnemu ponavljanju kodnih vrstic, je veliko njih shranjevalo najbolj pogosto uporabljane operacije. Z namenom lažjega in hitrejšega dela so začeli oblikovati knjižnice izbranih operacij, kar predstavlja velik korak v smeri razvoja sodobnih operacijskih sistemov (MaFiRa-Wiki, 2014).
Podjetje IBM je na začetku petdesetih let začelo prodajati prve komercialne računalnike, med katerimi je prvi komercialni izdelek bil IBM 701, znan tudi kot *Defense Calculator*. Nekaj let

pozneje se je pojavil revolucionarni računalnik, IBM 704. Bil je prvi, ki je vseboval določeno prednameščeno število integriranih strojnih ukazov. Za ta računalnik je bil razvit tudi prvi operacijski sistem v zgodovini, GM-NAA I/O, ki ga je razvil Robert L. Patrick za potrebe podjetij General Motors in North American Aviation. Glavna funkcija sistema GM-NAA I/O je bila ta, da je lahko izvajal zaporedne serije programov. Tako jo je, ko je prvi program izvedel svoj proces, avtomatsko začel izvajati naslednji program (angl. *batch processing*). Za namene istega računalnika so razvili tudi prve visokonivojske programske jezike, kot so FORTRAN in LISP (Boyer, 2004).

Leta 1964 je IBM predstavil novo serijo računalnikov, System/360. To je bila prva serija računalnikov, ki je kljub različni strojni opremi temeljila na enaki prednameščeni sistemski arhitekturi OS/360 oz. *System/360 Operating System* (IBM, 1965). Z računalniki serije System/360 so se tudi določili nekateri konceptni standardi, kateri so še danes zanimivi. Definirali so dolžino bita (8 bajtov) ter glede na bajte tudi uredili lociranje podatkov v pomnilniku, določili 32-bitno dolžino dolgih besed (angl. *longword*) ter uvedli magnetno disketno enoto (IBM 2302).

Na koncu šestdesetih let so programerji *AT&T Bell Labs* razvili UNIX, prvi operacijski sistem, ki je bil zmožen selitve med različnimi arhitekturami. UNIX je bil napisan v programskem jeziku C, ki je bil izdelan posebej za ta sistem (MaFiRa-Wiki, 2014). Zaradi strukture v C-jeziku (lažje konfiguriranje na različni strojni opremi) ter velikega števila vgrajenih ukazov je bil do takrat UNIX najzmogljivejši operacijski sistem.

2.2.1 Projekt GNU in odprtokodne alternative

Zaradi korporativno določenih avtorskih pravic, UNIX-a niso smeli prodajati samostojno kot operacijski sistem. Zato sta se širili zastonjska akademska različica ter različica, vgrajena v računalnike podjetja AT&T Bell Labs, ki pa je bila zelo draga. V takratni vojni operacijskih sistemov so zmagovali računalniki z manj zmogljivim sistemom MS-DOS, čigar marketinški uspeh je temeljil na ugodnejši ceni. V osemdesetih letih je velik del programerjev menil, da je nesmiselno razvijati že v začetku slabšo platformo; menili so tudi, da je arhitektura UNIX-a optimalna za nadaljnji razvoj računalniških sistemov. Celo Microsoft je želel spremeniti MS-DOS v različico vse bolj podobni UNIX-u (Lewis, 1986). Spreminjanje in razvijanje UNIX-a se je nadaljevalo v veliko različnih projektih, kot so: BSD (Berkeley Software Distribution), Xenix (Microsoft), HP-UX (Hewlett-Packard UniX), IBM AIX, Solaris (Oracle Corporation), Sequent in MINIX(Wikipedia, 2014).

Takrat se je porodila ideja o izdelavi novega operacijskega sistema, ki bo temeljil na UNIX-u in ki bi bil prosto dostopen vsem. Eden najbolj pomembnih akterjev v boju za prostodostopno programsko opremo je bil Richard Mathew Stallman (v hekerskih krogih znan kot RMS).

Stallman se je zavzemal za brezplačno distribucijo programske opreme. Za dosego tega koraka pa je bilo treba najprej izdelati brezplačni, odprtokodni OS. Leta 1983 je Stallman javno objavil svojo idejo novega, zastonjskega OS-a, ki bi temeljil na UNIX-u ter povabil vse zainteresirane k sodelovanju. Sistem je poimenoval GNU (GNU's Not Unix!) in projekt se je začel razvijati že leta 1984.

Slika 1: Logotip GNU projekta (gnu.org, 2014)
Vir: http://www.gnu.org/graphics/official%20gnu.svg (Pridobljeno 24. 7. 2014.)

Stallman je populariziral koncept licenciranja, *copyleft* (igra besed; nasprotje besedne zveze *copyright*). V praksi je *copyleft* oblika licenciranja *copyright*, ki odpravlja vse pravice, ki se tičejo avtorjev, ter opredeljuje programski izdelek kot prost (zastonjski ter odprtokoden), z neomejenimi pravicami spreminjanja in razmnoževanja. Pomembna prepoved, katera nastopa v licencah *copyleft*, je ta, da vsak programski izdelek, nastal na podlagi izdelka z licenco *copyleft*, mora to licenco prenašati naprej (t. i. virusna licenca; angl. *viral licence)*. Na ta način so se uporabniki zavarovali pred tem, da bi nekdo izkoristil obstoječo kodo za lastni profit in program pretvoril v zaprtokodnega ter tako odvzel uporabnikom prost dostop do izdelka.

Slika 2: "All wrongs reserved"; eno prvih sporočil *copylefta* v programu Tiny BASIC (1976)
Vir: http://upload.wikimedia.org/wikipedia/commons/5/5c/Copyleft_All_Wrongs_Reserved.png
(Pridobljeno 24. 7. 2014.)

Stallman je 4. oktobra 1985 ustanovil FSF (Free Software Foundation), neprofitno organizacijo, katere namen je bil osvoboditev računalniške programske opreme iz korporativnega okolja. Poleg Stallmana se je aktiviralo še več projektov, kjer so se razvijali odprtokodni sistemi, temelječi na UNIX-u, med katerimi je najbolj uspešen projekt GNU/Linux.

2.2.2 Začetki Linux-a

Leta 1991 je finski študent računalništva, Linus Torvalds, začel pisati jedro za nov, odprt in zastonjski operacijski sistem, temelječ na UNIX-u (oz. na UNIXOV-em derivatu, MINIX-u). Aprila istega leta je objavil svoj izdelek na medmrežju, jedro *Linux 0.01.* Izdelek je registriral kot del projekta GNU pod licenco *Splošno dovoljenje GNU (*angl. *GPL – GNU General Public License*). V njega je tudi vgradil GNU-jevo ukazno lupino *BASH* (angl. *BASH shell*). S tem korakom se je začela revolucija razvojne prakse. Torvalds je s tem tudi pokazal pravi pomen besede *skupnost*. Razvoj na Linux-u temelječih jeder je potekal tako, da je programska koda bila vedno prosto dostopna – kar pomeni, da jo lahko kdorkoli spreminja. Komunikacija je potekala prek FTP-protokolov (danes večinoma prek strani, kot je GitHub). Sistem se je razvijal tako, da so programerji, zainteresirani za projekt, prostovoljno razvijali kodo jedra ter izboljševali sistem (pisanje gonilnikov, popravljanje hroščev itd.). Zaradi zanimive narave projekta ter zaradi vse večjih hitrosti dostopa do medmrežja se je v samo nekaj letih skupnost, ki je programirala jedro Linux, povečala na več tisoč ljudi. Danes je število Linux-ovih uporabnikov, programerjev, podjetij ter sprememb izvirne kode še mnogo večje.

Trenutno v samo eni uri pride do povprečno 9 posodobitev jedrne kode (CNET, 2013).

Slika 3: Tux, maskota operacijskega jedra Linux
Vir: http://en.wikipedia.org/wiki/Tux#mediaviewer/File:Tux.png (Pridobljeno 5. 8. 2014.)

2.3 Linux distribucije

Po letu 1994, ko je bilo že objavljeno jedro Linux 1.0, je prišlo do pojava velikega števila odprtokodnih operacijskih sistemov. Le-ti so nastali na osnovi Linux-ovega sistemskega jedra. Take operacijske sisteme, ki vsebujejo jedro Linux, zbirko poljubnih sistemskih in programskih orodij ter namizij, imenujemo *Linux distribucije*.

Pri vsakodnevni uporabi računalnika ima vsak računalniški uporabnik svoje želje in preference. Izbira je lahko zgolj estetske narave, stvar vsiljene navade ali pa je premišljena izbira z namenom določene uporabe (od profesionalne do vsakodnevne). Zaradi tega, ker je skupnost, ki razvija Linux tako široka in je potreba za odprtokodnim sistemom prisotna na veliko različnih področjih, se je od leta 1991 do danes razvilo že 741 različnih Linux distribucij (Distrowatch, 2014). Distribucije se razlikujejo med seboj na mnogo načinov, saj Linux ni namenjen le osebnemu računalniku. Linux danes prevladuje na trgu superračunalnikov, kjer pokriva kar 97 % oz. upravlja 485 od 500 najhitrejših računalnikov (ZDNET, 2014). Prav tako tudi najbolj popularen operacijski sistem na pametnih telefonih, Android, temelji na Linux-ovem jedru in bo do konca 2014 imel milijardo uporabnikov (Computerweekly, 2014). Zaradi njegove stabilnosti, je ta OS za upravljanje vesoljskih postaj izbrala tudi NASA. Naštevanje se lahko nadaljuje na področju GPS-naprav, bankomatov, avtomobilskih sistemov, e-bralnikov, medicinskih pripomočkov itd. Zaradi narave diplomske naloge se bomo osredotočili le na Linux distribucije za osebne računalnike.

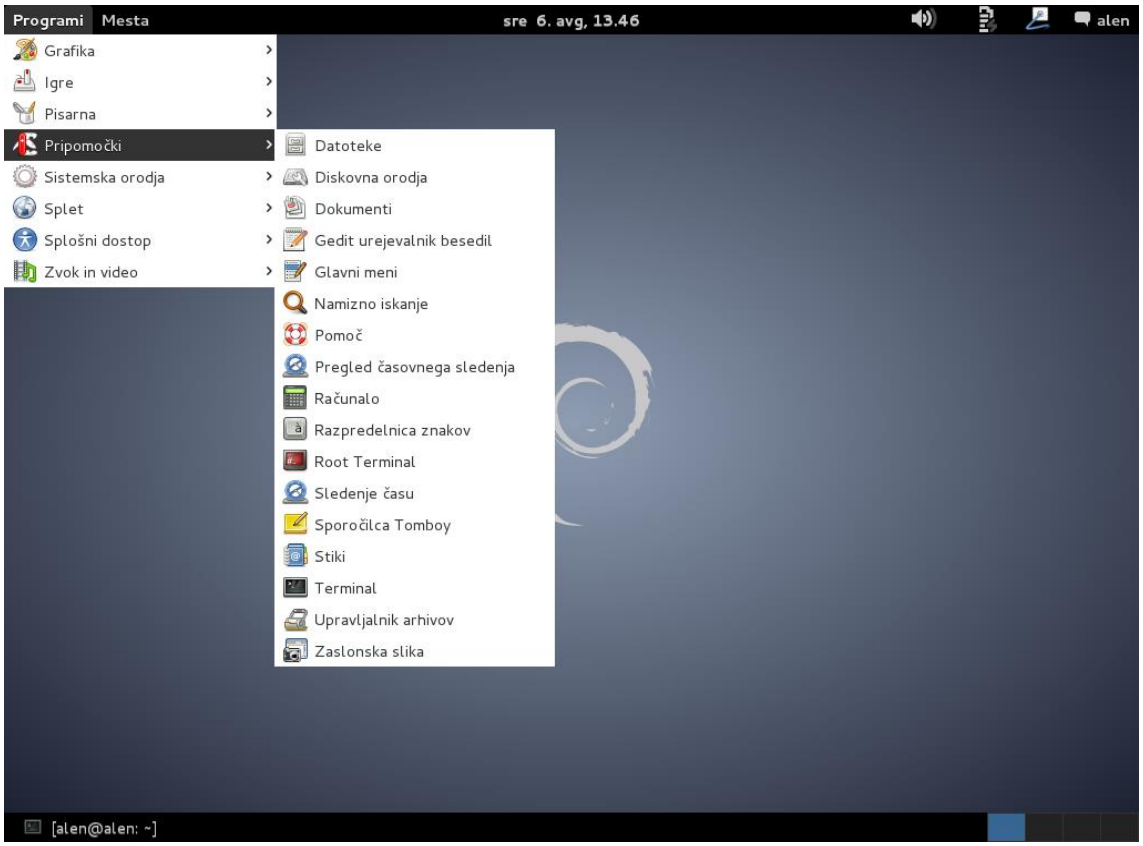
2.3.1 Linux distribucija za osebne računalnike

Distribucije, ki temeljijo na jedru Linux, so praviloma zastonjske in odprtokodne narave, čeprav nekatere ponujajo tudi zaprtokodne rešitve (določeni gonilniki, kodeki itd.). Glede na razvojno naravo delimo Linux distribucije na dve veliki skupini: na skupine s komercialnim ozadjem, kot so Ubuntu, Fedora, openSUSE, Mandriva Linux, ter na neodvisne distribucije, kot so Debian, Gentoo, Slackware in Arch Linux, katere so v celoti razvite v skupnosti. Obstaja nekaj primerov (npr. Red Hat), ki so plačljivi, vendar se v tem primeru ne trži z izdelkom, temveč z uslugo. Z nakupom licence se v tem primeru plačuje določen čas tehnične podpore za uporabnike (poslovni paketi) ter v tem času rešitve za vse tehnične probleme v kratkem odzivnem času. Zaradi posebne datotečne zgradbe zahtevajo Linux operacijski sistemi manj računalniških virov kot ostali komercialni sistemi (npr. Windows). Glavni razlog je v tem, da si programska orodja medsebojno delijo knjižnice. To pomeni da, npr. pri namestitvi programa za gledanje videov v Linux OS, namestitveni proces poišče morebitne obstoječe knjižnice, ki jih potrebuje za svoje delovanje (npr. kodeke za predvajanje zvoka), ter v primeru da jih zazna, naloži te neobstoječe knjižnice v sistem (več o tem na LGM.FRI). Nova, nespremenjena namestitvena različica Linux distribucije Ubuntu 14.04.1 LTS, zasede nekaj manj kot 5 GB (Ubuntu, 2014) ter že na začetku vsebuje vse potrebne gonilnike (angl. *out-of-the-box),* pisarniška orodja (Libre Office), programe za obdelavo grafik (GIMP), spletna orodja (Firefox in Thunderbird), multimedijske predvajalnike (VLC player, Banshee) ter vgrajeno podporo več različnih programskih jezikov (Python itd.). Nova, nespremenjena namestitev operacijskega sistema Windows 8.1 zahteva med 16 in 20 GB prostora (Microsoft, 2014). V današnjem valu Linux distribucij lahko razdelimo ta OS na več različnih načinov, vendar bomo v diplomski nalogi opisali le različice z največ uporabniki.

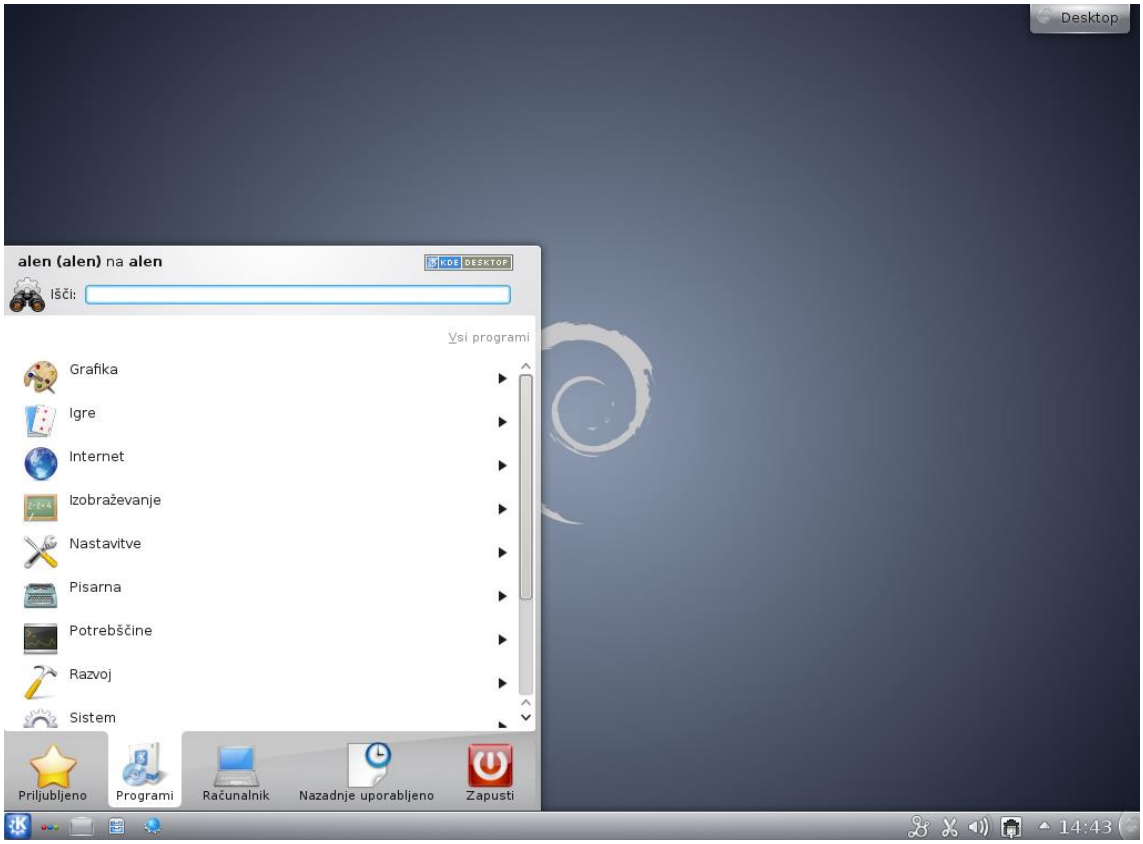
2.3.1.1 Debian

Debian ter njegovi derivati so glede na število uporabnikov ena najbolj vplivnih distribucij Linux-a. Prvi ga je razvil Ian Murdock leta 1993. Do danes se je razvilo veliko njegovih različic, ki se razlikujejo po namenu uporabe in po prijaznosti do uporabnika. Debian ima posebno strukturo nalaganja programske opreme prek paketov, ki se nahajajo v t. i. strežnih skladiščih (angl. *repository*). Paketi, ki se nalagajo na uradni distribuciji, so odprtokodni in v skladu s pravili GPL-licence. Zaradi tega, ker uporabnik lahko spreminja izvor paketov, je Debian zelo prilagodljiva distribucija, ki omogoča dodajanje paketov izven odprtokodne serije; uporabnik lahko uporablja testne različice programskih orodij ali tudi druge zaprtokodne programe. Debianov upravljalnik paketov je *APT (Advanced Packaging Tool),* k+i izvaja namestitve in posodobitve preko BASH-a ali orodja *Synaptic Manager* in njegovih

derivatov. Debian podpira 13 različnih procesorskih platform, seveda tudi najbolj uporabljane amd64 (64-bitni procesorje) in i386 (32-bitni procesorje) (Debian, 2014). Kot pri večini Linux distribucij, je treba pri nameščanju orodij biti prijavljen v sistem kot *super uporabnik* (angl. *superuser* ali *root).* Zaradi tega ne more priti do sprememb v sistemu brez dovoljenja uporabnika. Le-to doprinese k dodatni varnosti sistema (prepreči pristop računalniškim virusom, katerih je sicer v svetu Linuxa zelo malo). Zanesljivost in stabilnost Debiana je tudi razlog, da ga je NASA leta 2013 izbrala za operacijski sistem na računalnikih svojih vesoljskih postaj (ZDNET, 2013). Obstaja več različnih namizij, s katerimi Debian deluje, trenutno so uradne GNOME, KDE, Xfce in LXDE.



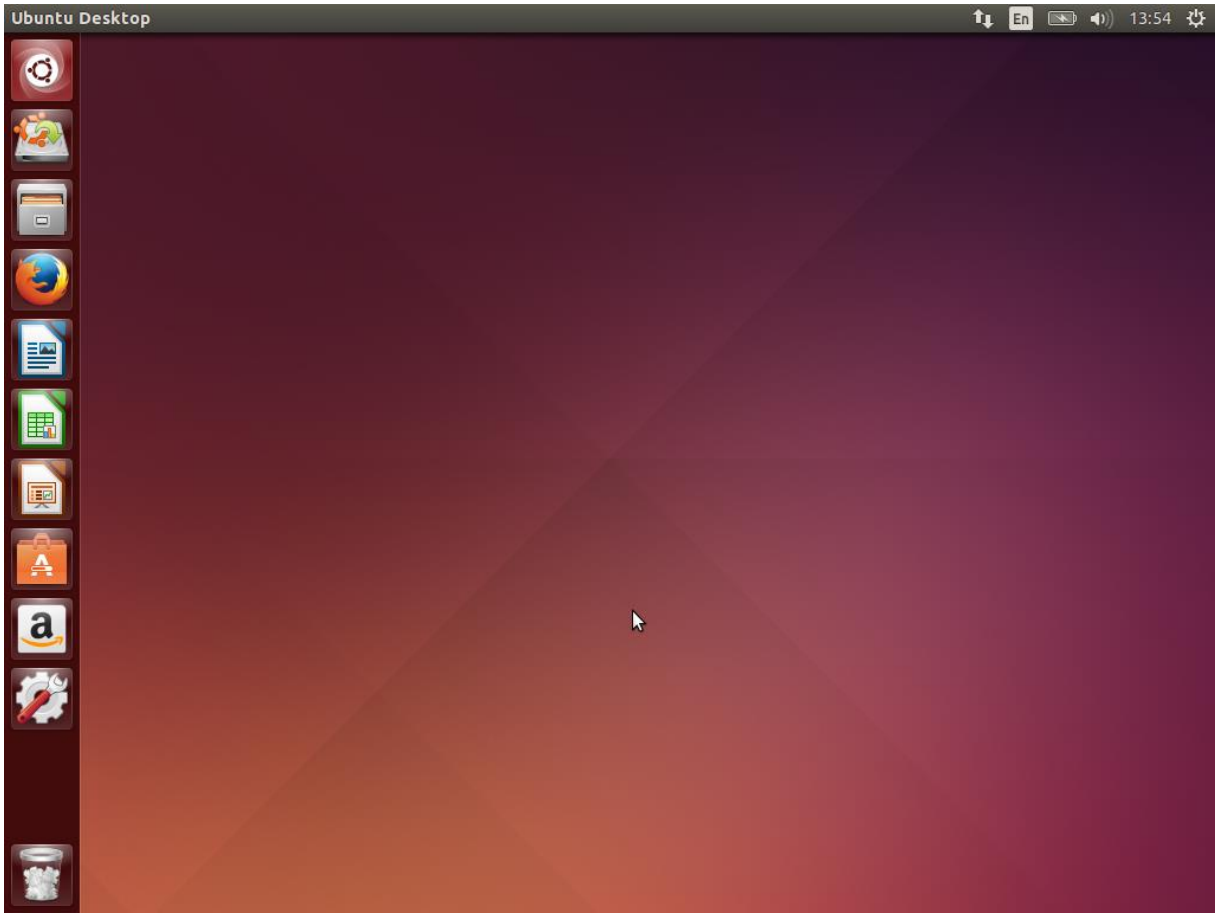
Slika 4: Debian Wheezy z GNOME namizjem



Slika 5: Debian Wheezy s KDE namizjem

2.3.1.1 Ubuntu

Ubuntu je distribucija, ki temelji na Debianu. Razvija jo podjetje Canonical Ltd. Glavni razlog uspeha te distribucije je udobnost pri vsakodnevni uporabi. Ubuntu je preprost za uporabo, posodablja se z vsako novo različico vsakega pol leta, glede na uporabnost pa ponuja kompromis med stabilnim ter sodobnim sistemom (najbolj sodobna oprema ne ponuja največje stabilnosti). Zelo lahko ga je namestiti ter, kot večina Linux OS, ponuja možnost t. i. *live boot*-a, kar pomeni, da ga lahko uporabnik naloži na medij (USB-ključ, DVD) in v realnem času testira na svojem računalniku, brez da bi karkoli nameščal oz. spreminjal na trenutnem sistemu. Obstaja veliko projektov Ubuntu-ja, ki se razlikujejo po vmesniku ter vsebini (*Edubuntu* za šolarje, *Ubuntu lite* za starejše računalnike itd.), zadnja različica pa uporablja vmesnik *Unity (Ubuntu 14.04.1 LTS, 2014).* S poenostavljenim orodjem *Ubuntu Software Center* ponuja takojšnjo namestitev želenih programov z le enim klikom. Enako kot Debian, pa ponuja širitev strežnih skladišč na zaprtokodne in ostale.



Slika 6: Ubuntu 14.04.1. LTS z Unity namizjem

2.3.1.2 Linux Mint

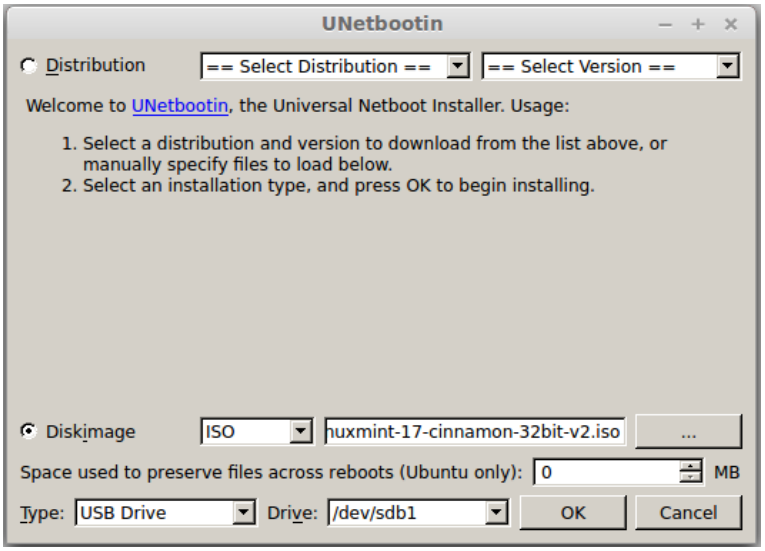
Linux Mint, distribucija, ki temelji na Ubuntu-ju ali Debian-u, je najbolj popularna Linux distribucija v letu 2014 (Distrowatch, 2014). Popularnost Linux Mint je povezana z njegovimi lastnostmi, kot so preprostost, varnost in modernost. Ponuja tudi vmesnike, podobne sistemu Windows, a uporabnikom sistemov bolj prijazne. Predvsem zaradi vsebovanja zaprtokodnih gonilnikov (Adobe Flash, MP3 itd.) pa novim Linux uporabnikom omogoča takojšnjo in lažjo uporabo. Zaradi takih vmesnikov, kot je *Cinnamon* in *MATE*, so pridobili veliko število Ubuntu uporabnikov – predvsem tistih, ki se niso strinjali z razvijanjem marsikomu nepraktičnega *Unity* vmesnika, ki ga je opravljal Canonical Ltd. Izvajanje prostorskih analiz s pomočjo skript Python bomo izvajali v operacijskem sistemu Linux Mint.

Slika 7: Linux Mint 17 “Qiana” s Cinnamon namizjem

2.4. Namestitev Linux Mint OS

2.4.1 Priprava namestitvenega medija

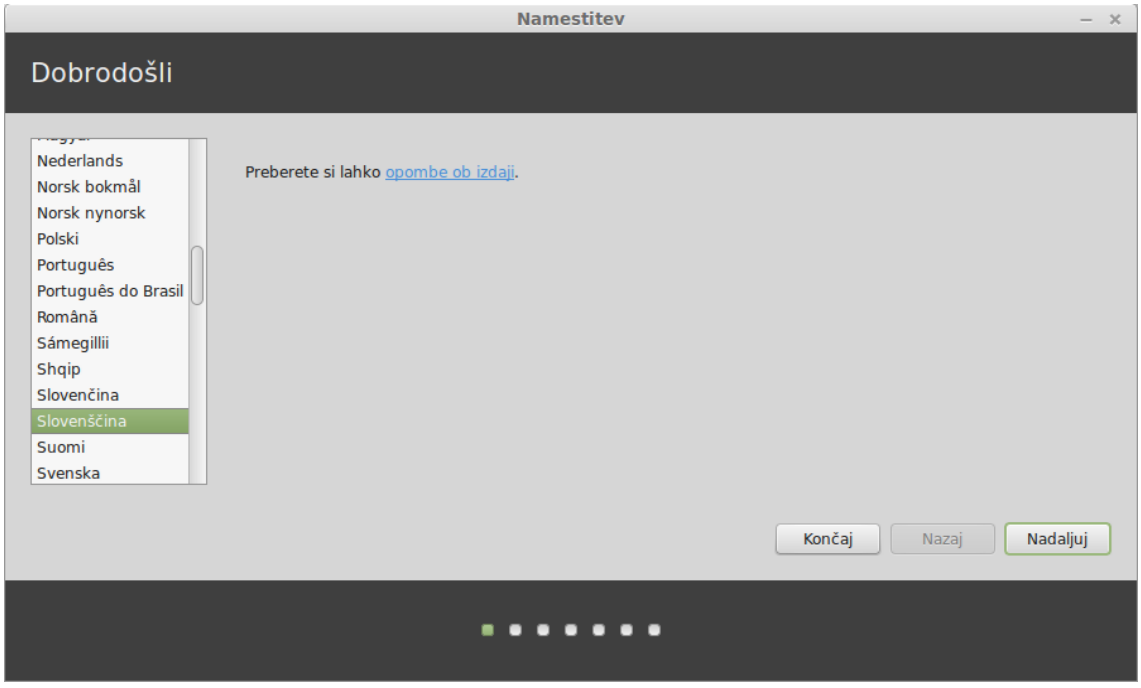
Linux Mint se lahko, tako kot ostale Linux distribucije, prenese z uradne spletne strani. ISO-sliko lahko prenesemo neposredno s ponujenih strežnikov ali preko protokola *torrent*. Preneseno sliko lahko zapišemo na optični medij (DVD/CD) ali zunanji pomnilnik (npr. USB-ključ). Zaradi večje hitrosti namestitve ter ekonomičnost smo izbrali namestitev preko USB-ključa. Pripravo medija smo izvedli s pomočjo programa *Unetbootin* ter v BIOS-u računalnika omogočili sistemski zagon z USB-vhoda.



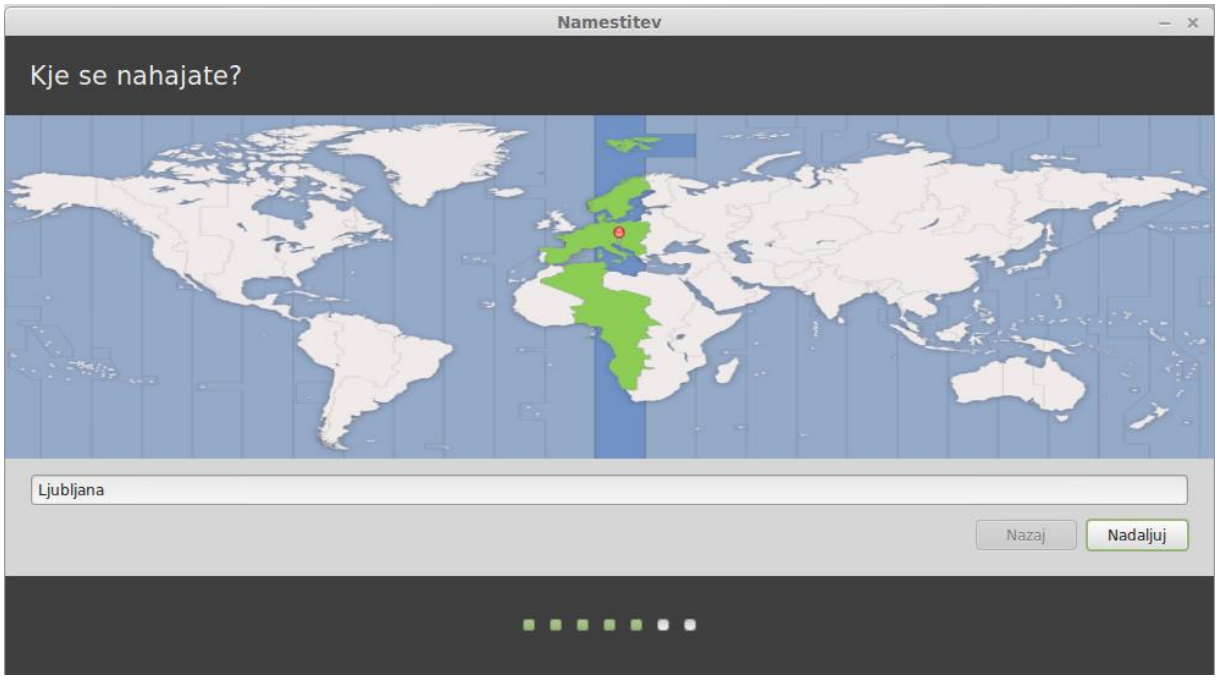
Slika 8: Izdelava namestitvenega USB-ključa v *Unetbootin*-u

2.4.2 Namestitev po korakih

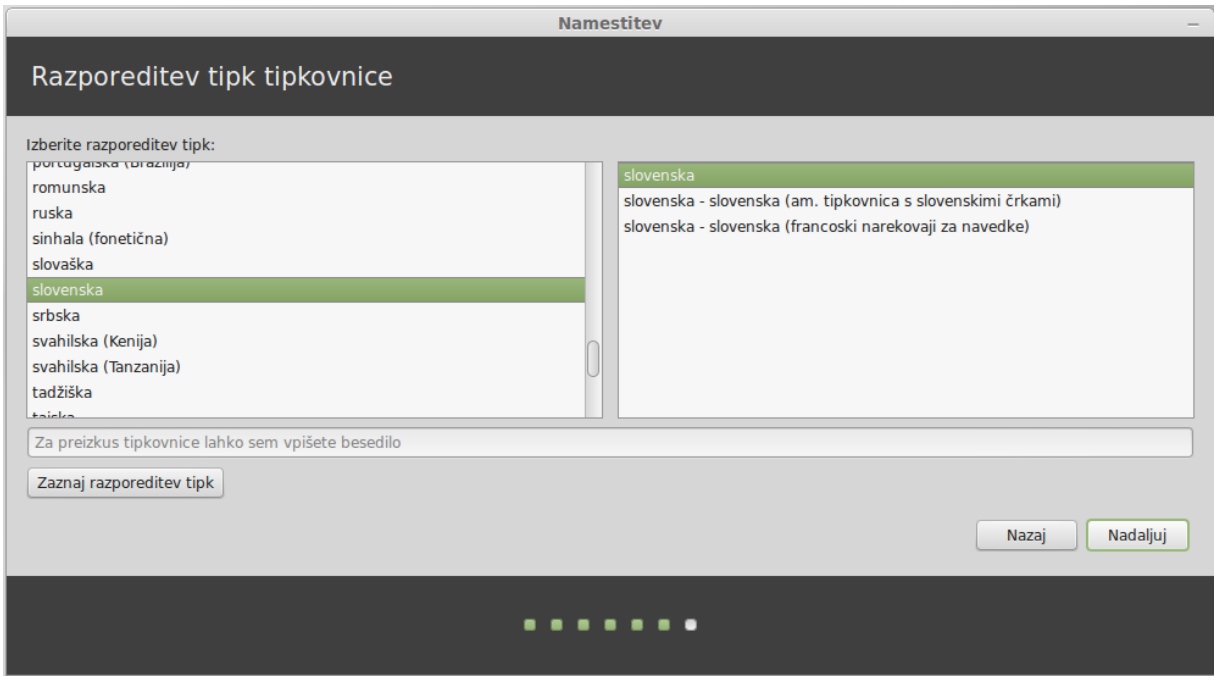
Po zagonu računalnika izberemo možnost sistemskega zagona z USB-ključka ter nadaljujemo z namestitvijo s pomočjo enostavnih grafičnih navodil. Izbrali smo najbolj enostaven način namestitve. V primeru, da bi želeli namestiti Linux Mint skupaj z Windows, ali drugim obstoječim sistemom (t. i. *multi-boot)*, bi postopek zahteval ročno delitev pomnilnika ter še druge dodatne nastavitve.



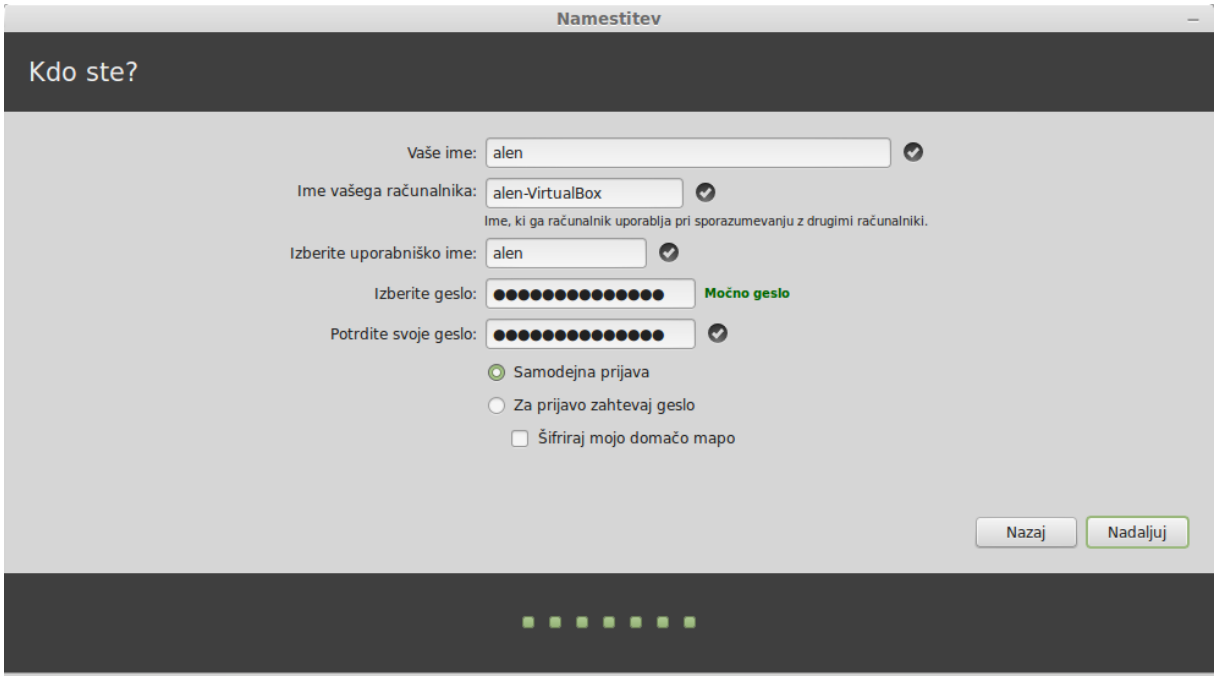
Slika 9: Izbira sistemskega jezika



Slika 10: Izbira lokacije zaradi datumskih in političnih postavk (valuta itd.)



Slika 11: Prilagoditev tipkovnice



Slika 12: Izdelava uporabniškega računa

Slika 13: Avtomatska namestitev operacijskega sistema

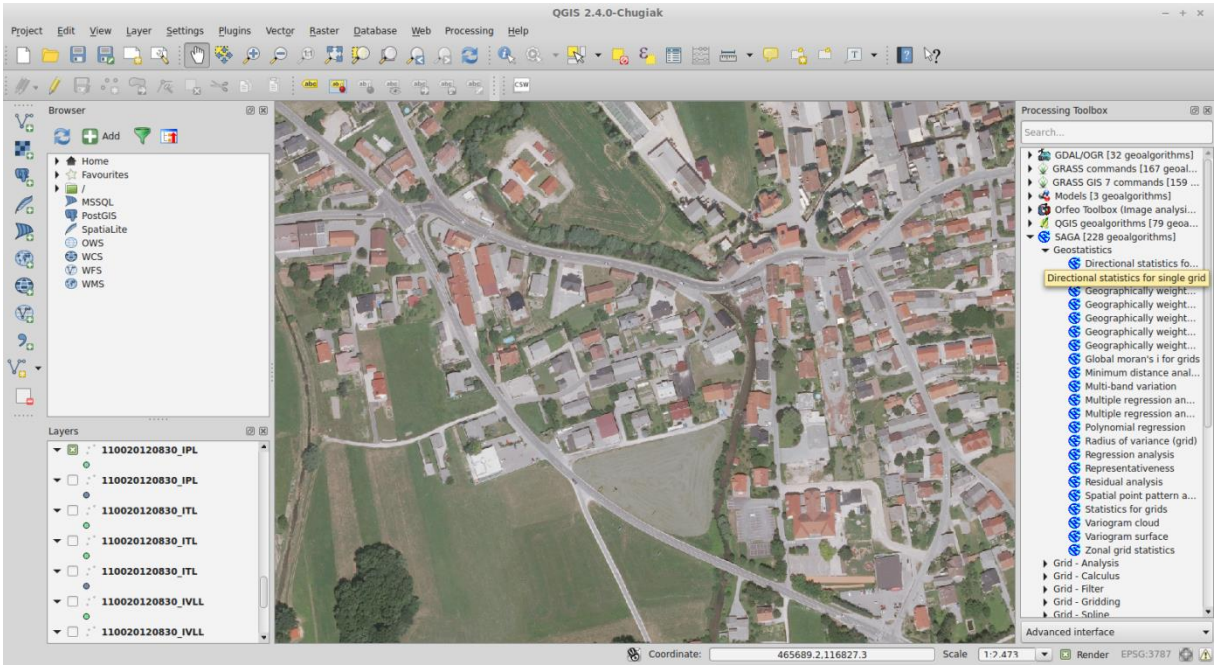
2.5 Primeri programov za inženirske in študijske namene v Linux-u

2.5.1 QGIS

QGIS (tudi Quantum GIS) je odprtokodno GIS (geografski informacijski sistem) programsko orodje. Večplatformno orodje je zgrajeno v jezikih C++ in Python. Vsebuje grafični vmesnik, zgrajen s pomočjo Qt-knjižnic gradnikov.

Z leti je QGIS pridobil veliko število funkcij z integracijo različnih GIS-programskih paketov in orodij, kot so:

- relacijska zbirka PostgreSQL in PostGIS dodatek,
- SEXTANTE (zbirka algoritmov iz GRASS GIS in SAGA GIS),
- *Plugin builder*, orodje za izdelavo in uvoz geoprostorskih in ostalih skript, izdelanih v Python-u.

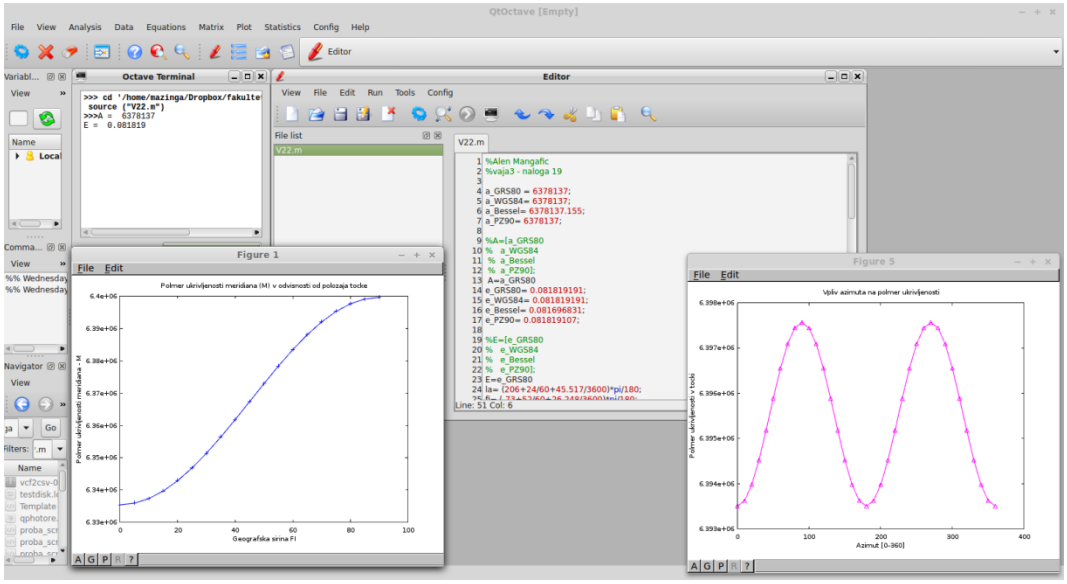


Slika 14: QGIS 2.4 Chugiak

2.5.2 GNU Octave

GNU Octave je odprtokodni odgovor na program MATLAB. Izvaja numerične analize, njegov glavni namen pa je izvajanje operacij z matrikami. Napisan je v jeziku C++ in vsebuje možnost širitve z različnimi orodji, kot so *gnuplot* (izrisovanje grafov), *mapping* (kartiranje) in *civil engeneering*. Octave do verzije 3.8 ni imel uradnega vmesnika, ampak je računanje potekalo z izvajanjem skript, pisanih v terminalu BASH različnih predvajalnikov (angl.

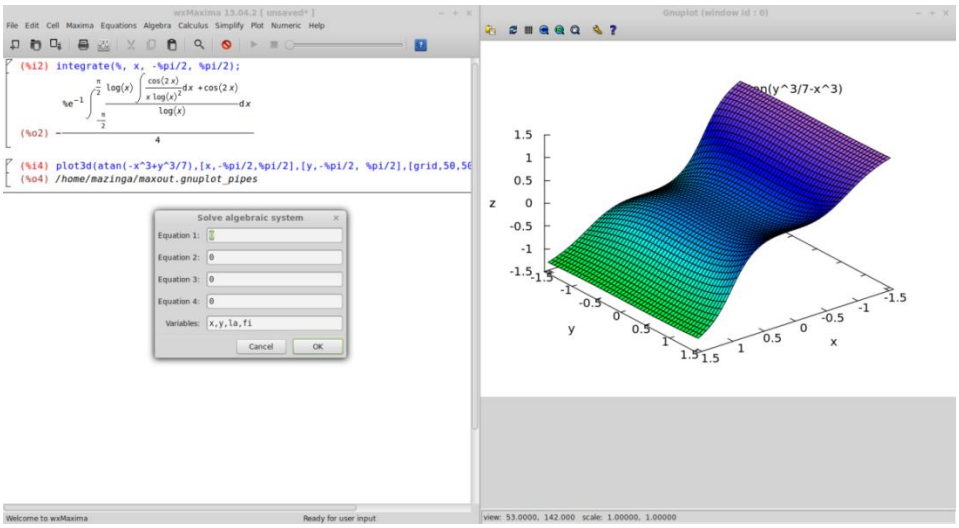
compiler) ali z neuradnimi vmesniki, kot je QtOctave.



Slika 15: GNU Octave z vmesnikom QtOctave

2.5.3 Maxima

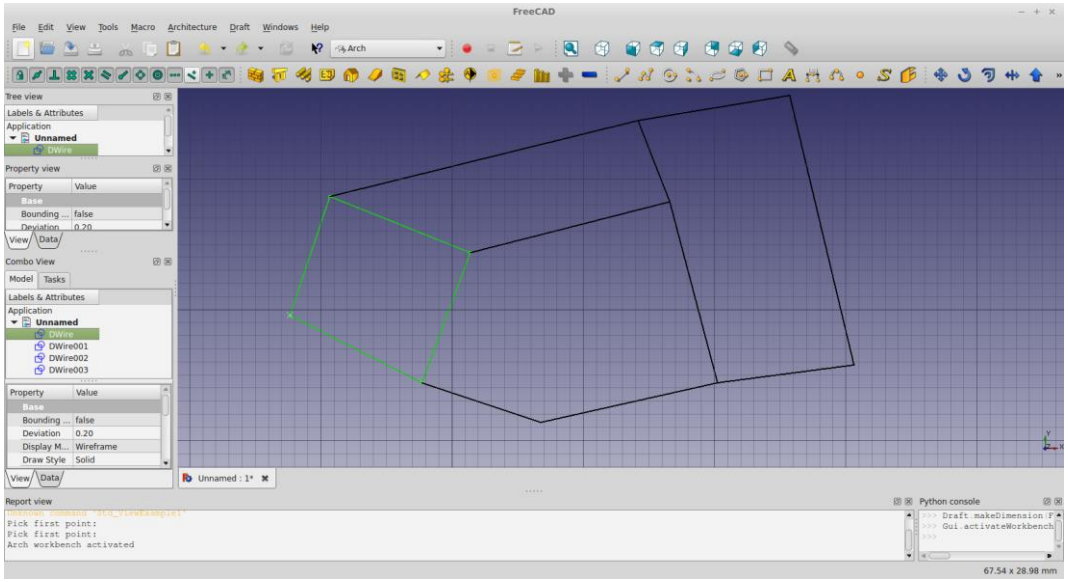
Maxima je odprtokodni algebrski program, uporabljan za izvajanje številnih numeričnih in simbolnih problemov, tako kot program Mathematica (Wolfram). Enako kot Octave, vsebuje različne integrirane pakete, kot je *gnuplot*, ter omogoča širitev in namestitev dodatnih. Pisan je v programskem jeziku Common Lisp. Eden najbolj priljubljenih vmesnikov za uporabo programa Maxima je wxMaxima, ki ponuja številne bližnjice za izvajanje matematičnih operacij, kot so rešitve enačb, matrične in vektorske analize, integriranje, izdelovanje Taylorjevih vrst, Laplaceovih transformacij itd.



Slika 16: Maxima z vmesnikom wxMaxima

2.5.4 FreeCAD

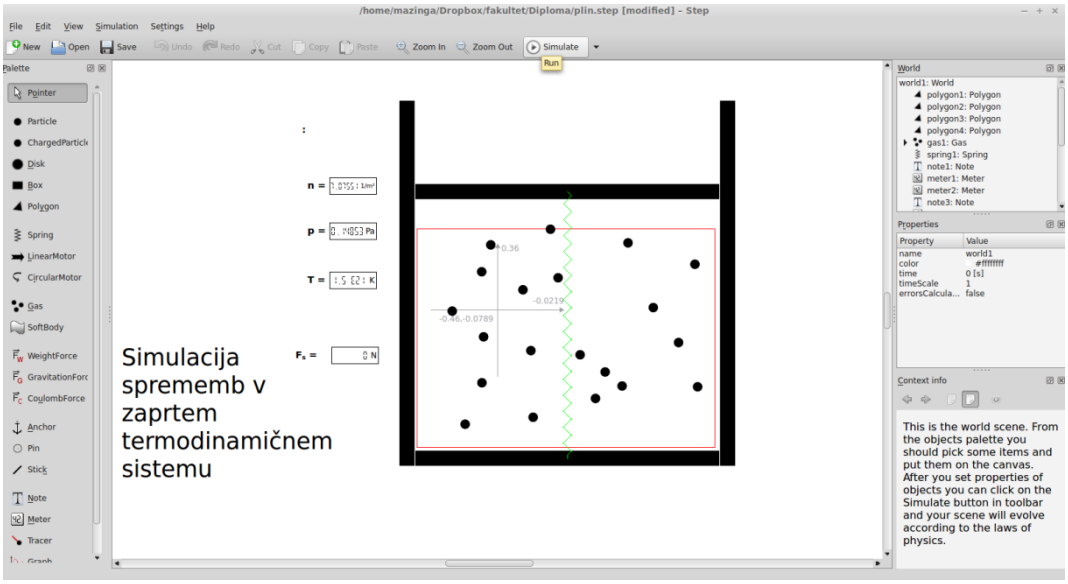
Freecad je odprtokodni programski paket za računalniško podprto konstruiranje (CAD). Zgrajen je v programskih jezikih C++ in Python ter ima vmesnik, zgrajen na podlagi Qt-knjižnic. Ima več različnih delovnih miz (angl. *workbench*), ki se razlikujejo glede na inženirsko področje in ponujajo tudi različne dodatke za podporo pri izdelavi arhitekturnih, strojniških ter mnogih drugih načrtov. Podpira 2D- in 3D-risanje ter DXF-, SVG-, STEP-, IGES- in IFC-formate. Z dodatkom Teigha File Converter lahko uvaža standardne AutoCAD-formate, kot je DWG. Ima integrirano orodje za izdelavo in uvoz skript, izdelanih v Python-u.



Slika 17: FreeCAD z delovno mizo *Architecture*

2.5.5 Step

Step je odprtokodni fizikalni simulator. Pisan je v jeziku C++ in se uporablja za 2D-simulacijo pojavov klasične mehanike ter termodinamike. Ima enostaven vmesnik in v orodjih vsebuje že programirane dodatke za obravnavanje gibanj, zunanjih sil in navorov, kinetične energije, posledic deformacij na določenem telesu itd. Ima možnosti izrisovanja grafov v realnem času.



Slika 18: *Step*

3 Python programski jezik

3.1 Programski jeziki

"Programski jezik je stroju berljiv umetni jezik, razvit z namenom, da izraža izračune, ki jih izvaja stroj oziroma računalnik." (Špegel, 1971)
Računalničar Kornad Zuse je leta 1945 dokončal pisanje prvega visokonivojskega programskega jezika, Plankalkül, ki je bil namenjen algebrskim izračunom. Od takrat do danes je zahteva po programskih jezikih eksponentno narasla. Zaradi tega danes obstaja veliko različnih programskih jezikov, ki se uporabljajo za različne namene, od strojnega programiranja (npr. zbirni jezik) do splošne rabe in se razlikujejo glede na sintakso in semantiko. Sintaksa določa strukturo stavkov programskega jezika, tako kot gramatika določa strukturo stavkov v besednem jeziku. Semantika določa pomen določene besede oz. stavka, tako kot so besede besednega jezika definirane v slovarjih. Zaradi narave diplomske naloge se bomo osredotočili le na programski jezik Python.

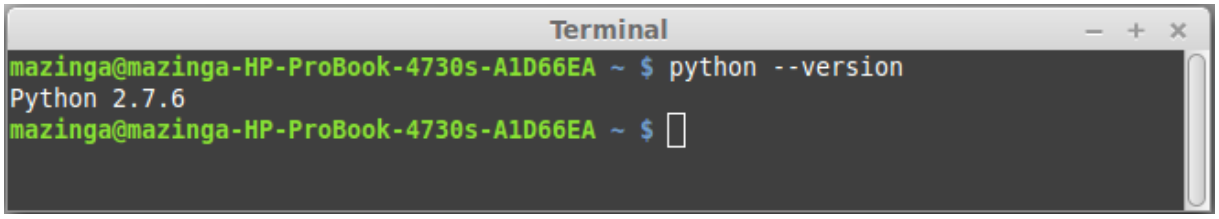
3.2 Python

Python je tolmačeni, visokonivojski programski jezik, katerega konstrukcijska filozofija temelji na lahki berljivosti in elegantnosti. Ustvaril ga je Guido van Rossum leta 1990 na Nizozemskem, poimenoval pa ga je po angleški humoristični nanizanki Leteči cirkus Montyja Pythona. Za inženirske namene je izjemno uporaben jezik, saj obstaja veliko profesionalnih Python-knjižnic za posebne namene, katere se lahko enostavno in brezplačno namestijo in s tem povečajo zmogljivost programskega okolja. Na ta način Python omogoča poenostavljanje celotnega procesa programiranja in omogoča inženirju, da se osredotoči na reševanje začetnega problema brez odvečnega dela pri zapleteni strukturi programske semantike, kot to zahtevajo jeziki, kot so C++ in Java (npr. deklariranje spremenljivk). Zaradi tega so skripte, spisane v jeziku Python, 3- do 5-krat krajše od podobnih skript v programskih jezikih, kot je Java (Python, 1997).

Slika 19: Logotip Python
Vir: https://www.python.org/community/logos/ (Pridobljeno 20. 8. 2014.)

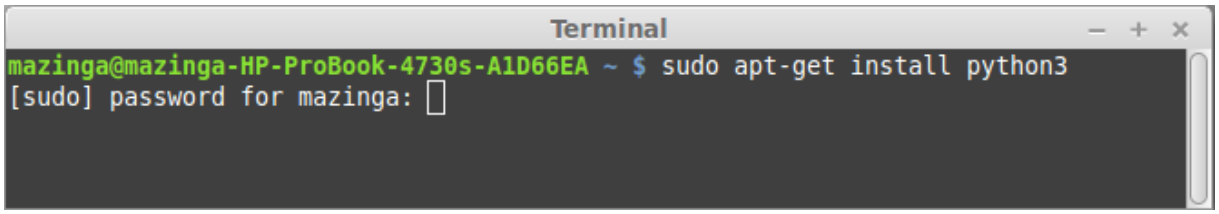
3.2.1 Namestitev Python-a

Najnovejšo različico jezika Python lahko enostavno naložimo z uradne strani na povezavi https://www.python.org/download. Tukaj imamo na voljo več različic omenjenega jezika (od 2.7 naprej) in namestitvene pakete za različne operativne sisteme, med katerimi so paketi za Linux, Unix, Mac OS X ter Windows. Python je že prednameščen v Linux Mint-u in ostalih, na Ubuntu-ju temelječih distribucij.
Trenutno verzijo lahko preverimo v terminalu BASH z ukazom “python-version”.



Slika 20: Preverjanje Python različice

V primeru, da želimo najnovejšo različico, jo lahko namestimo z ukazom “sudo apt-get install python3” v terminalu BASH.
Ukaz “sudo” nam dodeljuje pravice za namestitev (prijavimo se kot “superuser”, takrat vpišemo administratorsko geslo), “apt-get install” je ukaz za namestitev orodij prek *APT (Advanced Packaging Tool),* v zadnjem stavku pa definiramo, katero različico Pythona želimo (v našem primeru je to Python 3).



Slika 21: Nalaganje različice Python 3

3.2.2 Razvojna okolja Python

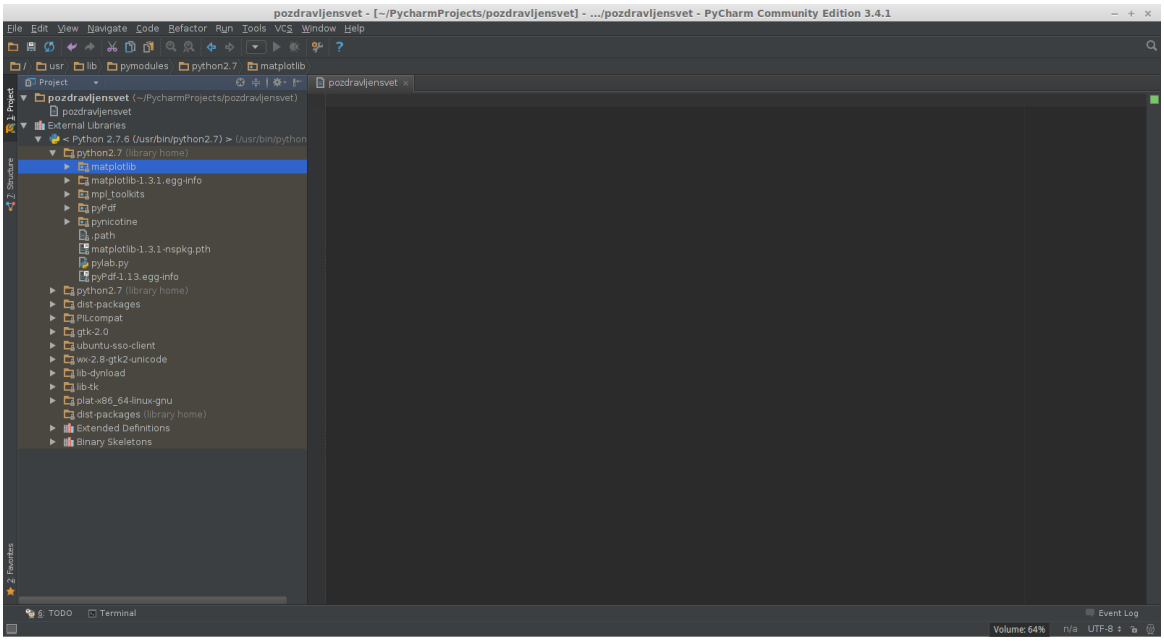
Izbira vgrajenega razvojnega okolja (angl. *IDE, Integrated Development Environment*) je odvisna od programskega jezika, v katerem želimo programirati. V Linux Mint-u in ostalih, Unix-u podobnih sistemih, ni nujno uporabljati-IDE vmesnikov, ker so programski jeziki in njihove knjižnice naloženi neposredno na sistem. To pomeni, da v primeru, ko želimo

napisati program, zadostuje uporaba urejevalnika besedila, shranjevanje datoteke z ustrezno končnico, označevanje datotek kot izvršljive (ang. *executable*) ter zagon datoteke neposredno ali iz terminala BASH.

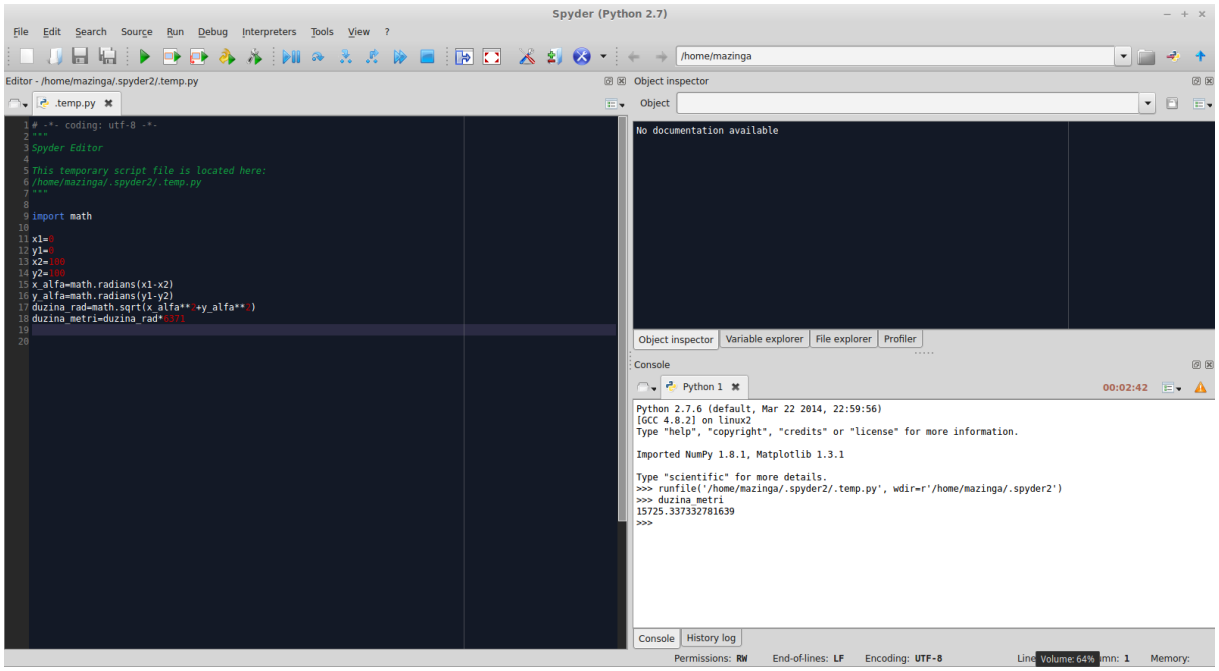
Programiranje z ustreznim razvojnim okoljem je najboljša izbira za vsakega programerja. Vsa integrirana razvojna okolja vsebujejo urejevalnik izvorne kode, tolmača (angl. *Interpreter*), razhroščevalnik (angl. *Debugger*) ter ponujajo ogromno funkcij in orodij, ki omogočajo hitrejše pisanje in lažje branje pisane kode (prepoznavanje ukazov ter barvanje besed glede na semantiko in sintakso), lažji dostop do dokumentacij knjižnic in baz podatkov ter veliko več.

Nekateri izmed najbolj uporabljanih vgrajenih razvojnih okolij za programiranje v programskemu jeziku Python so Komodo, PyCharm, Netbeans, PyDev ter Spyder, katerega bomo uporabljali za izvajanje prostorskih analiz. Spyder lahko naložimo iz terminala BASH z ukazom “sudo apt-get install spyder” ali z iskanjem v *Synaptic Manager*-ju ali *Software Manager*-ju.

Orodje, s katerim lahko namestimo knjižnice in pakete Python, je *pip*, katerega smo namestili z ukazom “sudo apt-get install python-pip” v terminalu BASH. Kot dodatek smo dodali *rope*, orodje, ki vsebuje različne funkcije za lažje pisanje, kot so avtomatsko dokončanje stavkov in izvažanje modulov v pakete (z namenom prenosa ali spreminjanja istih, za izdelavo osebnih knjižnic). Namestili smo ga z ukazom “sudo pip install rope” v terminalu BASH. Obstajajo tudi različne Python distribucije, kot je Anaconda, ki vsebujejo jezik Python z veliko dodanih knjižnic za znanstven izračun (numpy, GDAL itd.) in razvojno okolje (Anaconda ponuja Spyder).



Slika 22: Pycharm



Slika 23: Spyder

3.2.3 Primeri programa Pozdravljen Svet

Da bi na hitro razumeli enostavnost in berljivost jezika Python, lahko primerjamo izgled začetniškega računalniškega programa Pozdravljen Svet, s katerim se večina začne učiti programiranja, v različnih programskih jezikih.

C++

```
#include <iostream>
int main()
{
 std::cout << "Pozdravljen svet!\n";
}
```

C#

```
using System;
class Program
{
public static void Main(string[] args)
 {
Console.WriteLine("Pozdravljen svet!");
 }
```

Python 2

print "Pozdravljen svet!"

3.2.4 Python knjižnice

Programske knjižnice so zbirke programov in podprogramov, katere se uporabljajo za lažji in hitrejši razvoj programske opreme. Glede na namen programiranja, so na voljo številne odprtokodne Python knjižnice.

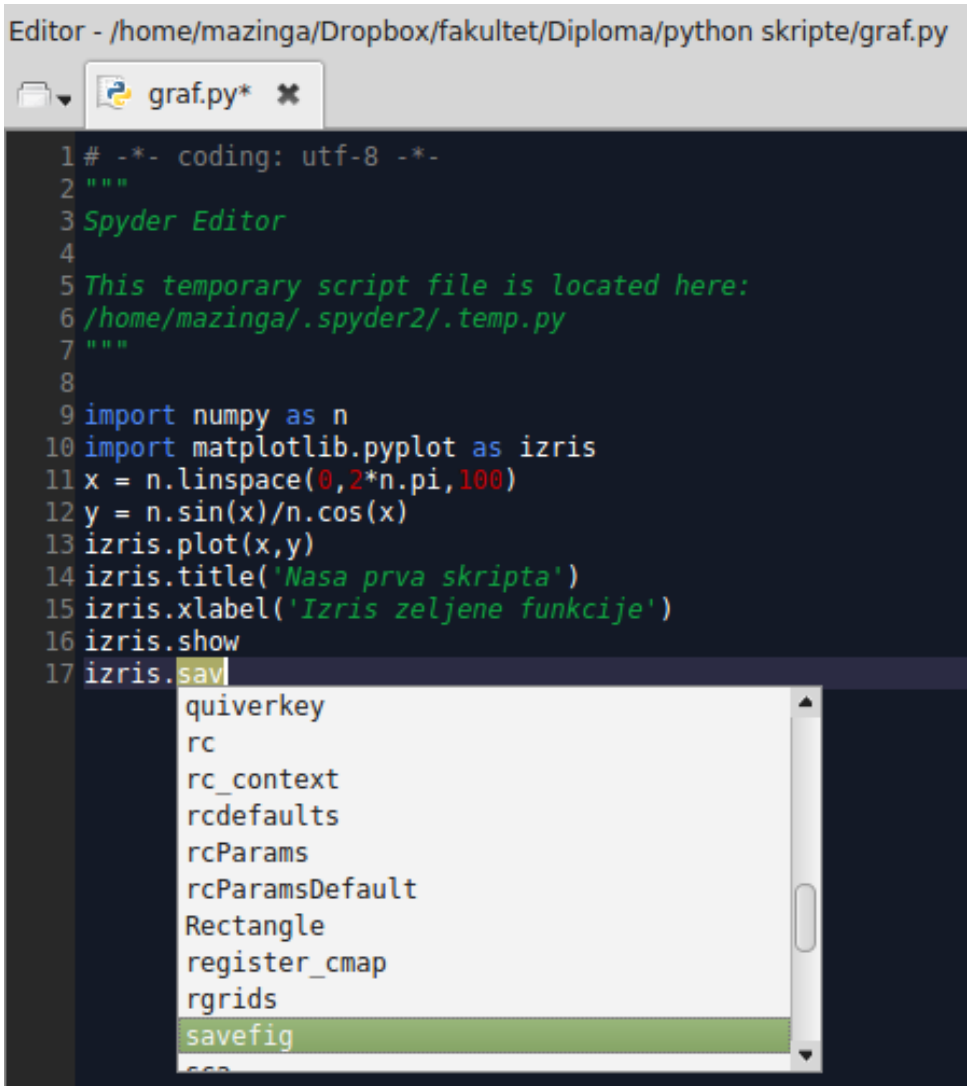
Primeri knjižnic:

- NumPy, ScientificPython in SymPy – knjižnice, ki vsebujejo funkcije za izvajanje večrazsežnih komputacij in izračunov na področjih algebre, matrične in vektorske analize, optimizacije, statistike, geometrije itd.;
- SymPy – matematične funkcije za izvajanje številnih numeričnih in simbolnih problemov (kot npr. Maxima in Mathematica);
- PySAL, Python Cartographic Library, OWSLib, GeoJSON in Rtree – knjižnice za obdelavo in analizo prostorskih podatkov;
- matplotlib.pyplot – knjižnica za izrisovanje 2D-grafov;
- Biopython – orodja za biološko kompjutacijo in bioinformatiko;
- Astropy, SunPy, Spacepy – orodja za astronomske in geofizične namene.

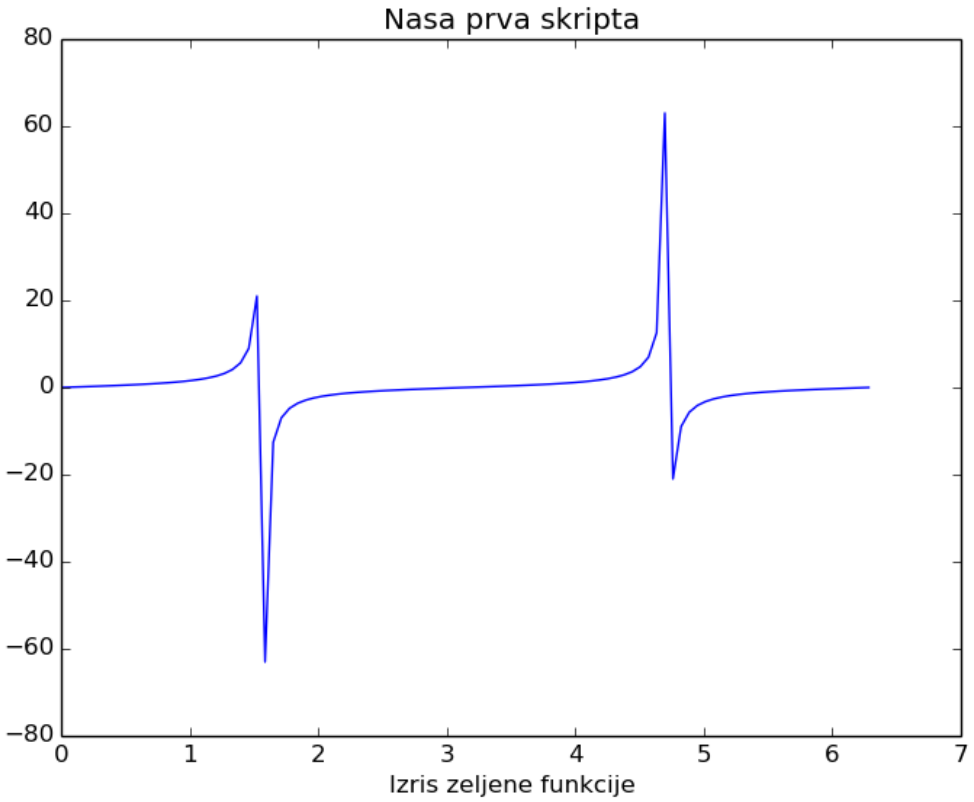
3.2.5 Primer izdelave skripte v Pythonu 2

Python loči velike in male črke in ne zahteva presledkov med števili in operatorji. Spremenljivke deklariramo s poimenovanjem, z enačajem opredelimo njihove vrednosti oz. funkcije. Naložene knjižnice uporabimo tako, da jih uvozimo z ukazom *import*, nato jih zaradi lažje nadaljnje uporabe preimenujemo v poljubno ime s funkcijo *as.*

Želeno funkcijo lahko pokličemo tako, da po vpisovanju imena knjižnice, katera jo vsebuje, damo piko ter potem ime želene funkcije. Vsaki funkciji moramo po odprtem oklepaju dodeliti spremenljivke, na podlagi katerih se bo izvajala. V primeru, da je funkcija bolj zapletena, seznam želenih spremenljivk, njihovo zaporedje in želen format lahko preberemo v dokumentaciji knjižnice. Ker smo namestili dodatek *rope*, se nam – potem ko vpišemo piko po imenu knjižnice – na pomoč odpre lista funkcij, katere ta knjižnica vsebuje.



Slika 24: Izrisovanje grafov v Spyder-ju



Slika 25: Rezultat po izvajanju skripte

4 PROSTORSKE ANALIZE S POMOČJO SKRIPT

4.1 PROSTORSKE ANALIZE

Prostorske analize opredelimo kot postopke, s pomočjo katerih obdelujemo prostorske podatke, iščemo povezave in sorodne vzorce ter ustvarjamo nove podatke oziroma posredno informacije (Šumrada, 2005, str. 322). S pomočjo tehnologije geografskih informacijskih sistemov (GIS) lahko prikazujemo, delimo in zajemamo prostorske podatke. Danes večino prostorskih analiz izvajamo s pomočjo posebnih programskih orodij in aplikacij s področja GIS, kot so QGIS, ArcGIS Desktop (ESRI), GRASS GIS itd. Ta programska orodja vsebujejo nameščena orodja za izvajanje prostorskih analiz. Orodja so najpogosteje programirana v programskih jezikih C++ ter Python.

4.2 Python skripte za izvajanje prostorskih analiz

Programska orodja, kot so ArcGIS in QGIS, vsebujejo orodja za izdelavo in dodajanje vtičnikov in skript (angl. *plugin*), kot so *QGIS Plugin Builderin ArcPy*. Programski jezik, v katerem programiramo vtičnike, je Python. Tako kot v orodjih za izdelavo vtičnikov, skripte za izvajanje programskih analiz lahko programiramo v kateremkoli Python razvojnem okolju, kot je v našem primeru Spyder. Tako kot smo v prejšnjem poglavju sprogramirali skripto za izris grafa poljubne funkcije, bomo na enak način sprogramirali nekaj poljubnih skript, s katerim bomo izvedli prostorske analize.

4.2.1 Razdalja

Pri računanju razdalj med dvema točkama moramo upoštevati veliko različnih dejavnikov, ki so ključni za izbiro metodologije računanja. To pomeni, da moramo pri izračunu razdalje definirati geometrijske lastnosti prostora, v katerem iščemo odgovore. Zato so postopki za računanje razdalje različni za točke v 2-razsežnem prostoru, za točke na določenem sferoidu ali rotacijskem elipsoidu (različni datumi). Postopek se še dodatno spreminja s projekcijami, v katerih je prostor definiran. Večina GIS orodij uporablja standardno evklidsko formulo (Drobne, 2012, str. 300), za izračun večjih razdalj pa se lahko poslužimo Haversinove formule oziroma algoritma:

```
import numpy as n
#koordinate
x1=50
```

```
y1=50
x2=80
y2=80
#priprava podatkov za formulo
R=6371
y1_rad=n.radians(y1)
y2_rad=n.radians(y2)
delta_x=n.radians(x1-x2)
delta_y=n.radians(y1-y2)
#Haversine-ova formula
a=n.sin(delta_y/2)**2+n.sin(delta_x/2)**2 \
*n.cos(y1_rad)*n.cos(y2_rad)
c=2*n.arcsin(n.sqrt(a))
razdalja=c*R
print 'razdalja je', razdalja, 'kilometrov'
```

4.2.2 Klasifikacija podatkov

4.2.2.1 NDVI

S pomočjo uporabe različnih svetlobnih spektrov lahko na določenih posnetkih vidimo informacije, katerih ne moremo zaznati s prostim očesom. Eden pogostih primerov je uporaba bližnjega infrardečega spektra za pridobitev vegetacijskega indeksa določenega območja. Eden najbolj uporabljanih indeksov za pridobitev vegetacije je normirani diferencialni vegetacijski indeks NDVI (Oštir, 2006, str. 165).
NDVI predstavlja razmerje med razliko infrardečega in rdečega kanala in njuno vsoto (prav tam):

$$NDVI = \frac{IR - R}{IR + R}$$

Za praktični primer smo uporabili prostorske podatke in metodologijo, ki ju pri izdelovanju skripte ponuja knjiga *Learning Geospatial Analysis with Python* (Lawhead, 2013).

Slika 26: Vhodni posnetek
Vir: http://geospatialpython.googlecode.com/files/NDVI.zip (Pridobljeno 8. 9. 2014.)

Za izvajanje analize smo uporabili GDAL (Geospatial Data Abstraction Library), eno od najbolj uporabljanih knjižnic za izvajanje rastrskih in vektorskih prostorskih analiz. GDAL smo namestili tako, da smo uporabili naslednje ukaze v terminalu BASH: "sudo apt-get install gdal-bin && sudo apt-get install python-gdal".

Algoritem, ki smo ga uporabili za izračun NDVI, je povzet po Lawhead, 2013, ter dopolnjen za lažje razumevanje:

```
import gdal, gdalnumeric, ogr, osr
import Image, ImageDraw

#pretvorba v gdalnumeric sliko
def imageToArray(i):
  a=gdalnumeric.numpy.fromstring(i.tostring(), 'b')
  a.shape=i.im.size[1], i.im.size[0]
  return a
#izracun koordinat na podlagi lokacije pikslov
def world2Pixel(geoMatrix, x, y):
  ulX=geoMatrix[0]
```

```
  ulY=geoMatrix[3]
  xDist=geoMatrix[1]
  yDist=geoMatrix[5]
  rtnX=geoMatrix[2]
  rtnY=geoMatrix[4]
  pixel=int((x-ulX)/xDist)
  line=int((ulY-y)/xDist)
  return (pixel, line)
#uvoz in izvoz datotek
source="/home/mazinga/Desktop/NDVI/farm.tif"
target="/home/mazinga/Desktop/NDVI/ndvi.tif"
srcArray=gdalnumeric.LoadFile(source)
srcImage=gdal.Open(source)
geoTrans=srcImage.GetGeoTransform()
r=srcArray[1]
ir=srcArray[2]
#rasterizacija: izdelava sloja, dolocanje robov, velikosti pikslov
field=ogr.Open("/home/mazinga/Desktop/NDVI/field.shp")
lyr=field.GetLayer("field")
poly=lyr.GetNextFeature()
minX, maxX, minY, maxY = lyr.GetExtent()
ulX, ulY=world2Pixel(geoTrans, minX, maxY)
lrX, lrY=world2Pixel(geoTrans, maxX, minY)
pxWidth=int(lrX-ulX)
pxHeight=int(lrY-ulY)
#nova, rezana slika
clipped=gdalnumeric.numpy.zeros((3,pxHeight,pxWidth), gdalnumeric.numpy.uint8)
rClip=r[ulY:lrY, ulX:lrX]
irClip=ir[ulY:lrY, ulX:lrX]
#matrika vrednosti nove slike
geoTrans=list(geoTrans)
geoTrans[0]=minX
geoTrans[3]=maxY
points=[]
pixels=[]
geom=poly.GetGeometryRef()
```

```
pts=geom.GetGeometryRef(0)
#pretvorba geometrijskih lastnosti v matrike pikslov
for p in range(pts.GetPointCount()):
  points.append((pts.GetX(p), pts.GetY(p)))
for p in points:
  pixels.append(world2Pixel(geoTrans, p[0], p[1]))
#izdelava rasterske, polygonske slike
rasterPoly=Image.new("L", (pxWidth, pxHeight),1)
rasterize=ImageDraw.Draw(rasterPoly)
rasterize.polygon(pixels, 0)
mask=imageToArray(rasterPoly)
#deljenje na r in infrar spektre
rClip=gdalnumeric.numpy.choose(mask,(rClip, 0)).astype(gdalnumeric.numpy.uint8)
irClip=gdalnumeric.numpy.choose(mask,(irClip, 0)).astype(gdalnumeric.numpy.uint8)
#uporaba formule NDVI
#*1.0 prevarja vrednost v plavajočo vejico;
#+1.0 preprecuje napake z deljenjem z niclo
gdalnumeric.numpy.seterr(all="ignore")
ndvi=1.0*(irClip-rClip)/irClip+rClip +1.0
ndvi=gdalnumeric.numpy.nan_to_num(ndvi)
gdalnumeric.SaveArray(ndvi, target, format="GTiff", prototype=source)
```

Slika 27 prikazuje rezultat izračuna NDVI obravnavnega območja.

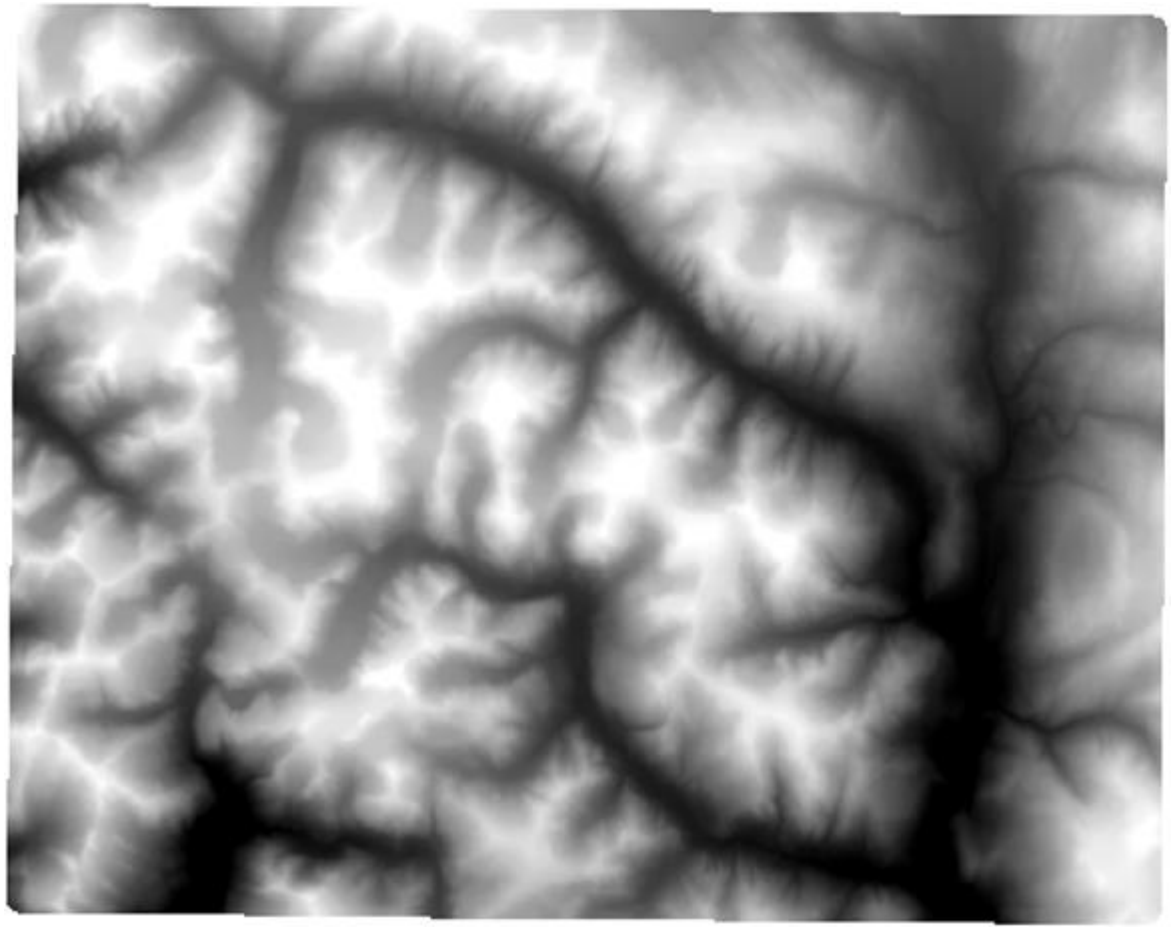
Slika 27: Rezultat obdelave in NDVI območja (format GeoTiff)

4.2.3 Analize morfologije ploskve

4.2.3.1 Senčenje

Na podlagi podatkov digitalnega modela reliefa (angl. *Digital Elevation Model*, *DEM*) lahko izračunamo senčeno podobo določenega območja. Podatki o reliefu so trirazsežne narave, zato je v številnih primerih iz neobdelane karte moč "prebrati" stvarno stanje terena na podlagi pogleda dvorazsežne slike. Zaradi lažje obravnave območja izvajamo postopek senčenja, po katerem se vrednosti višin klasificirajo v razrede po različnih svetlobnih spektrih. Kontrast le-teh pa ponuja precej bolj pregledno stanje terena. Barve temnih spektrov prikazujejo globlje, barve svetlih spektrov pa višje predele.

Za izvajanje analize smo uporabili modul *linecache* in knjižnico *numpy*. Za praktični primer smo uporabili podatke in metodologijo, katere pri izdelovanju skripte ponuja knjiga *Learning Geospatial Analysis with Python.*



Slika 28: Vhodni posnetek za senčenje

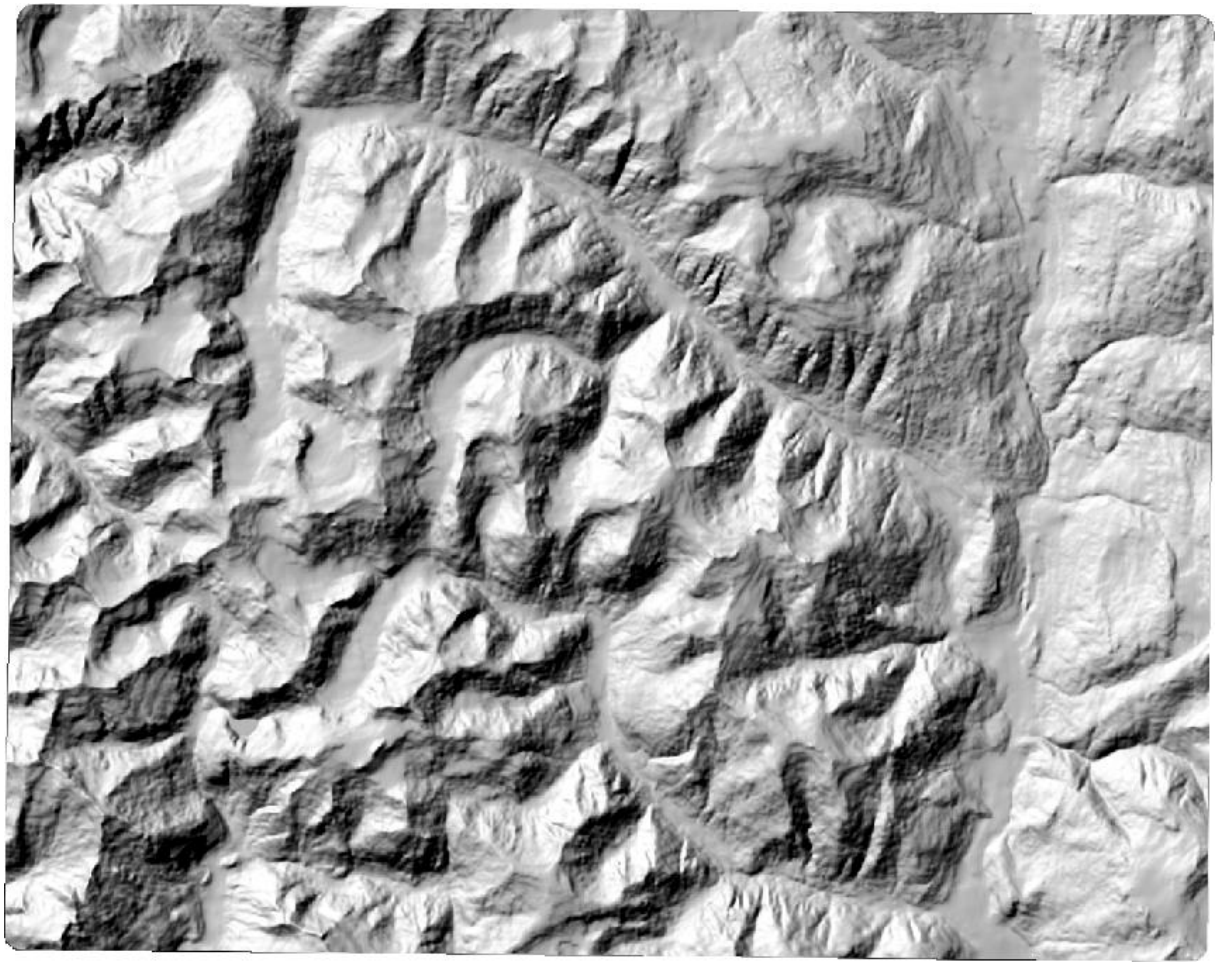
Vir: https://geospatialpython.googlecode.com/files/dem.zip (Pridobljeno 8. 9. 2014.)

Skripta za izvajanje senčenja je izpisana spodaj, na sliki 29 pa je prikazan rezultat senčenja. V skripti smo upoštevali naslednje parametre: azimut od 315° ter višinski kot od 45°.

```
from linecache import getline

import numpy as np

source="/home/mazinga/dem/dem.asc"
slopegrid="/home/mazinga/dem/slope.asc"
aspectgrid="/home/mazinga/dem/aspect.asc"
shadegrid="/home/mazinga/dem/relief.asc"
#smer in kot Sonca
azimuth=315.0
altitude=45.0
z=1.0
scale=1.0
NODATA=-9999
#orodja za pretvorbo
```

```
deg2rad=np.pi/180
rad2deg=180/np.pi
hdr=[getline(source, i) for i in range (1,7)]
values=[float(h.split(" ")[-1].strip()) for h in hdr]
cols, rows, lx, ly, cell, nd=values
xres=cell
yres=cell*-1
arr=np.loadtxt(source, skiprows=6)
window=[]
for row in range(3):
for col in range(3):
window.append(arr[row:(row+arr.shape[0]-2), \
col:(col+arr.shape[1]-2)])
x=((z*window[0]+z*window[3]+z*window[3]+z*window[6])-\
(z*window[2]+z*window[5]+z*window[5]+z*window[8]))/(8.0*xres*scale);
y=((z*window[6]+z*window[7]+z*window[7]+z*window[8]) \
-(z*window[0]+z*window[1]+z*window[1]+z*window[2]))\
/(8.0*yres*scale)
slope=90.0-np.arctan(np.sqrt(x*x+y*y))*rad2deg
aspect=np.arctan2(x,y)
shaded=np.sin(altitude*deg2rad)*np.sin(slope*deg2rad)\
+np.cos(altitude*deg2rad)*np.cos(slope*deg2rad)\
*np.cos((azimuth-90.0)*deg2rad-aspect);
shaded=shaded*255


header="ncols %s\n" % shaded.shape[1]
header+="nrows %s\n" % shaded.shape[0]
header+="xllcorner %s\n" % (lx+(cell*(cols-shaded.shape[1])))
header+="yllcorner %s\n" % (ly+(cell*(rows-shaded.shape[0])))
header+="cellsize %s\n" % cell
header+="NODATA_value %s\n" % NODATA


for pane in window:
slope[pane == nd]= NODATA
aspect[pane == nd]= NODATA
shaded[pane == nd]= NODATA
with open(slopegrid, "wb") as f:
f.write(header)
np.savetxt(f, slope, fmt="%4i")
with open(aspectgrid, "wb") as f:
f.write(header)
np.savetxt(f, aspect, fmt="%4i")
with open(shadegrid, "wb") as f:
f.write(header)
```

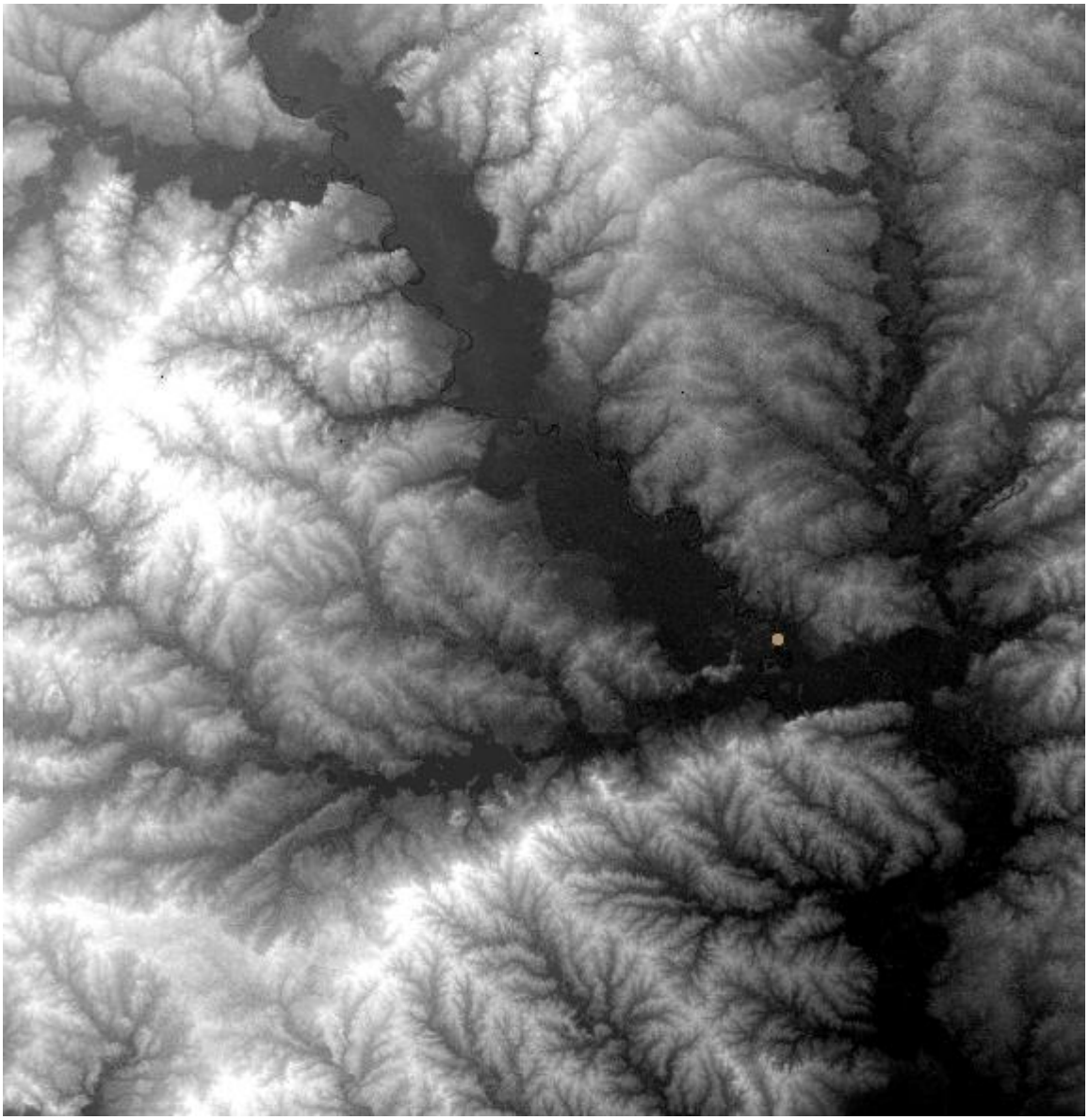
```
np.savetxt(f, shaded, fmt="%4i")
```



Slika 29: Rezultat senčenja

4.2.3.2 Izračun poplavnih območij

Na podlagi digitalnega modela reliefa lahko – z upoštevanjem rekurzivne formule za širitev vrednosti na podlagi vrednosti pikslov – pridobimo enostavno simulacijo in sliko poplavnega območja. V tem primeru posamezne vrednosti širimo na podlagi višinskih razlik, pri čemer pa geoloških lastnosti območja ne upoštevamo. Algoritem, podoben algoritmu izračuna poplavnih območij, se uporablja pri različnih grafičnih programskih orodjih, kot je denimo GIMP, za izvajanje funkcije “zapolni s kanglico” (ang. *Bucket fill*).



Slika 30: Območje za simulacijo poplavnega območja

Vir: https://geospatialpython.googlecode.com/files/FloodFill.zip (Pridobljeno 8. 9. 2014)

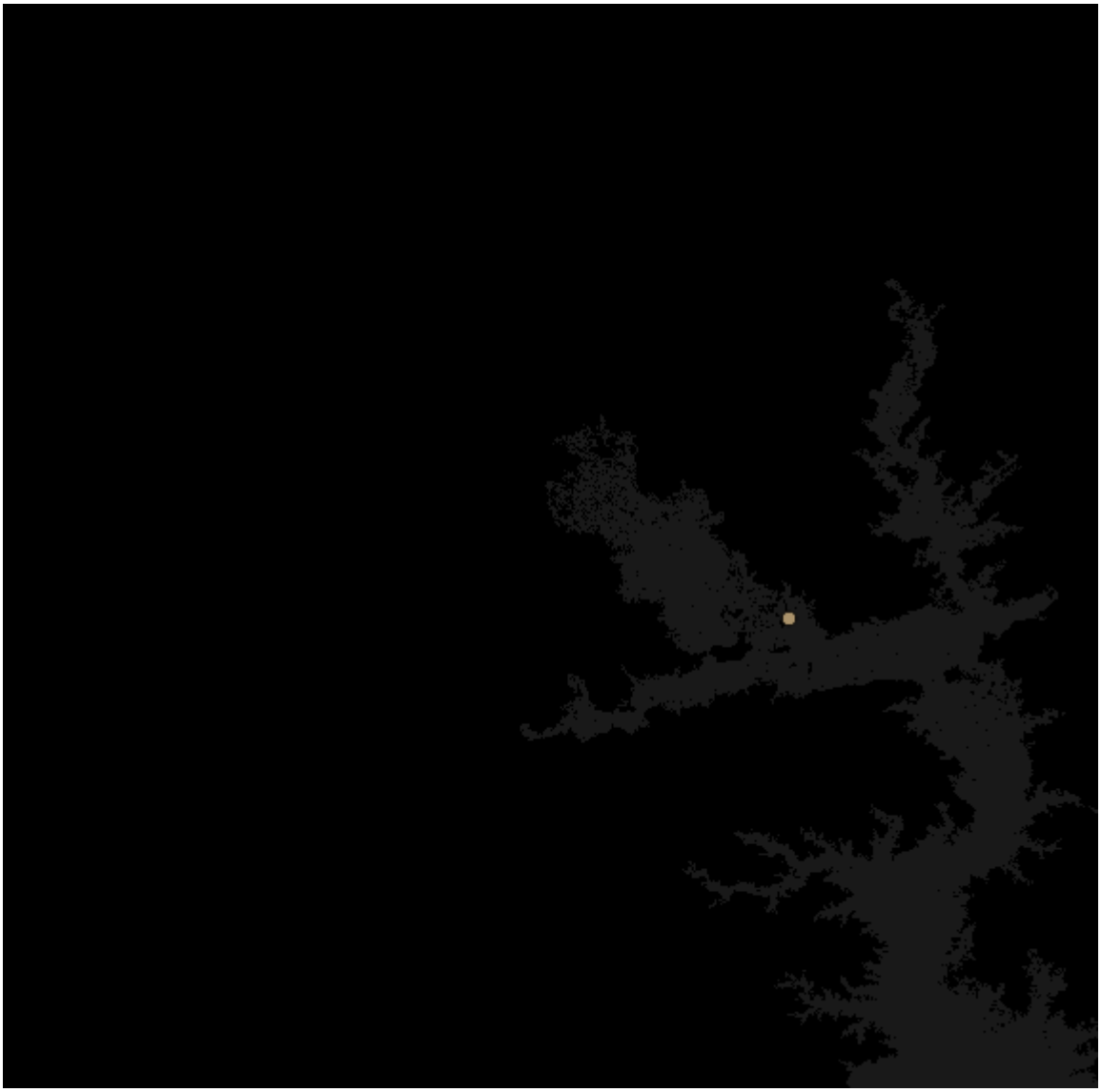
Za izvajanje analize smo uporabili modul *linecache* in knjižnico *numpy*. Za praktični primer delovanja skripte smo uporabili prostorske podatke in metodologijo iz knjige *Learning Geospatial Analysis with Python*, zapis skripte je spodaj. V skripti smo nastavili vrednosti parametra lokacije, oz točke, iz katere se bo simulirala poplava (sx in sy):

```
import numpy as np

from linecache import getline

def floodFill(c,r,mask):
filled=set()
```

```
fill=set()
fill.add((c,r))
width=mask.shape[1]-1
height=mask.shape[0]-1
flood=np.zeros_like(mask, dtype=np.int8)
while fill:
x,y=fill.pop()
if y== height or x == width or x<0 or y<0:
continue
if mask[y][x]==1:
flood[y][x]=1
filled.add((x,y))
west=(x-1,y)
east=(x+1,y)
north=(x, y-1)
south=(x,y+1)
if not west in filled:
fill.add(west)
if not east in filled:
fill.add(east)
if not north in filled:
fill.add(north)
if not south in filled:
fill.add(south)
return flood
#simulacija poplave
source="/home/mazinga/Desktop/FloodFill/terrain.asc"
target="/home/mazinga/Desktop/FloodFill/flood.asc"
print "Odpiram sliko..."
img=np.loadtxt(source, skiprows=6)
print "Slika nalozena"
a=np.where(img<70,1,0)
print "Slika pretvorjena v matriko"


hdr=[getline(source, i) for i in range (1,7)]
values=[float(h.split(" ")[-1].strip()) for h in hdr]
cols, rows, lx, ly, cell, nd = values
xres=cell
yres=cell*-1
#poplavna tocka
sx=2582
sy=2057
print "Zacetek poplave"
fld= floodFill(sx,sy,a)
```

```
print "Konec poplave"
header=""
for i in range(6):
header+=hdr[i]
print "Shranjujem grid"
with open(target, "wb") as f:
f.write(header)
np.savetxt(f, fld, fmt="%1i")


print "Koncano!"
```

Slika 31 prikazuje izračun poplavnega območja.



Slika 31: Izračunano poplavno območje

4.3 Izvajanje skript Python v OS Linux in v Windows

Python je programski jezik, ki deluje na različnih sistemskih platformah. V ta namen smo zadnjo skripto – skripto za računanje poplavnih območij – izvajali v operacijskem sistemu Linux Mint 17 Qiana (L) ter v Windows 8.1 (W). Oba sistema sta bila sveže naložena na isti računalnik, posodobljena do zadnjih posodobitev (8.9.14), brez sistemskih sprememb ali dodanih programov, razen Python distribucije Anaconda s Spyder-jem na obeh. Edina sprememba v skripti je bila lokacija vhodnih datotek na sistemu Windows. Windows OS ima namreč drugačno hierarhijo datotek, čeprav je bila datoteka v obeh sistemih na namizju (ang. *Desktop*).

Zaradi primerjave delovanja skripte v obeh testiranih okoljih smo izvajali meritev časa, v

katerem se skripta izvede. Čas smo merili z modulom *time* in njegovo funkcijo *time.time*, katero smo dodali v izvorno skripto izračuna poplavnega območja. Spodaj je izpis dodane kode.

```
import numpy as np
from linecache import getline
import time

start=time.time()

def floodFill(c,r,mask):
#poteka enaka skripta, na koncu pa dodamo:
print "Izvajalo se", time.time() - start, "sekund."
```

Izvedli smo po pet meritev v vsakem testiranem okolju oziroma na vsakem testiranem operacijskem sistemu. Rezultati meritev so zapisani v preglednici 1.

Preglednica 1: Rezultati meritev časa izvajanja skripte za izračun poplavnih območij

meritev	L Linux Mint 17 Qiana [s]	W Windows 8.1 [s]
1	20,7521	22,7409
2	20,4340	22,2549
3	20,4142	22,5320
4	20,3323	22,1119
5	19,8911	22,9409

Povprečji meritev sta $\bar{L} = 20{,}3647$ in $\bar{W} = 22{,}5161$, standardna odklona meritev sta $s_L^* = 0{,}3093$ in $s_W^* = 0{,}3402$, koeficienta variacije pa $KV_L = 0{,}0152$ in $KV_W = 0{,}0151$. Iz rezultatov je razvidno, da je izvajanje testne skripte v Linux Mint 17 Qiana hitrejše kot v Windows 8.1. Če primerjamo najhitrejše čase obeh, opazimo, da se je skripta za izračun poplavnih območij v Linux Mint-u izvedla 2,2208 sekund prej, kar je približno 10 % hitreje. V testnem primeru so bili uporabljani podatki, ki ne zasedajo veliko računalniškega prostora,

prav tako pa je tudi algoritem izračuna poplavnega območja – v primerjavi z nekaterimi kompleksnimi algoritmi, ki se izvajajo vsakodnevno v GIS – enostaven. V primeru, da bi izvajali take algoritme za obdelavo večje količine podatkov, bi se časi iz sekund pretvorili v minute – v takih primerih pa je 10-odstotna razlika občutna. V tem primeru govorimo tudi o primerjavi brezplačnega in prosto dostopnega ter komercialnega izdelka.

5 VREDNOTENJE REZULTATOV

Tako kot v programih z vgrajenimi ukazi, lahko datoteke, shranjene v standardnih formatih prostorskih podatkov, pretvarjamo v poljubne matrične formate, s pomočjo različnih programskih jezikov, kot je Python. Visokonivojski programski jezik Python omogoča širok spekter uporabe v inženirstvu. Za razliko od programskih jezikov C++ in Java je veliko lažji za uporabo ter v trenutkih iskanja rešitve, za določene probleme, ponuja hitre rešitve. Ne glede na operacijski sistem, v katerem se uporablja, Python deluje na podoben način.

V diplomski nalogi smo s pomočjo znanstvenih knjižnic, katere vsebujejo znanstvene Python distribucije, kot je Anaconda (GDAL, geos, numpy itd.), izvajali različne izračune. Knjižnice in module, s katerimi smo izvajali obdelavo vhodnih podatkov, smo namestili s pomočjo orodij, primer je *pip*. Po izpisu ustrezne funkcije smo pretvorili vhodne podatke v različne datoteke formatov GeoTIFF, ASCII, *.dbf (zbirke podatkov), *.shp in *.shx (vektorski podatki), katere lahko uporabimo v različnih GIS-orodjih, na primer, v QGIS. Standardni zapis podatkov omogoči tudi neposredno uporabo le-teh v različnih operacijskih sistemih; mi smo testirali podatke in Phython skripte v operacijskem sistemu Linux Mint 17 Qiana ter v Windows 8.1. Rezultati meritev časa izvedbe testirane skripte izračuna poplavnega območja v testiranih okoljih (OS) so pokazali, da je izvedba v Linux Mintu za ca. 10 % hitrejša od delovanja skripte v Windows 8.1 okolju.

6 ZAKLJUČEK

Geodeti lahko različne prostorske analize, ki jih vsakodnevno izvajamo v okoljih GIS, izvedemo na hiter način, brez dodatne potrebe po definiranju ukazov ter brez odvečnega programiranja, temveč le s kombiniranjem obstoječih funkcij iz obstoječih knjižnic. Za izvedbo prostorskih analiz ne potrebujemo komercialnih orodij, ampak lahko želene postopke izvajamo v brezplačnih operacijskih sistemih in razvojnih okoljih ter naše izdelke (npr. skripte) delimo z drugimi inženirji in tako gradimo boljšo skupnost. Stroški programske opreme predstavljajo velik del začetnih stroškov, tako lahko uporaba zastonjskih in odprtokodnih rešitev bistveno olajša začetke samostojnega podjetništva.

V tej diplomski nalogi smo pokazali, da je izvajanje skript Python v Linux OS zanesljivo in gospodarno. Dodatni prednosti uporabe takšnih skript sta svoboda in prilagodljivost, ki nastopata v odprtokodnem svetu, saj lahko izdelke spreminjamo s spoštovanjem do odprtih standardov in na ta način uporabljamo ter delimo izdelke neodvisno od komercialnih trendov, s čistim fokusom le na uporabi in bogatenju inženirske skupnosti.

Hiter razvoj medmrežja in veliki porast GNU/Linux skupnosti je pripeljal do velikega napredka v razvoju jedra Linux ter na njemu temelječih distribucij. Velika raznolikost programerjev (sistemski, omrežni, dizajnerski itd.) je poleg funkcionalnosti doprinesla tudi estetsko popolnost grafičnih vmesnikov in namizij ter tako privabila vse večje število uporabnikov. Odprta koda je prihodnost računalniškega sveta. V koraku s tem moramo biti tudi študenti in inženirji, zato je smiselno, da poleg “vsiljenih” ter standardnih orodij preizkusimo čim več alternativnih rešitev. Le s tovrstnimi izkušnjami si lahko ustvarimo svoje mnenje ter tako spoznamo in izberemo boljše. Hkrati pa bi se z uvedbo odprtokodnih operacijskih sistemov, pisarniških ter inženirskih orodij, tako kot vsaka institucija, tudi naša fakulteta oziroma univerza izognila velikemu delu stroškov in ta finančna sredstva bi lahko namenili za druge, strokovne namene.

VIRI

Boyer. The 360 Revolution. http://www01.ibm.com/software/os/systemz/pdf/360Revolution_0406.pdf (Pridobljeno 25. 7. 2014.)

Datotečni sistem Linux. http://lgm.fri.uni-lj.si/OS/LINUX_FILE_SYSTEM/os_linux_file_system.pdf (Pridobljeno 25. 7. 2014.)

Debian. https://www.debian.org (Pridobljeno 6. 8. 2014.)

Distrowatch. http://distrowatch.com/search.php?status=All (Pridobljeno 28. 8. 2014.)

Drobne, S. 2012. Metode prostorskih analiz v GIS. Ljubljana, Fakulteta za gradbeništvo in geodezijo.

IBM, 1965. IBM Operating System/360 Concepts and Facilities. http://bitsavers.trailing-edge.com/pdf/ibm/360/os/R01-08/C28-6535-0_OS360_Concepts_and_Facilities_1965.pdf (pridobljeno: 25. 7. 2014.)

Lawhead, J. 2013. Learning Geospatial Analysis with Python. PACKT publishing: 364 str.

Lewis, Peter J. 1986. Peripherals; UNIX and MS-DOS: Dueling For Dominance in Computers. http://www.nytimes.com/1986/05/13/science/peripherals-unix-and-ms-dos-dueling-for-dominance-in-computers.html (Pridobljeno: 25. 7. 2014.)

MaFiRa-Wiki. Operacijski sistemi. http://wiki.fmf.uni-lj.si/wiki/Operacijski_sistemti (Pridobljeno 25. 7. 2014.)

Oštir, K. 2006. Daljinsko zaznavanje. Ljubljana: Ljubljana, Založba ZRC.

Python. https://www.python.org/doc/essays/comparisons/ (Pridobljeno 20. 8. 2014.)

Shankland, S. 2014. Linux development by the numbers: Big and getting bigger. http://www.cnet.com/news/linux-development-by-the-numbers-big-and-getting-bigger (Pridobljeno 30. 7. 2014.)

Scott, J. 2014. Android set to reach one billion users in 2014. http://www.computerweekly.com/news/2240212085/Android-set-to-reach-one-billion-users-in-2014 (Pridobljeno 31. 7. 2014.)

Silberschatz, A., Galvin, Peter B., Gagne, G. 2009. Operating System Concepts. 8th edition. Hoboken, New Jersey: Wiley.

Špegel, M. 1971. Programski jeziki. V: Spiller-Muys, Franc, Elektronski računalniki, Ljubljana: Elektrotehniška zveza Slovenije.

Šumrada, R. 2005. Tehnologija GIS. Ljubljana, Fakulteta za gradbeništvo in geodezijo.

Ubuntu. https://help.ubuntu.com/community/Installation/SystemRequirements (Pridobljeno 5. 8. 2014.)

Vaughan-Nichols, Stephen J. 2013. To the space station and beyond with Linux. http://www.zdnet.com/to-the-space-station-and-beyond-with-linux-7000014958/ (Pridobljeno 6. 8. 2014.)

Vaughan-Nichols, Stephen J. 2014. Linux dominates supercomputers as never before. http://www.zdnet.com/linux-dominates-supercomputers-as-never-before-7000030890/ (Pridobljeno 31. 7. 2014.)

Enigma machine. 2014. http://en.wikipedia.org/wiki/Enigma_machine (Pridobljeno 5. 8. 2014.)

Gee-H (navigation). 2014 http://en.wikipedia.org/wiki/Gee-H_(navigation) (Pridobljeno 5. 8. 2014.)

History Of Unix. 2014 http://en.wikipedia.org/wiki/History_of_Unix (Pridobljeno 25. 7. 2014.)

Hydrophone. 2014 http://en.wikipedia.org/wiki/Hydrophone (Pridobljeno 5. 8. 2014.)

IBM 704. 2014 http://en.wikipedia.org/wiki/IBM_704 (Pridobljeno 25. 7. 2014.)

Lorenz cipher. 2014

http://en.wikipedia.org/wiki/Lorenz_cipher (Pridobljeno 5. 8. 2014.)