SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Tomislav Gelo

Automatizacija aktivnosti u naredbenom retku pomoću skripti

ZAVRŠNI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Tomislav Gelo

Matični broj: Z-40818/11-Izv

Studij: Primjena informacijske tehnologije u poslovanju

Automatizacija aktivnosti u naredbenom retku pomoću skripti

ZAVRŠNI RAD

Mentor/Mentorica:

Dr. sc. Zlatović Miran

Varaždin, prosinac 2021.

Tomislav Gelo

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Rad se bavi temom automatizacije u operacijskom sustavu Linux, koristeći naredbeni redak. Tema je razrađena u poglavljima koja se nadovezuju jedno na drugo, u stilu uputa za korištenje. Prvo prolazimo osnovne podatke operacijskog sustava i zatim koncept automatizacije. Prvi dio središnjeg rada se odnosi na ljusku, definiramo što je, a fokus stavljamo na pristupamo i korištenje. Zatim pažnju posvećujemo naredbama ljuske u naredbenom retku, prolazimo osnove korištenja i postepeno ih gradimo prema sve kompleksnijim primjerima. Prema kraju rada se okrećemo prema skriptama, kreiranju i korištenju. Fokus je na korištenju prije navedenih naredbi i logike kako bi naša skripta imala uporabnu funkciju. Za sam kraj ostaje projektni rad koji obuhvaća napisana poglavlja te se demonstrira moć i potencijal skripti da se uspije u naumu automatizacije.

Ključne riječi: Linux; Automatizacija; Naredbeni redak; Skripte; Ljuska; Bash; Skriptiranje

Sadržaj

1. Uvod

Ovim radom ću pokušati pokazati mogućnosti automatizacije na Linux operacijskom sustavu koristeći skripte i naredbeni redak. U današnje vrijeme zahtjevi i složenost posla rastu paralelno s potrebom da se taj posao brzo odradi. Upravo je zato automatizacija potrebna, mnoge se radnje mogu pojednostaviti i/ili ubrzati ako se smanji broj koraka i čimbenika u njima. Skriptiranje je jako jednostavno za naučiti, pogotovo na Linuxu koji ima pregršt opcija za pomoć korisnicima. Jednom kada se pohvataju osnove brzo se može shvatiti da se čak i najjednostavnije radnje mogu ubrzati, pogotovo kada se izbaci grafičko sučelje iz upotrebe. Uzeo sam ovu temu iz dva razloga:

1. potreba za automatizacijom redovito raste te vidim veliki potencijal u njoj
2. radim kao sistemski administrator te mi tema nije strana i ovo je savršena prilika da proširim i nadopunim svoje znanje o automatizaciji.

Kroz rad ću pokušati približiti osnove korištenja ljuske i naredbenog retka te pokazati koliko je jednostavno kreirati svoju prvu skriptu. Veliki fokus ću staviti na primjere i praktični prikaz nego na teoriju.

2. Linux i automatizacija

Kao što sam naslov nalaže, tema ovog rada je automatizacija u Linux operacijskom sustavu. Automatizaciju ćemo postići koristeći naredbeni redak te kreiranjem skripti koje će služiti kao praktičan primjer korištenja ljuske (eng. *shell*). Samu temu ću razraditi kroz niz osnovnih i naprednijih naredbi te primjerima kako se koriste i čemu služe. Kako bi uspjeli u tom naumu koristiti ćemo određene alate i aplikacije: Hyper-V, Sublime, Ubuntu 20.04 LTS, Naredbeni redak.

Prije nego što krenemo u kreiranje skripti i pisanje naredbi, definirati ćemo što je automatizacija te ćemo u kratko objasniti Linux i kako funkcionira.

2.1. Automatizacija

Prema Međunarodnom Društvu za Automatizaciju (eng. *International Society of Automation* - ISA) automatizaciju možemo definirati kao:

- tehniku automatskog rada uređaja, procesa ili sustava
- stvaranje i primjenu tehnologije za praćenje i kontrolu proizvodnje i isporuke proizvoda i usluga

(What is Automation? - ISA, bez dat.)

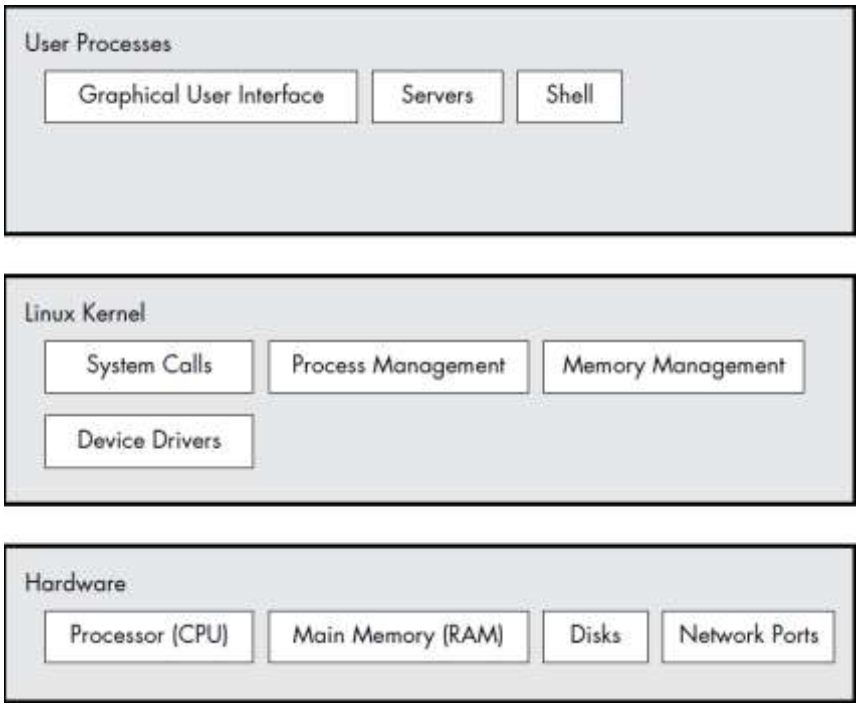
Automatizaciju možemo definirati kao izvršavanje neke radnje s minimalnim ili bez prisustva izvršitelja, pogotovo u grani informatike gdje se hrpa kompleksnih zadataka redovito izvršava u pozadini bez da korisnik ima ikakvo saznanje o tome.

2.2. Linux

Linux je operacijski sustav otvorenog-koda (eng. *open-source*) koji radi na bazi UNIX-a. Kreiran 1991. od strane Linusa Torvaldsa, Linux je nastao kao njegov privatni projekt koji je s vremenom prerastao u projekt globalnih proporcija. Linux se koristi gotovo svugdje; serveri, pametni mobiteli, tableti, općenito internet stvari (eng. *Internet of Things*), itd. Baš zbog svoje rasprostranjenosti je Linux savršeni kandidat za automatizaciju hrpe svakodnevnih procesa i zadataka. Da bi mogli automatizirati nešto na Linuxu prvo moramo shvatiti kako radi, a da bi razumjeli kako Linux radi podijeliti ćemo ga u tri dijela:

- Računalna oprema (eng. *hardware*)
- Linux jezgra (eng. *kernel*)

- Korisnički procesi (eng. *processes*)



Slika 1: Osnovna organizacija Linux operacijskog sustava (Ward, 2015)

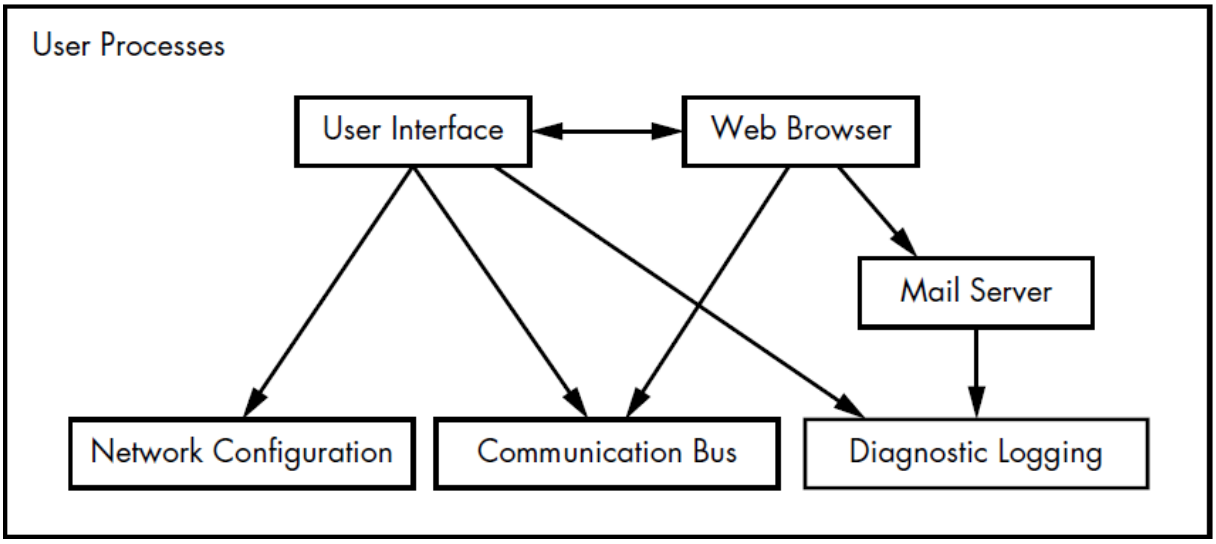
Hardware – Od sve opreme na računalnom sustavu, glavna memorija je možda najvažnija. U najsirovijem obliku, glavna memorija samo je veliko skladište za hrpu 0 i 1. Svaki 0 ili 1 se naziva bit. Ovdje se nalaze pokrenuta jezgra i procesi - oni su samo velike zbirke bitova. Sav ulaz i izlaz s perifernih uređaja teče kroz glavnu memoriju, također kao hrpa bitova. CPU (eng. *Central Processing Unit*) je samo operator memorije; čita svoje upute i podatke iz memorije i zapisuje podatke u memoriju. Često ćete čuti pojam stanje u odnosu na memoriju, procese, jezgru i druge dijelove računalnog sustava. Strogo govoreći, stanje je poseban raspored bitova. Na primjer, ako imate četiri bita u memoriji, 0110, 0001 i 1011 predstavljaju tri različita stanja. Kada uzmete u obzir da se jedan proces može lako sastojati od milijuna bitova memorije, često je lakše koristiti apstraktne pojmove kada govorimo o stanjima. Umjesto da opisujete stanje pomoću bitova, opisujete što je nešto učinilo ili radi u ovom trenutku. Na primjer, mogli biste reći "proces čeka na unos" ili "proces izvodi drugu fazu svog pokretanja".

Kernel - Jedan od zadataka jezgre je podijeliti memoriju na mnoge dijelove i mora u svakom trenutku održavati određene podatke o stanju tih dijelova. Svaki proces dobiva vlastiti dio memorije, a jezgra mora osigurati da svaki proces zadrži svoj udio. Jezgra je zaduženo za upravljanje zadacima u četiri opća područja sustava:

- Procesi - Jezgra je odgovorna za određivanje procesa koji smiju koristiti CPU.

- Memorija - Jezgra mora pratiti svu memoriju - što je trenutno dodijeljeno određenom procesu, što se može dijeliti između procesa i što je slobodno.
- Upravljački programi - Jezgra djeluje kao posrednik između hardvera (poput diska) i procesa. Obično je posao jezgre upravljati hardverom.
- Sistemski pozivi i podrška - Procesi obično koriste sistemske pozive za komunikaciju s jezgrom.

Process - Budući da je proces jednostavno stanje (ili slika) u memoriji, korisnički prostor također se odnosi na memoriju za cijelu zbirku pokrenutih procesa. (Možda ćete čuti i neformalniji izraz korisnička zemlja koji se koristi za korisnički prostor.) Većina stvarnih radnji na Linux sustavu događa se u korisničkom prostoru. Iako su svi procesi u biti jednaki sa stajališta jezgre, oni izvode različite zadatke za korisnike. Postoji osnovna struktura razine usluge (ili sloja) prema vrstama komponenti sustava koje predstavljaju korisnički procesi. Osnovne usluge su na donjoj razini (najbliže jezgri), uslužne usluge su u sredini, a aplikacije koje korisnici koriste nalaze se na vrhu. (Ward, 2015)



Slika 2: Tipovi procesa i njihova interakcija (Izvor: (Ward, 2015)

3. Ljuska

Što je ljuska (eng. *shell*)? Da bi smo mogli shvatiti što je, trebali bi vidjeti kako je neki autori definiraju. Neki od njih smatraju da je ljuska:

„Program ljuske UNIX tumači korisničke naredbe koje korisnik izravno unosi ili se mogu pročitati iz datoteke koja se zove shell skripta ili shell program. Shell skripte se tumače, a ne kompajliraju. Ljuska čita naredbe iz skripte redak po redak i traži te naredbe u sustavu, dok kompajler pretvara program u strojno čitljiv oblik, izvršnu datoteku - koja se zatim može koristiti u skripti ljuske.

Osim prosljeđivanja naredbi jezgri, glavni zadatak ljuske je pružanje korisničkog okruženja, koje se može pojedinačno konfigurirati pomoću konfiguracijskih datoteka resursa ljuske.“ (Garrels, 2008, str. 6)

„Na UNIX sustavima, iz kojih je izveden Linux, program korišten za tumačenje i upravljanje naredbama nazivan je ljuskom.“ (Negus, 2015, str. 65)

„GNU/Linux ljuska je poseban interaktivni uslužni program. Omogućuje korisnicima da pokreću programe, upravljaju datotekama u datotečnom sustavu i upravljaju procesima koji se izvode na Linux sustavu.“(Blum, bez dat., str. 13)

Svi se slažu s jednim, ljuska je program koji izvršava naredbe unesene od strane korisnika. Možemo reći da je ljuska interpretator naredbenog jezika, a korisnicima najčešće služi kao okruženje za razbijanje ponavljajućih zadataka u manje programske cjeline, provjeru procesa i resursa, pokretanje programa i procesa, itd.

3.1. Linux ljuske

Ljuska nije unificirani program, postoje mnoge vrste s različitim namjenama. Zadana ljuska na svim Linux distribucijama je bash ljuska, „Bash ljusku je razvio GNU projekt kao zamjenu za standardnu Unix ljusku, nazvanu Bourneova ljuska (po njenom tvorcu). Naziv bash ljuske je igra s ovim izrazom, koji se naziva "Bourne again shell".“.(Blum & Bresnahan, 2015, str. 10)

Razlozi za promjenom ljuske su najčešće:

- korištenje skripti napisanih za određene ljuske/okoline
- preferencija/navika
- ljuske se međusobno razlikuju po mogućnostima te načinom kako rade, možda će nam trebati mogućnosti određene ljuske te zato moramo raditi s njom

Tablica 1: Različite vrste ljuski (Blum & Bresnahan, 2015, str. 10)

Ljuska	Opis
ash	Jednostavna, lagana ljuska koja radi u okruženjima s malo memorije, ali ima potpunu kompatibilnost sa bash shellom
korn	Programerska ljuska kompatibilna s Bourneovom ljuskom, ali podržava napredne značajke programiranja kao što su asocijativni nizovi i aritmetika s pomičnim zarezom
tcsh	Ljuska koja uključuje elemente iz programskog jezika C u skripte ljuske
zsh	Napredna ljuska koja uključuje značajke iz bash, tcsh i korn ljuske, pružajući napredne značajke programiranja, zajedničke datoteke povijesti i tematske upute

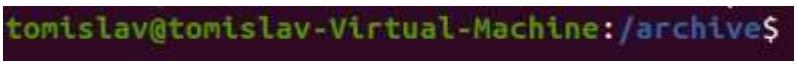
3.1.1. Bash ljuska

„Bash ili Bourne Again ljuska: standardna GNU ljuska, intuitivna i fleksibilna. Vjerojatno preporučljivo za početnike, a istovremeno je moćan alat za napredne i profesionalne korisnike. Na Linuxu, bash je standardna ljuska za obične korisnike. Ova ljuska je takozvani nadskup Bourneove ljuske, skup dodataka i priključaka. To znači da je ljuska Bourne Again kompatibilna s Bourne ljuskom: naredbe koje rade u sh, također rade u bash-u. Međutim, nije uvijek obrnuto.“(Garrels, 2008, str. 6)

Bash će biti ljuska koju ćemo koristiti u ovom radu, svaki primjer i zadatak će biti napisan u njoj te tako i projekt vezan uz rad.

3.2. Naredbeni redak

Naredbeni redak je polje za unos u emulatoru terminala (eng. *Command Line Interface* - CLI) koje nam omogućuje unos/izdavanje naredbi. Naredbeni redak pruža neke korisne informacije korisniku. Prvo u nazivu imamo ime korisnika, zatim znak „@“ koji služi za odvajanje korisničkog imena od imena stroja. Iza njega je ime stroja na kojem radimo te najčešće znak „~“ koji služi da nas obavijesti da radimo u standardnom direktoriju prijavljenog korisnika. Ako se premjestimo u neki drugi direktorij koristeći naredbu „cd“, npr. cd /archive, „~“ će zamijeniti naziv/putanja direktorija u kojem se trenutno nalazimo. Na kraju nam ostaje znak „$“, služi raspoznavanje tipa korisnika koji trenutno koristi CLI. Standardno je da „$“ znači obični korisnik dok „#“ znači da CLI koristimo kao root korisnik.

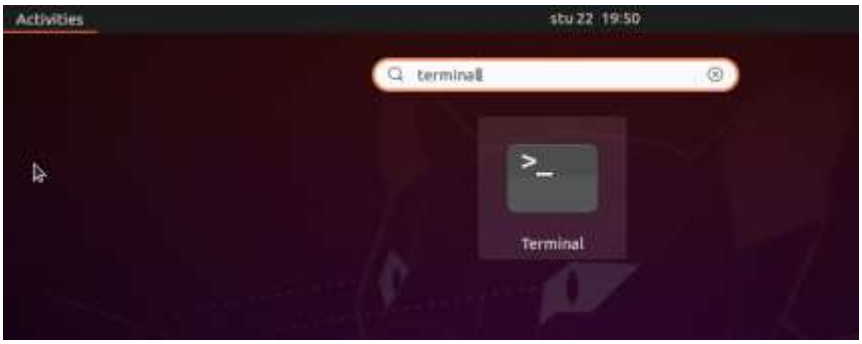


Slika 3: Naredbeni redak

3.3. Pristup

Ako koristimo Linux operacijski sustav, koji nema grafičko sučelje (eng. *Graphical User Interface* - GUI), čim uđemo u sustav koristiti ćemo naredbeni redak, no ako imamo operacijski sustavs sa GUI-em moramo pokrenuti emulator terminala. To možemo učiniti na više načina:

- u gornjem lijevom kutu stisnemo na „Activities“ što će nam otvoriti tražilicu u koju upišemo „terminal“ i ponuditi će nam CLI emulator
- u donjem desnom kutu odaberemo „Show Applications“ što će nam otvoriti listu instaliranih aplikacija među kojima je i CLI emulator
- pritiskom na kombinaciju tipki „CTRL+ALT+T“



Slika 4: Pristupanje terminalu 1. način

Slika 5: Pristupanje terminalu 2. način

3.4. Korištenje

Korištenje ljuske je zapravo vrlo jednostavno. Nakon što otvorimo prozor u terminalu, samo počnemo pisati naredbe za našu ljusku. U ovom dijelu rada ćemo proći najosnovnije naredbe za korištenje te najbolju praksu pri radu s ljuskom.

3.4.1. Osnovne naredbe

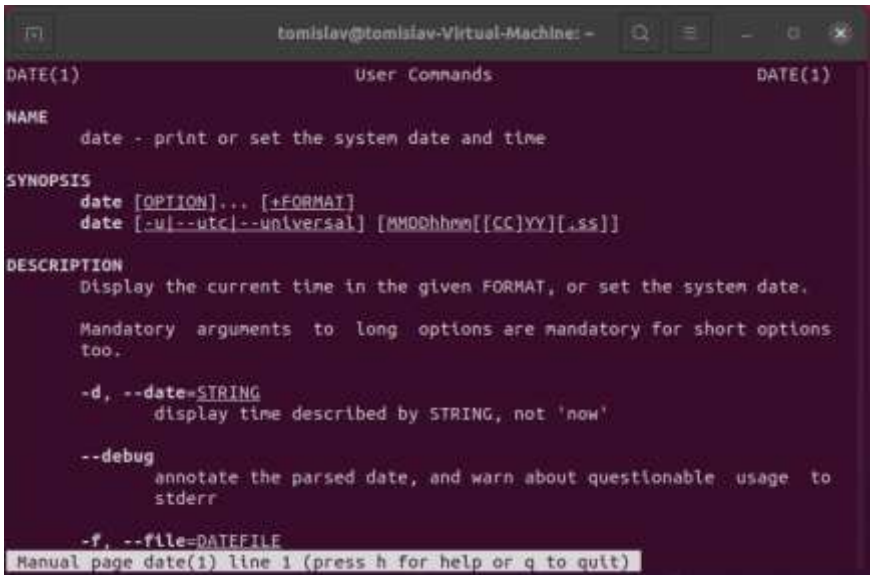
Osnovne naredbe bi bile naredbe koje su nam potrebne za neko osnovno korištenje ljuske i snalaženje u terminalu. Primarno navigacija kroz datotečni sustav te anredbe koje pomažu pri korištenju terminala.

3.4.1.1. Naredba man

Ova naredba služi kao priručnik za naredbe. Korisno je upoznati se s njom jer nam daje uvid u većinu naredba, što rade i kako se koriste te koje parametre možemo zadati naredbi. Naprimjer:

```
tomislav@tomislav-Virtual-Machine:~$ man date
```

Primjer 1: Naredba man

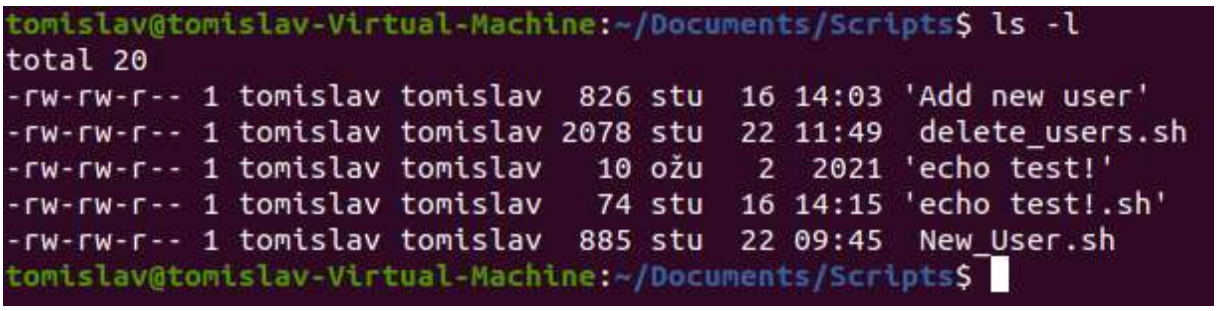


Slika 5: rezultat naredbe man date

Naredba će vratiti prozor kao što je prikazano na slici. Iz prikazanog možemo vidjeti naziv naredbe, što radi, njen opis, opcije za naredbu, itd. Možemo i listati što se sve nalazi u priručniku koristeći tipku za razmak (eng. *spacebar*) ili strelice na tipkovnici. Kada smo pronašli informaciju koja nam je potrebna iz priručnika izlazimo pritiskom na tipku 'Q'.

3.4.1.2. Parametri

Parametri služe kao dodatne opcije za naše naredbe. Mogu biti u skraćenoj verziji ili punoj, razlika je u tome što ako koristimo skraćenu verziju, možemo ih povezati u niz i koristi zajedno na nekoj naredbi. U prošloj naredbi smo vidjeli da možemo provjeriti koje parametre možemo staviti uz neku naredbu te uz njih piše što oni rade.



Slika 6: Naredba sa parametrom

3.4.1.3. Naredba echo

Služi za ispis teksta koji je proslijeđen kao argument. Ugrađena naredba koja se najčešće koristi u skriptama kada je potreban prikaz izlazne vrijednosti. Naprimjer:

```
tomislav@tomislav-Virtual-Machine:~$ echo FOI je najbolji
FOI je najbolji
```

Primjer 2: Korištenje naredbe echo

3.4.1.4. Naredba cd

Služi za navigaciju kroz datotečni sustav. Možemo je koristiti uz relativnu i apsolutnu putanju. Ako umjesto putanje stavimo .. nakon naredbe, vratiti ćemo se u roditeljski direktorij (ako nam je dostupan).

```
tomislav@tomislav-Virtual-Machine:~$ cd Documents
tomislav@tomislav-Virtual-Machine:~/Documents$
tomislav@tomislav-Virtual-Machine:~$ cd home/tomislav/Documents
tomislav@tomislav-Virtual-Machine:~/Documents$
tomislav@tomislav-Virtual-Machine:~/Documents$ cd ..
tomislav@tomislav-Virtual-Machine:~$
```

Primjer 3: Korištenje naredbe cd

3.4.1.5. Naredba ls

Služi za popis sadržaja direktorija u kojem se nalazimo ili koji direktorij navedemo uz naredbu. Prikazuje direktorije i datoteke.

```
tomislav@tomislav-Virtual-Machine:~/Documents$ ls
Desktop Documents Downloads Music Pictures Public snap Templates Videos
tomislav@tomislav-Virtual-Machine:~$ ls /home
tomislav
```

Primjer 4: Korištenje naredbe ls

3.4.1.6. Naredba pwd

Služi za prikaz putanje direktorija u kojem se trenutno nalazim u terminalu. Naprimjer:

```
tomislav@tomislav-Virtual-Machine:~$ pwd
/home/tomislav
```

Primjer 5: Korištenje naredbe pwd

3.4.2. Naredbe za datoteke i direktorije

Ovdje ćemo spomenuti par naredbi za kreiranje i brisanje datoteka i direktorija. Nećemo ih spomenuti sve, niti ćemo navesti sve njihove mogućnosti i kombinacije parametara. Proći ćemo osnove za uvod u korištenje tih naredbi te njihovu najučestaliju primjenu.

3.4.2.1. Naredba cat

Služi za čitanje datoteke te njen sadržaj prikazuje korisniku. Služi za kreiranje, gledanje te spajanje dokumenata. Naprimjer:

```
tomislav@tomislav-Virtual-Machine:~$ cat >new_test_file
Test
^C
tomislav@tomislav-Virtual-Machine:~$ cat new_test_file
Test
```

Primjer 6: Korištenje naredbe cat

Slične naredbe bi bile naredba more, less, tail i head koje se isto koriste za prikaz sadržaja datoteke.

3.4.2.2. Naredba touch

Služi za stvaranje prazne datoteke. Stvorena datoteka nema nikakav sadržaj. Naredba može primiti više argumenata, što će proizvesti više kreiranih datoteka. Osim kreiranja

datoteka, touch se može koristiti i za kreiranje te modificiranje vremenskih oznaka datoteka. Naprimjer:

```
tomislav@tomislav-Virtual-Machine:~$ touch FOI1 FOI2
tomislav@tomislav-Virtual-Machine:~$ ls
Desktop Documents Downloads Music Pictures Public snap Templates Videos
new_test_file FOI1 FOI2
```

Primjer 7: Naredba touch

3.4.2.3. Naredba cp

Služi za kopiranje, mogu se kopirati datoteke i direktoriji. Kreirana kopija je identična originalu sa različitim nazivom. Sama naredba zahtijeva minimalno 2 argumenta kako bi radila. Naprimjer:

```
tomislav@tomislav-Virtual-Machine:~$ cp new_test_file newer_test_file
tomislav@tomislav-Virtual-Machine:~$ ls
Desktop Documents Downloads Music Pictures Public snap Templates Videos
new_test_file FOI1 FOI2 newer_test_file
tomislav@tomislav-Virtual-Machine:~$ cp new_test_file Documents
tomislav@tomislav-Virtual-Machine:~$ ls Documents
new_test_file Scripts
```

Primjer 8: Naredba cp

3.4.2.4. Naredba mv

Služi za premještanje datoteka i direktorija, no uz to služi i za preimenovanje datoteka i direktorija. Naprimjer:

```
tomislav@tomislav-Virtual-Machine:~$ mv new_test_file old_test_file
tomislav@tomislav-Virtual-Machine:~$ ls
Desktop Documents Downloads Music Pictures Public snap Templates Videos
FOI1 FOI2 newer_test_file old_test_file
tomislav@tomislav-Virtual-Machine:~$ mv old_test_file Documents
tomislav@tomislav-Virtual-Machine:~$ ls Documents
new_test_file Scripts TEST old_test_file
```

Primjer 9: Naredba mv

3.4.2.5. Naredba rm

Služi za micanje direktorija i datoteka, preciznije miče reference na objekt iz datotečnog sustava. Naredba sama po sebi ne miče direktorije osim ako ne koristimo parametar za to. Naprimjer:

```
tomislav@tomislav-Virtual-Machine:~$ rm Documents/old_test_file
tomislav@tomislav-Virtual-Machine:~$ ls Documents
new_test_file Scripts TEST
tomislav@tomislav-Virtual-Machine:~$ ls Documents/TEST
File1 File2 File3
tomislav@tomislav-Virtual-Machine:~$ rm -rf Documents/TEST
tomislav@tomislav-Virtual-Machine:~$ ls Documents
new_test_file Scripts
```

Primjer 9: Naredba rm

3.4.2.6. Naredba mkdir

Služi za kreiranje direktorija. Koristeći naredbu mkdir možemo kreirati više direktorija odjednom i po volji postaviti dopuštenja na njih. Naprimjer:

```
tomislav@tomislav-Virtual-Machine:~$ mkdir FOI_ZABOK
tomislav@tomislav-Virtual-Machine:~$ ls
Desktop Documents Downloads Music Pictures Public snap Templates Videos
FOI1 FOI2 newer_test_file FOI_ZABOK
```

Primjer 10: Naredba mkdir

3.4.2.7. Naredba rmdir

Služi za brisanje praznih direktorija. Može primiti više direktorija kao argumente, no obrisati će ih samo ako su prazni. Naprimjer:

```
tomislav@tomislav-Virtual-Machine:~$ rmdir FOI_ZABOK
tomislav@tomislav-Virtual-Machine:~$ ls
Desktop Documents Downloads Music Pictures Public snap Templates Videos
FOI1 FOI2 newer_test_file
tomislav@tomislav-Virtual-Machine:~$ rmdir Documents
rmdir: faield to remove 'Documents/': Directory not empty
```

Primjer 11: Naredba rmdir

3.4.3. Naredbe za nadzor

Ovdje ćemo spomenuti naredbe koje su najučestalije u nadzoru okoline (najčešće ih koriste sistemski administratori). Služe nam za nadzor resursa i procesa te u kombinaciji s nekim drugim naredbama često služe kao skripte za izvještaje.

3.4.3.1. Naredba ps

Služi za prikaz informacija o procesima koji se trenutno izvršavaju u sustavu. Ovisno o korištenom parametru možemo dobiti različite prikaze informacija. Naprimjer:

```
tomislav@tomislav-Virtual-Machine:~$ ps
      PID   TTY          TIME        CMD
      13907 pts/0        00:00:00    bash
      20979 pts/0        00:00:00    ps
tomislav@tomislav-Virtual-Machine:~$ ps -e
      PID   TTY          TIME        CMD
      1     ?            00:00:04    systemd
      2     ?            00:00:00    kthreadd
      3     ?            00:00:00    rcu_gp
      4     ?            00:00:00    rcu_par_gp
      ...
```

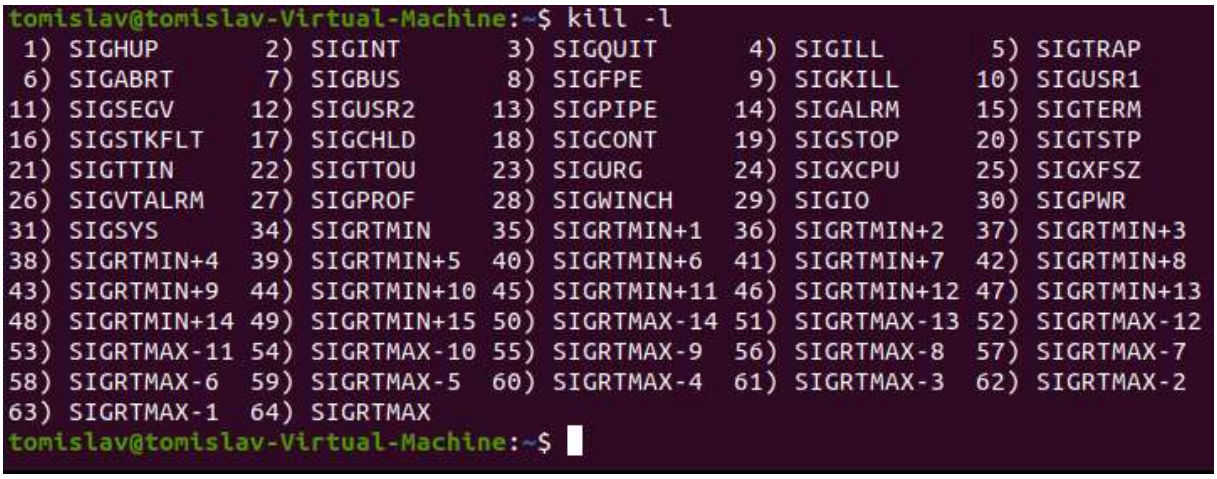
Primjer 12: Naredba ps

Dobiveni podaci su:

- PID – jedinstveni broj procesa (služi za identifikaciju procesa)
- TTY – tip terminala na koji je korisnik prijavljen
- TIME – količina Centralne Procesne Jedinice u minutama i sekundama koliko je dugo proces izvršavan
- CMD – naziv naredbe koja je pokrenula proces

3.4.3.2. Naredba kill

Služi za ručno gašenje procesa. Naredba kill šalje signal koji prekida proces i tako ga gasi. Ako se ne navede koji signal se šalje poslat će se zadani signal koji je SIGTERM. Naprimjer:



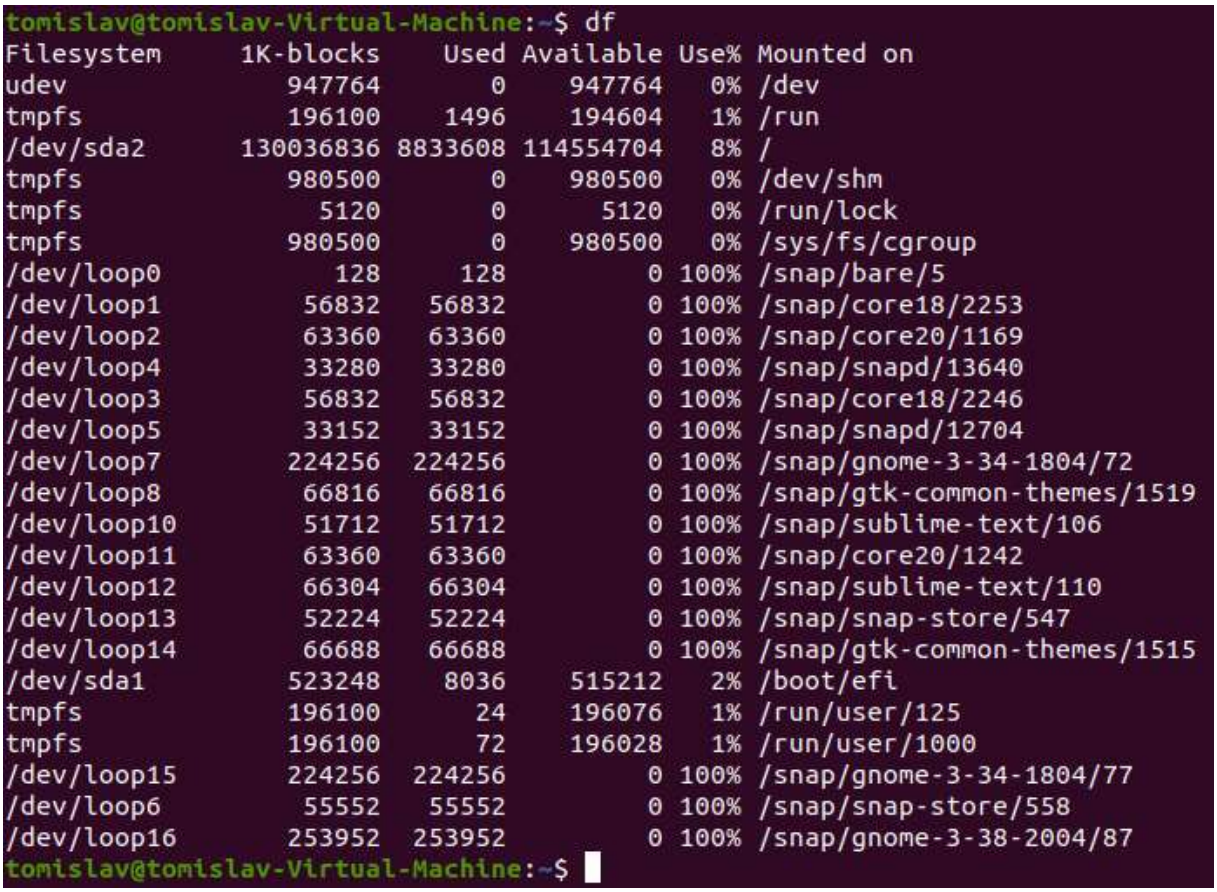
Slika 7: Naredba kill

```
tomislav@tomislav-Virtual-Machine:~$ kill 20979
bash: kill: (20979) – No such process
```

Primjer 13: Naredba kill

3.4.3.3. Naredba df

Služi za informaciju koliko prostora zauzima određeni datotečno sustav ili koliko je diska slobodno unutar pojedinog datotečnog sustava. Može se koristiti i za provjeru veličine pojedine datoteke ili direktorija. Naprimjer:



Slika 8: Naredba df

```
tomislav@tomislav-Virtual-Machine:~/Documents/Scripts$ df New_User.sh
Filesystem  1K-blocks   Used        Avaliable   Use%  Mounted on
/dev/sda2   130036836   8833616     114554696   8%    /
```

Primjer 14: Naredba df

3.4.4.Naredbe za podatke

Ovdje ćemo spomenuti par naredbi koje često služe za manipulaciju podataka. Vidjeli smo da nam neke naredbe mogu dati pregršt informacija te je potrebno korištenje parametara za lakše snalaženje. Sljedeće naredbe služe da bi dohvaćene informacije bile u preglednijem/jednostavnijem obliku ili da zauzimaju manje prostora na disku.

3.4.4.1. Naredba sort

Služi za sortirani prikaz datoteke. Ovisno o parametru ou naredbu, možemo priakz sortirati uzlazno, silazno, abecedno, itd. Naprimjer:

```
tomislav@tomislav-Virtual-Machine:~/Documents& cat >test.txt
```

```
6
5
3
2
1
4
^C
tomislav@tomislav-Virtual-Machine:~/Documents$ sort test.txt
1
2
3
4
5
6
```

Primjer 15: Naredba sort

3.4.4.2. Naredba grep

Služi kao filter. Pretražuje zadani dokument za specifičnim uzorkom znakova i prikazuje sve linije koje sadrže taj uzorak. Uzorak se još naziva regularnim izrazom (eng. regular expression skraćeno regex). Naprimjer:

```
tomislav@tomislav-Virtual-Machine:~/Documents$ grep 6 test.txt
6
```

Primjer 16: naredba grep

3.4.4.3. Naredba zip

Služi za kompresiju i pakiranje datoteka i direktorija kako bi im se smanjila veličina. Iznimno korisna naredba ako imamo limitirano koliko podataka smijemo premjestiti preko mreže. Datoteke kreirane kroz ovu naredbu imaju ekstenziju .zip. Naprimjer:

```
tomislav@tomislav-Virtual-Machine:~/Documents$ zip zipfile.zip
test.txt
tomislav@tomislav-Virtual-Machine:~/Documents$ ls
new_test_file Scripts test.txt zipfile.zip
tomislav@tomislav-Virtual-Machine:~/Documents$ zmore zipfile.zip
6
5
3
2
```

```
1
4
```

Primjer 17: Naredba zip

Ovdje smo uz naredbu zip naveli i naredbu zmore koja služi za prikaz sadržaja .zip datoteke. Ne ćemo je dodatno objašnjavati, pošto nam je potrebna samo za ovaj primjer. Slična naredba bi bila naredba tar koja služi za kreiranje arhiva ili izvlačenje podataka iz postojećih.

3.4.5.Varijable i varijable okoline

Varijable su niz znakova kojima dodjeljujemo neku vrijednost. Dodijeljena vrijednost može biti tekst, broj, ime datoteke, uređaj, itd. Kao i mnogi programski jezici, bash ljuska podržava kreiranje, dodjeljivanje i brisanje varijabli. Uz varijable koje možemo kreirati, postoje i one koje su od prije definirane. Njih zovemo varijable okoline (eng. *environment variables*), a dijele se na globalne i lokalne. Podjela ovisi o opsegu varijable, tj. regija iz koje joj se može pristupiti ili preko koje je definirana. Varijabla okruženja u Linuxu može imati globalni ili lokalni opseg.

Varijable su ekstremno bitne upravo iz tog razloga što u njih možemo pohraniti bilo što. Njihova primjena u skriptama je neupitna, a varijablu postavljamo i pozivamo ovako:

```
tomislav@tomislav-Virtual-Machine:~$ test=FOI
tomislav@tomislav-Virtual-Machine:~$ echo $test
FOI
```

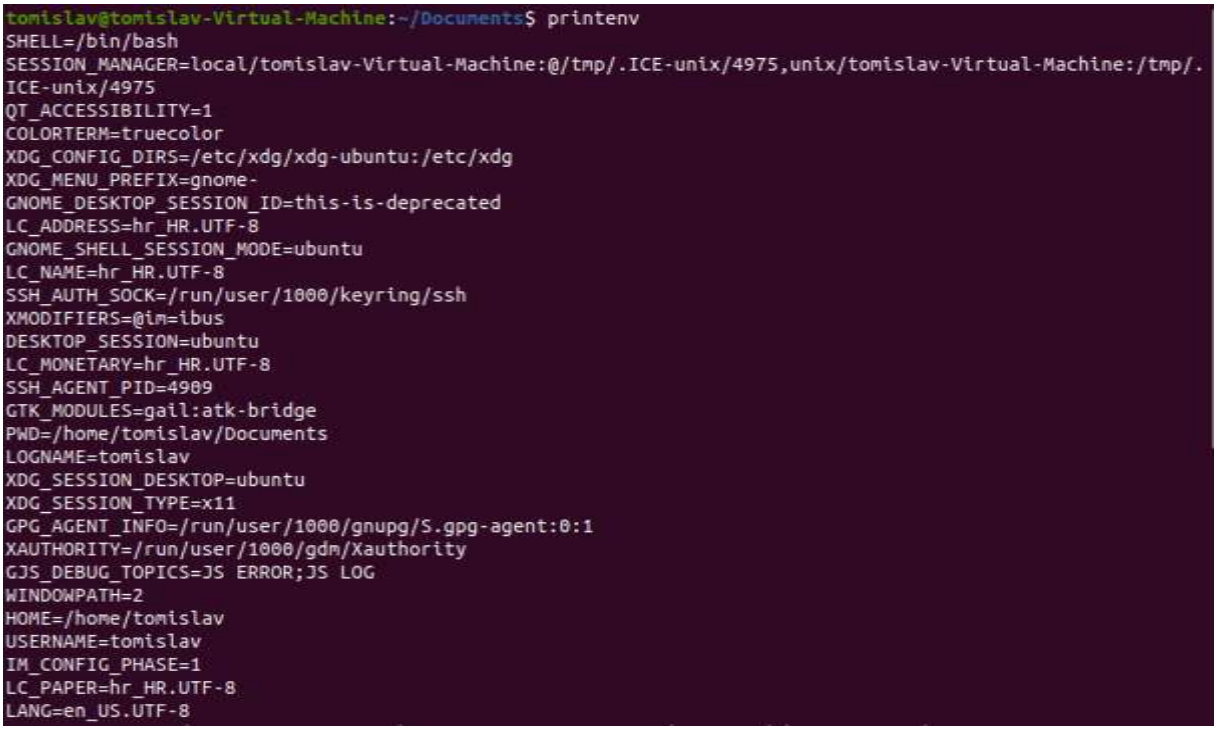
Primjer 18: Postavljanje i pozivanje varijable

3.4.5.1. Globalne

„Globalne varijable okoline vidljive su iz sesije ljuske i bilo kojeg podređenog procesa kojeg ljuska stvara. Lokalne varijable dostupne su samo u ljusci koja ih stvara. To čini globalne varijable okruženja korisnim u aplikacijama koje pokreću podređene procese koji zahtijevaju informacije od nadređenog procesa. Linux sustav postavlja nekoliko globalnih varijabli okruženja kada pokrenete svoju bash sesiju. Varijable okoline sustava uvijek koriste sva velika slova kako bi se razlikovale od normalnih varijabli korisničkog okruženja.“ (Blum, bez dat., str. 125)

Varijabla globalnog opsega koja je definirana u terminalu može se pristupiti s bilo kojeg mjesta u tom posebnom okruženju koje postoji u terminalu. To znači da se može koristiti u

svim vrstama skripti, programa ili procesa koji se izvode u okruženju vezanom za taj terminal. Naredba koju koristimo kako bi dohvatili listu globalnih varijabli okoline je naredba printenv.

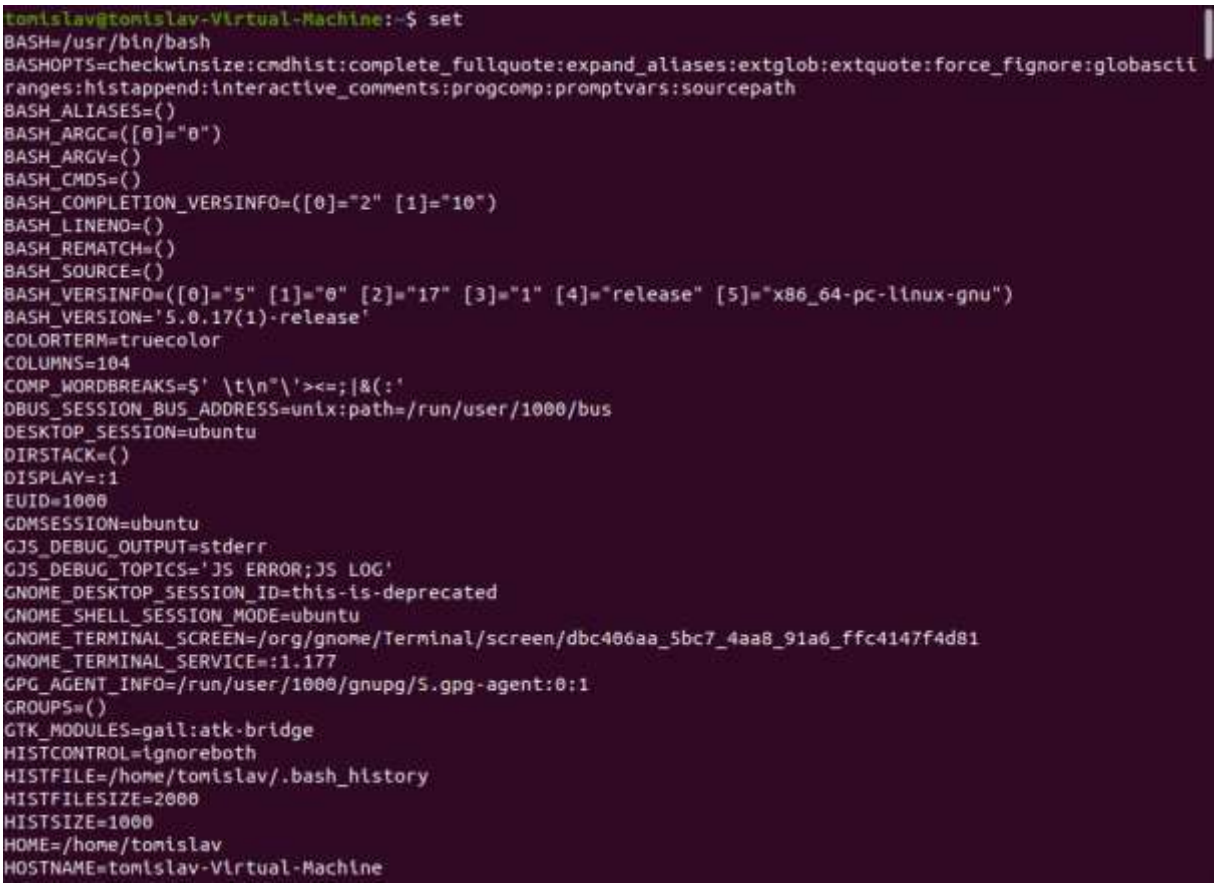


Slika 9: Naredba printenv

3.4.5.2. Lokalne

„Lokalne varijable okruženja, kao što njihov naziv implicira, mogu se vidjeti samo u lokalnom procesu u kojem su definirane. Nemojte se zbuniti oko lokalnih varijabli okruženja, one su jednako važne kao i globalne varijable okruženja. Zapravo, Linux sustav također definira standardne varijable lokalnog okruženja za vas prema zadanim postavkama. Pokušaj vidjeti popis varijabli lokalnog okruženja malo je zeznut. Nažalost, ne postoji naredba koja prikazuje samo varijable lokalnog okruženja. Naredba set prikazuje sve varijable okruženja postavljene za određeni proces. Međutim, to također uključuje varijable globalnog okruženja.“(Blum, bez dat., str. 125)

Varijabli lokalnog opsega koja je definirana u terminalu ne može pristupiti niti jedan program ili proces koji se izvodi u terminalu. Može mu pristupiti samo sam terminal (u kojem je varijabla definirana).



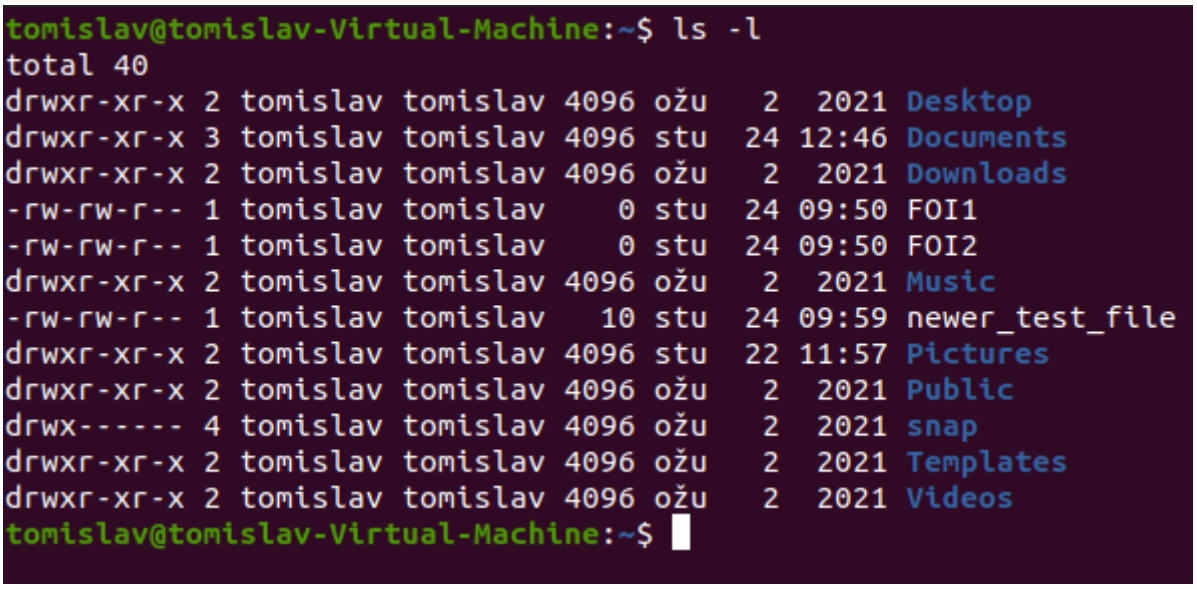
Slika 10: Naredba set

3.4.6.Prava

Prije nego što krenemo na prava pristupa datoteka i direktorija, moramo spomenuti sigurnost Linux operacijskog sustava.

„Svaki pojedinac koji pristupa Linux sustavu trebao bi imati dodijeljen jedinstveni korisnički račun. Prava korisnika za objekte u sustavu ovise o korisničkom računu s kojim se prijavljuju. Korisnička prava se prate pomoću korisničkog ID-a (često se naziva UID), koji se dodjeljuje računu kada je stvoren. UID je brojčana vrijednost, jedinstvena za svakog korisnika. Međutim, ne prijavljujete se na Linux sustav koristeći svoj UID. Umjesto toga, koristite ime za prijavu. Ime za prijavu je alfanumerički tekstualni niz od osam znakova ili manje koji korisnik koristi za prijavu na sustav (zajedno s pridruženom lozinkom). Linux sustav koristi posebne datoteke i uslužne programe za praćenje i upravljanje korisničkim računima na sustavu.“(Blum & Bresnahan, 2015, str. 161)

Kada izvršite naredbu “ls”, ne dobivate nikakve informacije o sigurnosti datoteka, jer prema zadanim postavkama “ls” navodi samo nazive. Više informacija možete dobiti korištenjem "opcije" s naredbom "ls". Naprimjer:



Slika 11: Prava nad datotekama i direktorijima

```
-rw-rw-r-- 1 tomislav tomislav 885 stu  22 09:45 New_User.sh
|[-][-][-]-   [------] [---]
| |  |  | |      |       |
| |  |  | |      |       +-----------> 7. Group
| |  |  | |      +-------------------> 6. Owner
| |  |  | +--------------------------> 5. Alternate Access Method
| |  |  +----------------------------> 4. Others Permissions
| |  +-------------------------------> 3. Group Permissions
| +----------------------------------> 2. Owner Permissions
+------------------------------------> 1. File Type
```

Primjer 19: Prava

U ovih par linija imamo pregršt informacija:

- prvi znak je uvijek '–' što znači datoteka ili 'd' što znači direktorij
- sljedećih 9 znakova pokazuje sigurnost i prava (-rwx)
- sljedeći stupac je vlasnik
- stupac iza toga grupa
- veličina u bajtovima
- datum zadnje promjene

- i na kraju ime datoteke ili direktorija

Ovih devet znakova zapravo dijelimo u skupine od tri znaka: korisnik, grupa i ostalo:

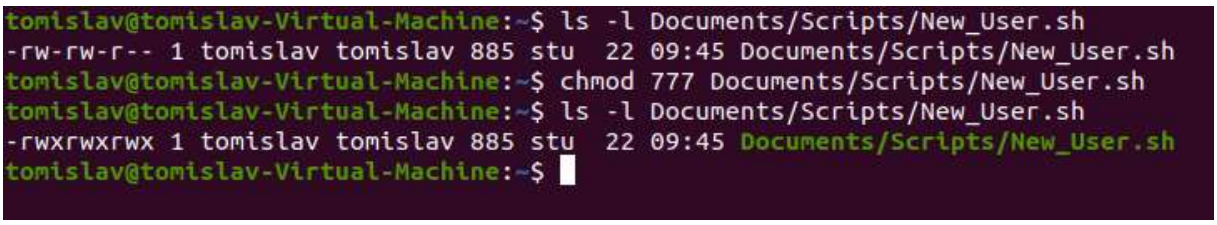
- korisnik – korisnička prava se odnose na vlasnika datoteke ili direktorija i ne utječu na druge korisnike
- grupa – prava grupe se odnose na grupu koja je dodijeljena dokumentu ili direktoriju, ne utječe na druge korisnike
- ostalo – ostala prava se odnose na sve ostale korisnike u sustavu

Znakove '-rwx' čitamo kao:

- r – čitanje (eng. *read*)
- w – pisanje (eng. *write*) ili modificiranje
- x – izvršavanje (eng. *execute*)
- '-' – ako je ovaj znak umjesto bilo kojeg od navedenih znakova, to znači da je određena dozvola oduzeta

3.4.6.1. Naredba chmod

Ponekad imamo potrebu dodati, maknuti ili promijeniti određena prava nad dokumentima i direktorijima. U tom slučaju koristimo naredbu 'chmod' koja znači promijeniti način rada (eng. *change mode*). Naprimjer:



Slika 12: Naredba chmod

U gornjoj slici možemo vidjeti promjenu prava nad skriptom nakon primjene naredbe 'chmod' uz argument '777' što znači da svakoj skupini dodjeljujemo 'rwx' pravo.

Tablica 2: Argumenti za naredbu chmod

0 (0+0+0)	Bez prava
1 (0+0+1)	Pravo pokretanja
2 (0+2+0)	Pravo pisanja/modificiranja

3 (0+2+1)	Pravo pokretanja + Pravo pisanja/modificiranja
4 (4+0+0)	Pravo čitanja
5 (4+0+1)	Pravo čitanja + Pravo pokretanja
6 (4+2+0)	Pravo čitanja + Pravo pisanja/modificiranja
7 (4+2+1)	Pravo čitanja + Pravo pisanja/modificiranja + Pravo pokretanja

Naredba slična 'chmod' bi bila naredba 'chown' koja služi za promjenu vlasništva nad direktorijima i dokumentima.

3.4.7.Naredbe za korisnike

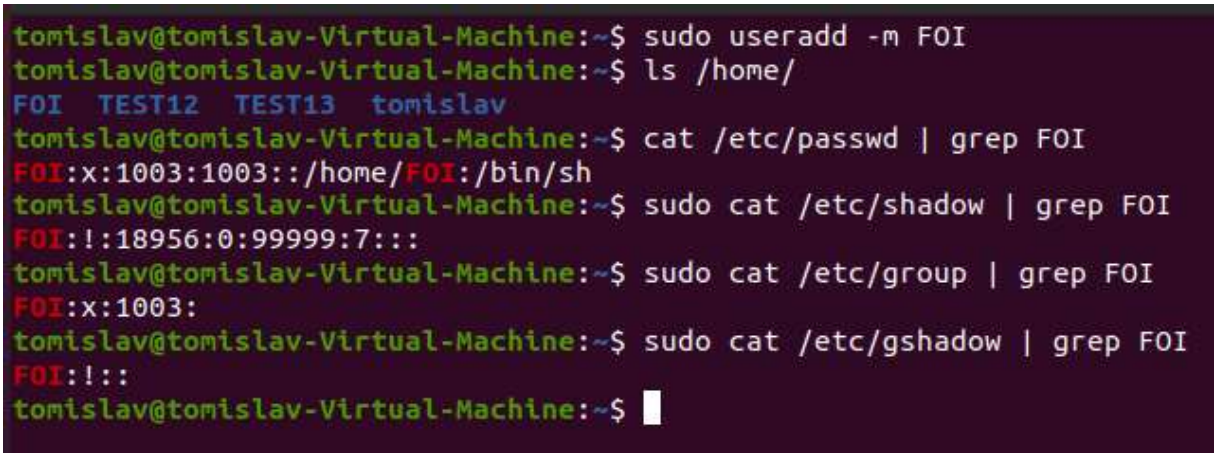
Ovdje ćemo u kratko proći naredbe za kreiranje, modificiranje i brisanje korisnika te promjene lozinke korisnicima (ove naredbe se mogu pokrenuti samo kao 'root' korisnik ili uz 'sudo' prefiks).

3.4.7.1. Naredba useradd

Služi za dodavanje novih korisnika u sustav te radi promjene nad sljedećim datotekama:

- /etc/passwd
- /etc/shadow
- /etc/group
- /etc/gshadow

Uz određene parametre može utjecati i na direktorije.



Slika 13: Naredba useradd

3.4.7.2. Naredba userdel

Služi za brisanje postojećih korisnika u sustavu te radi promjenu na datoteci '/etc/passwd'. Ako uz naredbu dodamo određene parametre može utjecati na više datoteka i direktorija.

```
tomislav@tomislav-Virtual-Machine:~$ sudo userdel -r FOI
userdel: FOI mail spool (/var/mail/FOI) not found
tomislav@tomislav-Virtual-Machine:~$ ls /home/
TEST12  TEST13  tomislav
tomislav@tomislav-Virtual-Machine:~$ cat /etc/passwd | grep FOI
tomislav@tomislav-Virtual-Machine:~$ sudo cat /etc/shadow | grep FOI
tomislav@tomislav-Virtual-Machine:~$ sudo cat /etc/group | grep FOI
tomislav@tomislav-Virtual-Machine:~$ sudo cat /etc/gshadow | grep FOI
tomislav@tomislav-Virtual-Machine:~$
```

Slika 14: Naredba userdel

Na slici možemo vidjeti grešku 'userdel: FOI mail spool (/var/mail/FOI) not found'. Ta greška nas ne mora zamarati jer ona znači da naredba nije pronašla direktorij elektroničke pošte, no unatoč tome korisnik je obrisan. To smo provjerili nizom naredbi da vidimo ako se obrisani korisnik i dalje pojavljuje u nekim zapisima.

3.4.7.3. Naredba usermod

Služi za promjenu svojstava korisnika putem naredbenog retka. Nakon kreiranja korisnika ponekad moramo promijeniti njegove atribute, poput lozinke. Da bismo to učinili koristimo naredbu 'usermod' koja radi promjene na sljedećim dokumentima:

- /etc/passwd
- /etc/group
- /etc/shadow
- /etc/login.defs
- /etc/gshadow
- /etc/login.defs

Slika 15: Naredba usermod

U priloženoj slici možemo vidjeti da se nakon pozivanja 'usermod -L' dodao '!' uz korisnika na kojem je bila pozvana naredba. To znači da smo traženi korisnički račun zaključali te se taj korisnik više ne može prijaviti u sustav.

3.4.7.4. Naredba passwd

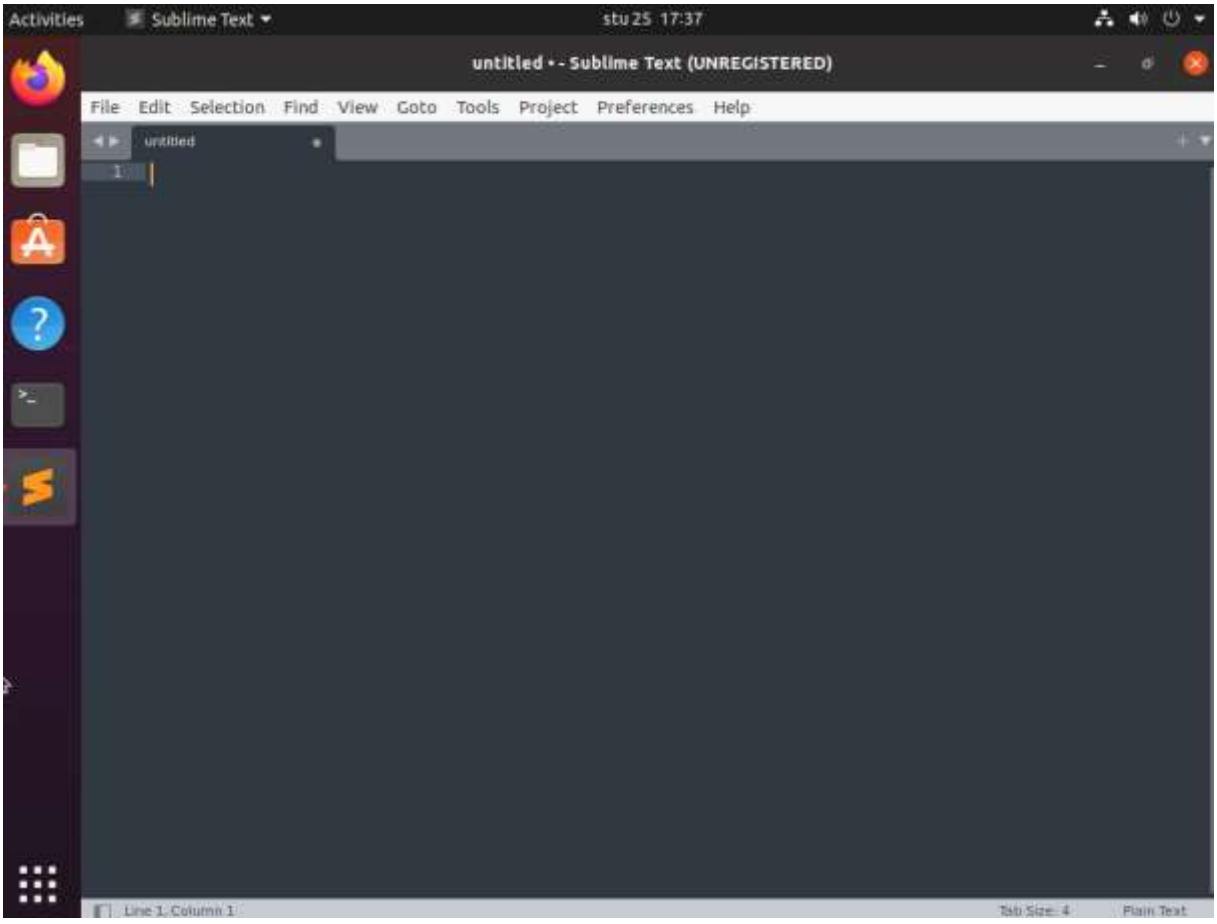
Služi za promjenu lozinki korisničkog računa. 'root' korisnik zadržava privilegiju promjene lozinke za bilo kojeg korisnika na sustavu, dok običan korisnik može promijeniti samo lozinku za svoj vlastiti račun.

```
tomislav@tomislav-Virtual-Machine:~$ passwd
Changing password for tomislav.
Current password:
New password:
Retype new password
passwd: password updated successfully
```

Primjer 20: Naredba passwd

4. Kreiranje skripte

Za kreiranje skripte trebamo program koji može uređivati tekst, no možemo ih i pisati u samom naredbenom retku. Za potrebe pisanja skripti koristiti ćemo Sublime program za uređivanje teksta. Program ne dolazi sa Linuxu OS-om te je potrebno skinuti i instalirati ga. Nakon instalacije program pokrećemo s liste dostupnih programa.



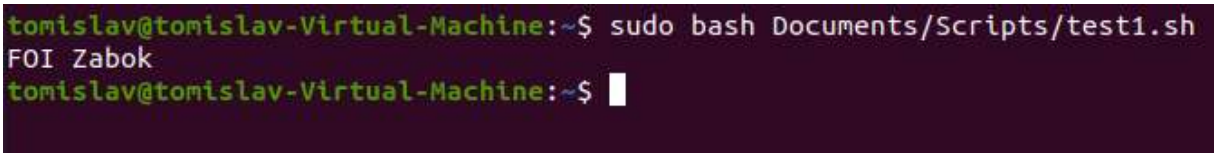
Slika 16: Sublime tekstualni editor

Prilikom pisanja skripte, u prvu liniju moramo navesti koju ljusku ćemo koristiti. Ta početna oznaka mora biti u prvoj liniji skripte inače bi se znak '#' ponašao kao da želi označiti komentar. Ovo označavanje se još poznaje kao 'shebang'. Potiče se pisanje komentara u skriptu kako bi imali više detalja čemu skripta služi te što pojedini dio radi.

```
#!/bin/bash
#Naša prva skripta!
#Ova skripta ispisuje zadanu vrijednost
VRIJEDNOST="FOI Zabok"
```

```
echo $Vrijednost
```

Primjer 21: Pisanje skripte



Slika 17: Izvršavanje skripte

Iz priloženog se vidi kako koristeći naredbeni redak pozivamo skriptu koja će odraditi svoje. U ovom slučaju smo koristili 'sudo' jer napravljenoj skripti nismo promijenili prava (pravo za pokretanje). U buduće, nakon kreiranja skripte, ćemo omogućiti da naš korisnik slobodno pokreće kreiranu skriptu.

4.1. Osnove skriptiranja

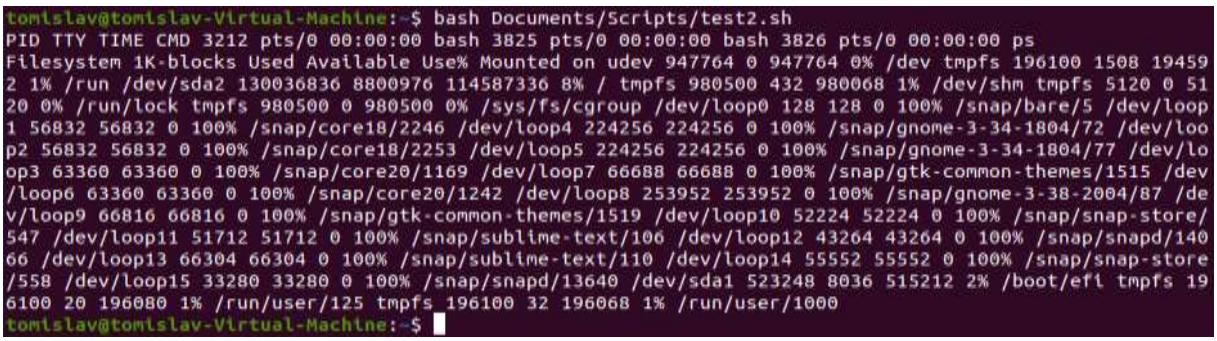
U ovom ćemo dijelu staviti fokus na način pisanja skripte: korištenje varijabli, prikaz podataka, matematičke radnje, povezivanje naredbi, itd…

4.1.1. Varijable

Već smo prije spomenuli varijable te kako im zadati vrijednost, no što ako u varijablu želimo spremiti rezultat neke naredbe? U tom slučaju moramo koristiti povratnu kvačicu, znak '`' (eng. *backtick*) ili '$()' format pri pisanju skripte. Naprimjer:

```
#!/bin/bash
VRIJEDNOST1=`ps`
VRIJEDNOST2=$(df)
echo $VRIJEDNOST1
echo $VRIJEDNOST2
```

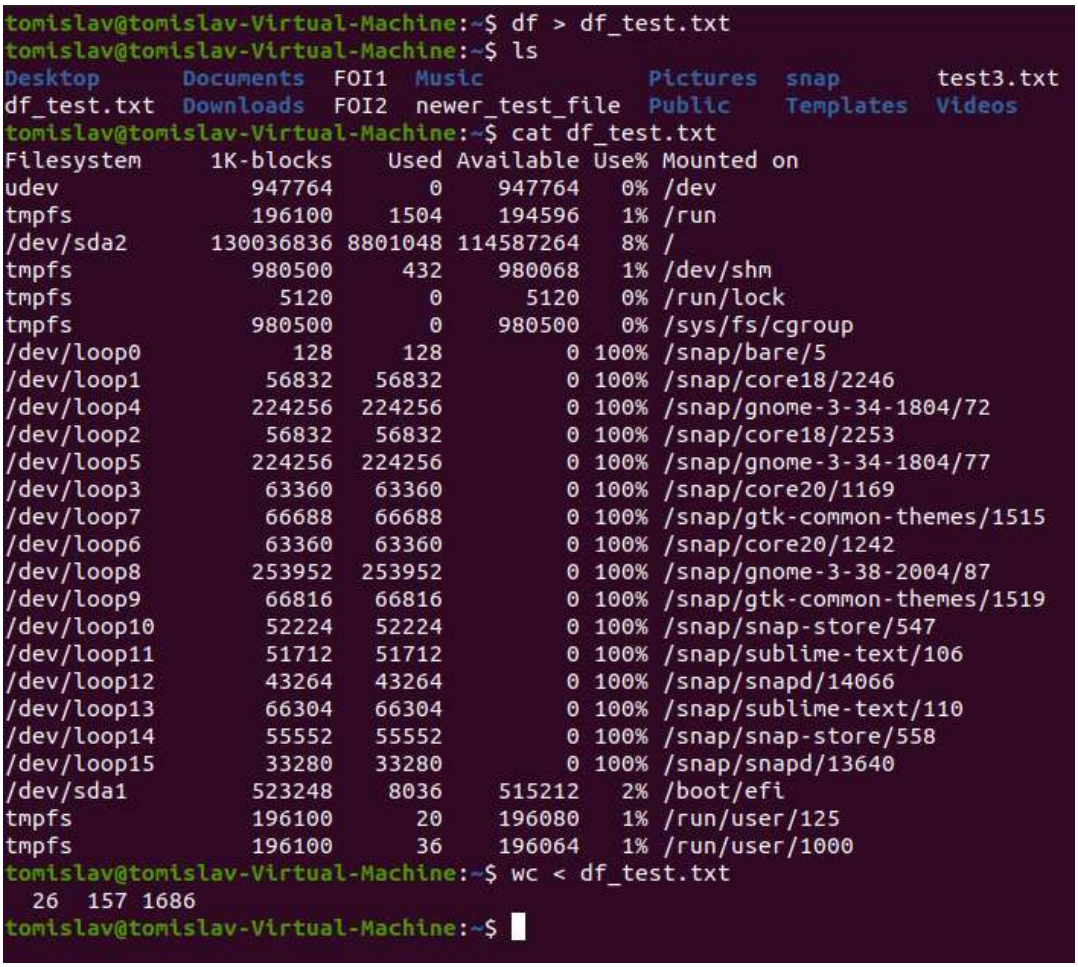
Primjer 22: Spremanje rezultata naredbe u varijablu



Slika 18: Izvršavanje skripte iz primjera 21

4.1.2. Preusmjeravanje ulaznih i izlaznih podataka

Ponekad umjesto prikaza podataka, želimo imati te podatke spremljene. Isto tako umjesto da ručno unosimo podatke u skriptu, možemo proslijediti datoteku s podacima. U slučaju izlaza koristiti ćemo operator '>' te '<' za ulaz. Naprimjer:



Slika 19: Preusmjeravanje izlaznih i ulaznih podataka

Postoje još '>>' i '<<' koji imaju sličnu svrhu. '>>' će dodati rezultat na postojeću datoteku (ne će je pregaziti s novom vrijednosti), a '<<' će tražiti da označimo koji dio priložene datoteke koristimo kao ulazne podatke (raspon). Naprimjer:

```
tomislav@tomislav-Virtual-Machine:~$ cat << df_test
>udev
>df_test
udev
```

Primjer 23: Operator <<

Osim preusmjeravanja rezultata ili izvora podataka, možemo preusmjeravati i greške. „Ako želite namjerno generirati poruke pogreške u svojoj skripti, možete preusmjeriti pojedinačni izlazni redak na STDERR. Vi samo trebate upotrijebiti simbol preusmjeravanja izlaza da preusmjerite izlaz na deskriptor STDERR datoteke. Kada preusmjeravate na deskriptor datoteke, ispred broja deskriptora datoteke morate staviti znak '&'“ (Blum & Bresnahan, 2015, str. 400)

4.1.3. Cijevi

Cijevi (eng. *pipes*) služe kako bi rezultat jedne naredbe proslijedili kao ulaznu vrijednost druge naredbe. Možemo za tu svrhu koristiti i preusmjeravanje, no ovo je bolji i jednostavniji način. Naprimjer:

```
tomislav@tomislav-Virtual-Machine:~$ df -h | sort > df_human_redable
tomislav@tomislav-Virtual-Machine:~$ ls
Desktop           df_test.txt  Downloads  FOI2   newer_test_file  Public  Templates  Videos
df_human_redable  Documents    FOI1       Music  Pictures         snap    test3.txt
tomislav@tomislav-Virtual-Machine:~$ cat df_human_redable
/dev/loop0      128K  128K     0 100% /snap/bare/5
/dev/loop10      51M   51M     0 100% /snap/snap-store/547
/dev/loop11      51M   51M     0 100% /snap/sublime-text/106
/dev/loop12      43M   43M     0 100% /snap/snapd/14066
/dev/loop13      65M   65M     0 100% /snap/sublime-text/110
/dev/loop14      55M   55M     0 100% /snap/snap-store/558
/dev/loop15      33M   33M     0 100% /snap/snapd/13640
/dev/loop1       56M   56M     0 100% /snap/core18/2246
/dev/loop2       56M   56M     0 100% /snap/core18/2253
/dev/loop3       62M   62M     0 100% /snap/core20/1169
/dev/loop4      219M  219M     0 100% /snap/gnome-3-34-1804/72
/dev/loop5      219M  219M     0 100% /snap/gnome-3-34-1804/77
/dev/loop6       62M   62M     0 100% /snap/core20/1242
/dev/loop7       66M   66M     0 100% /snap/gtk-common-themes/1515
/dev/loop8      248M  248M     0 100% /snap/gnome-3-38-2004/87
/dev/loop9       66M   66M     0 100% /snap/gtk-common-themes/1519
/dev/sda1       511M  7,9M  504M   2% /boot/efi
/dev/sda2       125G  8,4G  110G   8% /
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           192M  1,5M  191M   1% /run
tmpfs           192M   20K  192M   1% /run/user/125
tmpfs           192M   36K  192M   1% /run/user/1000
tmpfs           5,0M     0  5,0M   0% /run/lock
tmpfs           958M     0  958M   0% /sys/fs/cgroup
tmpfs           958M  432K  958M   1% /dev/shm
udev            926M     0  926M   0% /dev
tomislav@tomislav-Virtual-Machine:~$
```

Slika 20: Primjer cijevi

4.1.4. Matematičke operacije i operatori

Matematičke operacije se u naredbenom retku izvršavaju pomoću naredbe 'expr' ili korištenjem uglatih zagradi i dolar znaka '$[ ]'. Naprimjer:

```
tomislav@tomislav-Virtual-Machine:~$ expr 1 + 5
6
tomislav@tomislav-Virtual-Machine:~$ test=$[1+5]
tomislav@tomislav-Virtual-Machine:~$ echo $test
6
```

Primjer 24: Matematičke operacije

Bash matematiče operacije radi samo nad cijelim brojevima (eng. *integer*) te se za decimalne brojeve mora koristiti ugrađeni kalkulator, naredba 'bc'.

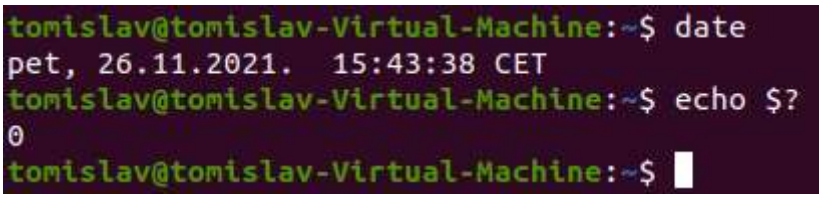
Slika 21: Ugrađeni kalkulator

Tablica 3: Aritmetički operatori

+	Zbrajanje
-	Oduzimanje
*	Množenje
/	Dijeljenje
%	Ostatak
**	Eksponencijalnost
++	Povećanje
--	Smanjenje

4.1.5.Izlaz

Svaka naredba ima svoj izlazni status koji se generira nakon pozivanja i izvršavanja naredbe. Izlazni status je broj te ovisno o uspješnosti izvršenja naredbe se taj broj mijenja (0 - 255). Označavamo ga posebnom varijablom '$?' i možemo ga koristiti u skriptama. Naprimjer:



Slika 22: Izlazni status

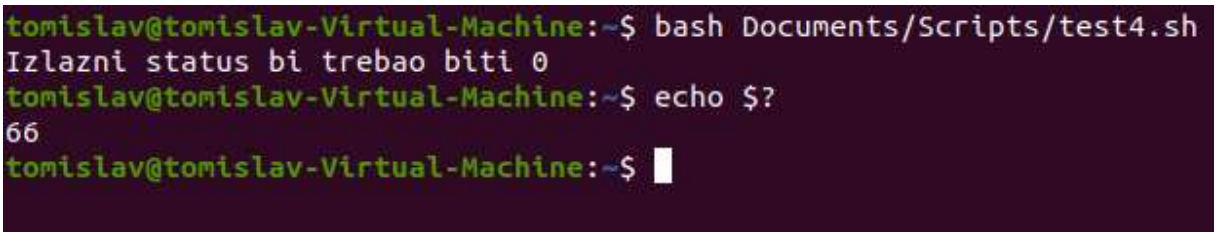
Tablica 4: Izlazni statusi

0	Uspješno izvršavanje naredbe
1	Opće nepoznata greška
2	Pogrešna upotreba naredbe
126	Naredba se ne može izvršiti
127	Naredba nije pronađena
128	Nevažeći izlazni argument
130	Naredba prekinuta (CTRL + C)
255	Izlazni status izvan dometa

Standardno, naša skripta nakon izvršavanja ima izlazni status zadnje izvršene naredbe. To možemo promijeniti sa naredbom 'exit'. Naprimjer:

```
#!/bin/bash
echo Izlazni status bi trebao biti 0
exit 66
```

Primjer 25: Naredba exit



Slika 23: Naredba exit

4.2. Napredno skriptiranje

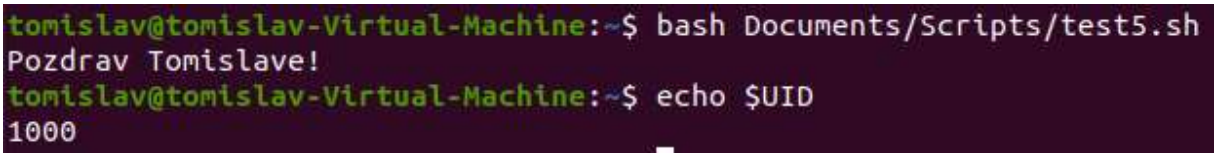
U ovome dijelu se fokusiramo na izjave i naredbe koje služe za logički slijed skripte te na uvjete i operatore vezane uz njih.

4.2.1.Grananja

Grananje nam služi kako bi mogli birati što će se izvršiti u našoj skripti ovisno o zadanim uvjetima i provjerama. Ako nešto odgovara uvjetima učini a), ako nešto odgovara drugim uvjetima učini b), inače nemoj ništa odraditi. Naprimjer:

```
#!/bin/bash
if [[ „${UID}“ == 1000 ]]
then
      echo Pozdrav Tomislave!
else
      echo Ti nisi Tomislav!
fi
```

Primjer 26: If-then-else



Slika 24: If-then-else skripta

Napomena! Uz izjavu 'if-then-else' možemo koristiti dvostruke obične i dvostruke uglate zagrade '(()) + [[]]'. Razlika je u tome što obične koristimo za matematičke izraze dok uglate koristimo za provjere nizova znakova (eng. *string*). Naprednija verzija ove izjave u sebi sadrži 'elif' izjavu, što nam omogućuje da u jednoj izjavi imamo više uvjeta. Slična izjava bi bila izjava 'case'.

```
#!/bin/bash
case $USER in
root)
      echo Hello ADMIN ;;
*)
      echo Hello USER ;;
esac
```

Primjer 27: Case

Za provjeru uvjeta koristimo operatore. Neke od operatora te njihovu funkciju ćemo navesti niže u tablicama.

Tablica 5: Logički operatori

==	Jednakost

!=	Nejednakost
<, <=	Manje, Manje ili Jednako
>, >=	Veće, Veće ili Jednako
&&	Logičko I
||	Logičko ILI
!	Negacija

Tablica 6: Operatori za datoteke i direktorije

-d	Provjerava ako je zadani direktorij postoji ili ne
-e	Provjerava ako zadana datoteka postoji ili ne
-r	Provjerava ako zadana datoteka ima pravo čitanja ili ne
-w	Provjerava ako zadana datoteka ima pravo pisanja ili ne
-x	Provjerava ako zadana datoteka ima pravo pokretanja ili ne
-s	Provjerava ako zadana datoteka ima veličinu > 0

4.2.2.Petlje

Petlje se koriste za izvođenje radnji iznova i iznova dok se ne ispuni uvjet ili dok se svi podaci ne obrade. Jedna od najčešće korištenih petlji je 'for' petlja, a u nju ćemo još spomenuti petlje 'while' i 'until'. Petlje:

- **For** petlja – prolazi kroz listu te za svaku stavku u listi odradi set naredbi definiran u tijelu petlje
- **While** petlja – provjerava ako je uvjet zadovoljen te ovisno o rezultatu uvjeta će se set naredbi izvršiti ili će naredba prestati (rezultat uvjeta mora biti 'true').
- **Until** petlja – izvršit će se onoliko puta dokle god rješenje uvjeta vraća 'false' (obrnuto od while petlje)

Naprimjer:

```
#!/bin/bash
for test in Fakultet Organizacije i Informatike
do
```

```
        echo $test
done
i=0
while [ $i -ne 3 ]
do
        echo 'While petlja.'
        i=$[ $i + 1 ]
done
i=5
until [ $i -eq 0 ]
do
        echo 'Until petlja.'
        ((i--))
done
```

Primjer 28: Petlje

```
tomislav@tomislav-Virtual-Machine:~$ sudo bash Documents/Scripts/test7.sh
Fakultet
Organizacije
i
Informatike
While petlja.
While petlja.
While petlja.
Until petlja.
Until petlja.
Until petlja.
Until petlja.
Until petlja.
tomislav@tomislav-Virtual-Machine:~$
```

Slika 25: Petlje

Uz petlje je potrebno spomenuti da se njihovo izvršavanje može prekinutu te nastaviti po potrebi. To možemo uz naredbe 'break' i 'continue':

- **Break** – naredba služi za prekid bilo koje petlje i koristimo je ako unutar liste tražimo specifičnu stvar ili ako uvjet može biti zadovoljen na više načina. Uz to služi i za prekide ugniježđenih petlji
- **Continue** – naredba služi za zaustavimo obradu naredbi unutar petlje, no da ne zaustavimo samu petlju, služi za postavljanje uvjeta unutar petlje kada se naredbe ne moraju izvršavati

Naprimjer:

```
#!/bin/bash
list=(1 2 3 4 5)
for i in ${list[@]}
do
      if [[ $i -eq 5 ]]
      then
            break
      fi
      echo $i
done
echo
for i in ${list[@]}
do
      if [[ $i -eq 2 ]]
      then
            continue
      fi
      echo $i
done
```

Primjer 29: Naredbe break i continue

Slika 26: Naredbe break i continue

4.2.3. Argumenti

Osnovna metoda slanja podataka naredbi je preko parametara naredbenog retka. Kako bi koristili te parametre u našim skriptama moramo koristiti pozicijske parametre. To su specijalne varijable koje su redoslijedno zadane. Naprimjer:

```
#!/bin/bash
args=(„$@“)
```

```
argsnum=${#args[@]}
for (( i =0 ; i <= argsnum; i++ )); do
      echo ${!i}
done
```

Primjer 30: Čitanje parametara

```
tomislav@tomislav-Virtual-Machine:~$ sudo bash Documents/Scripts/test9.sh test FOI Zabok
Documents/Scripts/test9.sh
test
FOI
Zabok
tomislav@tomislav-Virtual-Machine:~$
```

Slika 27: Pročitani parametri

Iz skripte se može zaključiti da parametri idu kao elementi niza (prvi element je onaj sa brojem (0), zatim se podižemo za jedno mjesto po sljedeći parametar (1), itd., dokle god vrijedi uvjet u petlji (uvjet je da broj izvođenja tijela petlje ne smije biti veći od broja parametara koje smo proslijedili). Trebalo bi naglasiti da parametar '$0' nije samo ime skripte, u njemu se može naći i putanja do pozvane skripte te se uz '$0' može koristiti naredba 'basename'. Naprimjer:

```
#!/bin/bash
args=(„$@“)
argsnum=${#args[@]}
for (( i =0 ; i <= argsnum; i++ )); do
      if [[ $i = 0 ]]
      then
            echo $(basename ${!i})
      else
            echo ${!i}
      fi
done
```

Primjer 31: Naredba basename

```
tomislav@tomislav-Virtual-Machine:~$ sudo bash Documents/Scripts/test9.sh test FOI Zabok
test9.sh
test
FOI
Zabok
tomislav@tomislav-Virtual-Machine:~$
```

Slika 28: Parametar $0 sa naredbom basename

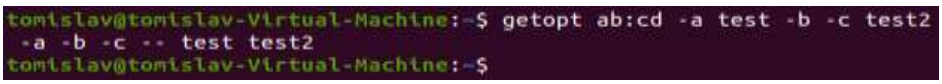
Ponekad uz predane parametre, naredbama predajemo i opcije. Opcije su parametri s jednim slovom te znakom '-' (-a, -b, -c) koji utječu na način kako će se naredba izvršiti. U naredbenom retku nalaze se odmah iza naredbe, kao i obični parametri koji se prosljeđuju. Zato u kombinaciji s opcijama koristimo naredbu 'shift', služi za pomak parametara jednu poziciju lijevo (prvi argument se gubi nakon korištenja). Pošto se opcije koriste prije argumenata 'shift' je savršena naredba za naše skripte. Naprimjer:

```
#!/bin/bash
while [[ -n „$1“ ]]
do
      case „$1“ in
            -a) echo „Option -a“;;
            -b) echo „Option -b“;;
            -c) echo „Option -c“;;
            -*) echo „$1 nije opcija“; shift; break;;
            *) for ARGUMENT in „${@}“
               do
                  echo $ARGUMENT
                  shift
               done;;
      esac
      shift
done
echo „Preostalo je ${#} parametara.“
```

Primjer 32: Prolazak kroz opcije i pronalaženje argumenata

Slika 29: Prolazak kroz parametre

Slične naredbe bi bile 'getopt' i 'getopts' koje su napravljene tako da prepoznaju koji je parametar opcija, a koji argument te nam olakšavaju kreiranje skripti.



Slika 30: Primjer getopt naredbe

4.2.4.Unos podataka

Osim prosljeđivanja podataka kroz argumente, korisnik može i sam, direktno unositi podatke u skriptu. U tome nam pomaže naredba 'read'. Naprimjer:

```
#!/bin/bash
echo „Unesite centar studiranja: “
read centar
echo „Vi studirate u centru $centar.“
```

Primjer 33: Naredba read

```
tomislav@tomislav-Virtual-Machine:~$ sudo bash Documents/Scripts/test11.sh
Unsite centar studiranja:
Zabok
Vi studirate u centru Zabok.
tomislav@tomislav-Virtual-Machine:~$
```

Slika 34: Rezultat naredbe read

Ova naredba nudi mnoge opcije pri korištenju; možemo postaviti da korisnik ima zadano vrijeme za unos podataka, da se podaci ne prikazuju pri unosu, itd… Uz unos podataka možemo je koristiti i za čitanje datoteka liniju po liniju. Naprimjer:

```
#!/bin/bash
cat Documents/test | while read line
do
      echo $line
done
echo „End of file.“
```

Primjer 34: Korištenje naredbe read za čitanje datoteke

```
tomislav@tomislav-Virtual-Machine:~$ sudo bash Documents/Scripts/test12.sh
Dobri prijatelji
Šamar
Politika
Jednim osmijehom
End of file.
tomislav@tomislav-Virtual-Machine:~$
```

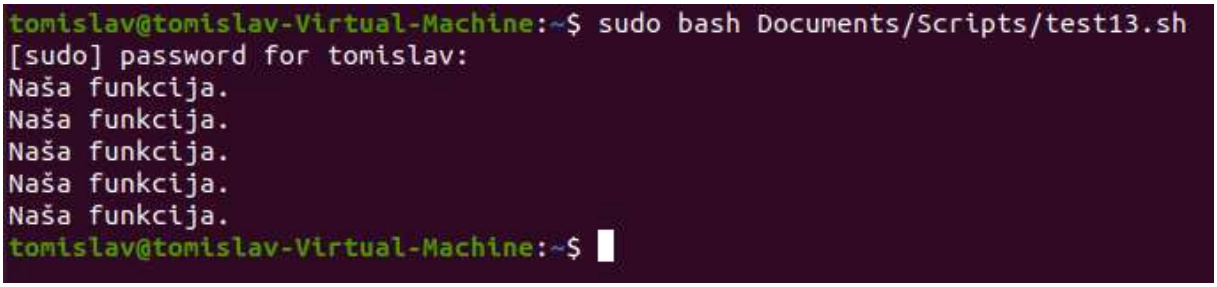
Slika 32: Čitanje datoteke

4.2.5.Funkcije

Funkcije u bashu se ne razlikuju od funkcija u programskim jezicima. Ako se u našim skriptama desi da često koristimo identičan kod na više mjesta, najbolje bi bilo taj dio koda pretvoriti u funkciju. Funkcija je dio koda s dodijeljenim imenom koji, kada se pozove ta funkcija, se izvrši. Naprimjer:

```
#!/bin/bash
function test {
      echo „Naša funkcija.“
}
for (( i=0; i < 5 ; i++))
do
      test
done
```

Primjer 35: Kreiranje funkcije



Slika 33: Izvršavanje skripte s funkcijom

Treba naglasiti da funkcije, kao i naredbe, imaju izlazni status. Status će ovisiti o uspješnosti izvršavanju zadnje naredbe u funkciji. Srećom postoji naredba 'return' koja postavlja određenu vrijednost statusa (vrijednost mora biti cijeli broj od 0 - 255).

4.2.6.Cron table

Cron je program zadužen za raspored izvršavanja zadataka na sistemu. Svaki korisnik ima svoju ceon tablicu, no prvo je korisnik mora kreirati. Ona se najčešće koristi u serverskim okruženjima gdje je potrebno da se neki zadatak vrti 24/7, ako je računalo ugašeno tijekom rasporeda nekog zadatka, taj zadatak se ne će izvršiti.

Iz gore navedenog razloga postoji i anacrontab, koji služi da provjerava ako se neki zadatak vrti u pravilnim vremenskim intervalima. Ako se neki interval preskoči zadatak će se izvršiti u prvom mogućem trenutku.

Nakon što kreiramo skriptu možemo je dodati u raspored crona ako je potrebno da se ta skripta redovito izvršava. Naprimjer:

```
*/10 * * * * /usr/bin/somedirectory/somecommand arg1 arg2
```

```
|    | | | |          |                          |     |
|    | | | |          |                          |     - argument
|    | | | |          |                          ------- argument
|    | | | |          ------ putanja do skripte i skripta/naredba
|    | | | ---------------------------------- dan u tjednu (0-7)
|    | | ---------------------------------- mjesec u godini (0-12)
|    | ----------------------------------------dan u mjesecu (0-31)
|    ----------------------------------------sati u danu (0-23)
----------------------------------------minute u satu (0-59)
```

Primjer 36: Crontab

5. Projekt

Za primjer automatizacije u naredbenom retku sam napisao dvije skripte čiji ću rad demonstrirati uz pomoć slika te samog koda. Prva skripta, kada se pozove, dodaje novog lokalnog korisnika. Druga skripta, kada se pozove, briše postojećeg lokalnog korisnika.

Obje skripte rade na principu argumenata te je potreban unos od strane trenutnog korisnika kako bi se skripta uspješno izvršila.

5.1. Skripta za dodavanje novog lokalnog korisnika

Prvo moramo naglasiti koju ćemo ljusku koristiti.

```
#!/bin/bash
```

Skripta je namijenjena administratorima te je potrebna određena privilegija za pokretanje.

```
if [[ "${UID}" -ne 0 ]]
then
   echo 'Please run with sudo or as root.' >&2
   exit 1
fi
```

Ako korisnik ne doda argument dobit će ispisane upute za korištenje.

```
if [[ "${#}" -lt 1 ]]
then
  echo "Usage: ${0} USER_NAME [COMMENT]..." >&2
  echo 'Create an account on the local system with the name of USER_NAME
and a comments field of COMMENT.' >&2
  exit 1
fi
```

Prvi argument je korisničko ime.

```
USER_NAME="${1}"
```

Ostali argumenti su komentari.

```
shift
COMMENT="${@}"
```

Generiranje lozinke.

```
PASSWORD=$(date +%s%N | sha256sum | head -c48)
```

Kreiranje korisnika.

```
useradd -c "${COMMENT}" -m ${USER_NAME} &> /dev/null
```

Provjera ako je korisnik kreiran.

```
if [[ "${?}" -ne 0 ]]
then
  echo 'The account could not be created.' >&2
  exit 1
fi
```

Postavi generiranu lozinku na novo kreiranog korisnika.

```
echo ${PASSWORD} | passwd --stdin ${USER_NAME} &> /dev/null
```

Provjera ako je lozinka uspješno postavljena.

```
if [[ "${?}" -ne 0 ]]
then
  echo 'The password for the account could not be set.' >&2
  exit 1
fi
```

Forsiraj promjenu lozinke na sljedećem ulasku u operacijski sustav.

```
passwd -e ${USER_NAME} &> /dev/null
```

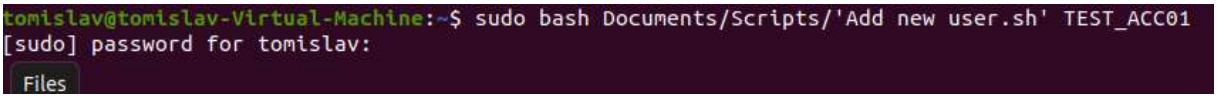
Prikaži korisnika, lozinku i računalo gdje je korisnik kreiran.

```
echo 'username:'
echo "${USER_NAME}"
echo
echo 'password:'
echo "${PASSWORD}"
echo
echo 'host:'
echo "${HOSTNAME}"
```

Izlazak sa 0 jer 0 = uspješna izvedba.

```
exit 0
```

Slika 34: Pozivanje skripte



Slika 35: Sudo password

Slika 36: Rezultat izvršavanja

5.2. Skripta za brisanje postojećeg lokalnog korisnika

Prvo moramo naglasiti koju ljusku ćemo koristiti.

```
#!/bin/bash
```

Kreiranje direktorija gdje će se spremati arhive.

```
ARCHIVE_DIR='/archive'
```

Funkcija koja vraća upute za korištenje.

```
usage() {
  echo "Usage: ${0} [-dea] USER [USERN]..." >&2
  echo 'Disable and/or delete and/or archive a local Linux account(s).' >&2
  echo '  -d  Disables account(s).' >&2
  echo '  -e  Deletes account(s) and removes associated home directory.'
>&2
  echo '  -a  Creates an archive of the home directory associated with the
account(s).' >&2
  exit 1
}
```

Skripta je namijenjena administratorima te je potrebna određena privilegija za pokretanje.

```
if [[ "${UID}" -ne 0 ]]
then
   echo 'Please run with sudo or as root.' >&2
   exit 1
fi
```

Petlja koja prolazi kroz parametre.

```
while getopts dea OPTION
do
  case ${OPTION} in
    d) DISABLE_USER='true' ;;
    e) DELETE_USER ='true' ;;
```

```
    a) ARCHIVE='true' ;;
    ?) usage ;;
  esac
done
```

Provjera ako je prvi argument parametar. Mora sadržavati: d), e) ili a) te se prelazi na argumente za korisnička imena. Ako ne sadrži poziva se usage funkcija.

```
if [[ "${1}" != *@(d|e|a)* ]]
then
  usage
else
  shift "$(( OPTIND - 1 ))"
fi
```

Provjera ako je korisnik zadao argumente (korisnike).

```
if [[ "${#}" -lt 1 ]]
then
  usage
fi
```

Petlja koja će se izvršiti n puta gdje n=broj zadanih argumenata.

```
for USERNAME in "${@}"
do
  echo "Processing user: ${USERNAME}"

  USERID=$(id -u ${USERNAME})
```

Provjera ako je UID < 1000, ako je, skripta će se prestati izvršavati.

```
  if [[ "${USERID}" -lt 1000 ]]
  then
    echo "Refusing to remove the ${USERNAME} account with UID ${USERID}."
>&2
    exit 1
  fi
```

Provjera ako je a) parametar bio zadan. Ako je, provjerit će se ako postoji /archive direktorij te ako ne postoji kreirat će se.

```
  if [[ "${ARCHIVE}" = 'true' ]]
  then
    if [[ ! -d "${ARCHIVE_DIR}" ]]
    then
      echo "Creating ${ARCHIVE_DIR} directory."
      mkdir -p ${ARCHIVE_DIR}
```

```
      if [[ "${?}" -ne 0 ]]
      then
        echo "The archive directory ${ARCHIVE_DIR} could not be created."
>&2
        exit 1
      fi
    fi
```

Postavljanje varijabli za putanju korisničkog direktorija te naziva datoteke arhive. Zatim će se pokušat kreirati datoteka sa .tgz ekstenzijom te nazivom kako je deklarirano u prijašnjoj varijabli za naziv.

```
    HOME_DIR="/home/${USERNAME}"
    ARCHIVE_FILE="${ARCHIVE_DIR}/${USERNAME}.tgz"
    if [[ -d "${HOME_DIR}" ]]
    then
      echo "Archiving ${HOME_DIR} to ${ARCHIVE_FILE}"
      tar -zcf ${ARCHIVE_FILE} ${HOME_DIR} &> /dev/null
      if [[ "${?}" -ne 0 ]]
      then
        echo "Could not create ${ARCHIVE_FILE}." >&2
        exit 1
      fi
    else
      echo "${HOME_DIR} does not exist or is not a directory." >&2
      exit 1
    fi
  fi
```

Provjera ako je e) parametar bio zadan. Ako je, pozvat će se naredba za brisanje korisnika te njegovog direktorija.

```
  if [[ "${DELETE_USER}" = 'true' ]]
  then
    userdel ${REMOVE_OPTION} ${USERNAME}
    if [[ "${?}" -ne 0 ]]
    then
      echo "The account ${USERNAME} was NOT deleted." >&2
      exit 1
    fi
    echo "The account ${USERNAME} was deleted."
  fi
```

Provjera ako je d) parametar bio zadan. Ako je, pozvat će se naredba za gašenjem korisničkog računa.

```
  if [[ "${DISABLE_USER}" = 'true' ]]
  then
    chage -E 0 ${USERNAME}
    if [[ "${?}" -ne 0 ]]
    then
      echo "The account ${USERNAME} was NOT disabled." >&2
      exit 1
    fi
    echo "The account ${USERNAME} was disabled."
  fi
done
```

Izlazak sa 0 jer 0 = uspješna izvedba.

```
exit 0
```

```
tomislav@tomislav-Virtual-Machine:~$ sudo bash Documents/Scripts/delete_users.sh -dea TEST10
```

Slika 37: Pozivanje skripte sa svim parametrima

```
tomislav@tomislav-Virtual-Machine:~$ sudo bash Documents/Scripts/delete_users.sh -dea TEST10
Processing user: TEST10
Archiving /home/TEST10 to /archive/TEST10.tgz
userdel: TEST10 mail spool (/var/mail/TEST10) not found
The account TEST10 was deleted.
chage: user 'TEST10' does not exist in /etc/passwd
The account TEST10 was not disabled.
tomislav@tomislav-Virtual-Machine:~$
```

Slika 38: Rezultat skripte

```
tomislav@tomislav-Virtual-Machine:~$ sudo bash Documents/Scripts/delete_users.sh -d TEST13
Processing user: TEST13
The account TEST13 was disabled.
tomislav@tomislav-Virtual-Machine:~$ sudo bash Documents/Scripts/delete_users.sh -e TEST14
Processing user: TEST14
userdel: TEST14 mail spool (/var/mail/TEST14) not found
The account TEST14 was deleted.
tomislav@tomislav-Virtual-Machine:~$ sudo bash Documents/Scripts/delete_users.sh -a TEST12
Processing user: TEST12
Archiving /home/TEST12 to /archive/TEST12.tgz
tomislav@tomislav-Virtual-Machine:~$
```

Slika 39: Pozivanje skripte s pojedinačnim parametrom i rezultati

```
tomislav@tomislav-Virtual-Machine:~$ ls /archive/
TEST10.tgz  TEST12.tgz
tomislav@tomislav-Virtual-Machine:~$
```

Slika 40: Arhivirani korisnici

```
tomislav@tomislav-Virtual-Machine:~$ sudo chage -l TEST13
Last password change                                    : password must be changed
Password expires                                        : password must be changed
Password inactive                                       : password must be changed
Account expires                                         : sij 01, 1970
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
tomislav@tomislav-Virtual-Machine:~$
```

Slika 41: Ugašeni korisnik

6. Zaključak

Vjerujem da je rad pokazao mogućnosti te potencijal naredbenog retka i automatizacije. Bash ljuska je jednostavan program za korištenje, a samo pisanje skripti se može uz bilo koji uređivač teksta. Naredbe su smislene i imaju opciju za svaku primjenu, a ako neka naredba nije smislena uvijek se može pozvati 'man' ili '--help' u pomoć. Osim jednostavnosti, bash ljuska ima i veliku zajednicu korisnika, ustaljena je i lako se pronađe svako rješenje vezano uz skriptiranje. Logika je lagana za pohvatati te niti jedan računalni korisnik ne bi trebao imati problema sa grananjem, petljama, itd. Da rezimiram ovaj rad; vjerujem da sam uz primjere i slike pokazao kako se i najjednostavnije korištenje računala može ubrzati ili pojednostaviti. Približio sam izradu skripti kroz svakodnevne zadatke koje bi sistemski administrator mogao raditi. Smatram da je potencijal za automatizacijom koristeći skripte iznimno velik, ne samo zbog mogućnosti ljuske nego i zbog potrebe na tržištu. Ako bash ljuska nekome ne odgovara, znanje stečeno njenim korištenju uvijek može biti odskočna daska prema nekim drugim skriptnim jezicima, primjerice Python-u.

7. Popis literature

Blum, R. (bez dat.). *Linux® Command Line and Shell Scripting Bible*. 818.

Blum, R., & Bresnahan, C. (2015). *Linux® Command Line and Shell Scripting Bible: Blum/Linux® Command Line and Shell Scripting Bible*. John Wiley & Sons, Inc. https://doi.org/10.1002/9781119209409

Cooper, M. (bez dat.). *Advanced Bash-Scripting Guide*. 916.

Garrels, M. (2008). *Bash Guide for Beginners*. 173.

Linux Shell Scripting Tutorial. (bez dat.). 46.

Negus, C. (2015). *Linux® Bible, Ninth Edition*. John Wiley & Sons, Inc.

Shotts, W. (bez dat.). *The Linux Command Line*. 555.

Ward, B. (2004). *How Linux works: What every superuser should know*. No Starch Press.

Ward, B. (2015). *How Linux works: What every superuser should know* (2nd edition). No Starch Press.

What is Automation? - ISA. (bez dat.). Preuzeto 08. kolovoz 2021., od https://www.isa.org/about-isa/what-is-automation

8. Popis slika

9. Popis tablica

10. Popis primjera