

Operacijski sustavi (OS)

Nositelj: doc. dr. sc. Ivan Lorencin

Asistent: Luka Blašković, mag. inf.

Ustanova: Sveučilište Jurja Dobrile u Puli, Fakultet informatike u Puli



Fakultet informatike u Puli

(1) Uvod u operacijske sustave

#1

OS

Svaki računalni sustav sastavljen je od procesora, radnog spremnika, vanjskih spremnika i različitih ulazno-izlaznih naprava. Te komponente čine sklopovlje računala. Međutim, samo sklopovlje nije nam od velike koristi ako uz njega ne postoji odgovarajuća programska oprema. Različiti primjenski programi, s pomoću kojih korisnici računala obavljaju sebi korisne zadatke, transformiraju računalni sustav u koristan virtualni stroj. Kao potpora svim tim raznovrsnim programima postoji skup osnovnih programa koji omogućuju prevođenje radnih zahvata na računalu. Taj se skup osnovnih programa naziva operacijskim sustavom. Na ovom kolegiju studenti će naučiti upravljati operacijskim sustavom računala kroz naredbeni redak koji za razliku od grafičkog sučelja (gdje se koriste miš i vizualni elementi), koristi isključivo tekstualne naredbe. Poznavanje naredbenog retka ključno je za svakog informatičara, jer je de-facto standard u administraciji sustava, automatizaciji zadataka i radu na udaljenim poslužiteljima. Moderna softverska rješenja gotovo uvijek se implementiraju u oblaku, gdje su naredbeni alati neophodni za upravljanje infrastrukturom.

 Posljednje ažurirano: 11.3.2025.

Sadržaj

- [Operacijski sustavi \(OS\)](#)
- [\(1\) Uvod u operacijske sustave](#)
 - [Sadržaj](#)
- [1. Uvod](#)
- [2. Naredbeni redak \(CLI\)](#)
 - [2.1 Bash - Bourne Again Shell](#)
 - [Ključne razlike između terminala, CLI-ja i shella](#)
 - [2.2 Kako pokrenuti bash?](#)
 - [Unix-like OS](#)

- [Windows OS](#)
- [3. Osnovne bash naredbe](#)
 - [3.1 Osnovne manipulacije datotekama/direktorijima](#)
 - [3.2 Tablica osnovnih bash naredbi](#)
- [Zadaci za Vježbu 1](#)

1. Uvod

Operacijski sustav (eng. *Operating System*, skraćeno *OS*) je temeljni softver svakog računalnog sustava. Njegova primarna funkcija je upravljanje hardverskim resursima te osiguravanje učinkovitog i pouzdanog sučelja između korisnika i hardverskih komponenti, omogućujući tako učinkovit i stabilan rad računala.

Dva važna zadatka operacijskog sustava su:

1. Olakšavanje uporabe računala korisnicima
2. Djelotvorno iskorištavanje svih dijelova računala

Naime, unutar računala može se istodobno odvijati više poslova.

Primjerice: istodobno dok se procesor "brine" o izvođenju jednog niza instrukcija, pristupni sklop pisača može iz glavnog spremnika prenositi sadržaj na pisač (printer). OS se mora pobrinuti da se procesor "prebacuje" s izvođenja jednog niza instrukcija na drugi i podržati **višeprogramski rad**.

Rani razvoj OS-a bio je usmjeren prema sustavima razvijenim za tzv. *mainframe* računala, s naglaskom na serijsko izvršavanje zadataka i minimizaciju potrebe za interakcijom s korisnikom. Samim tim, operacijskih sustava bilo je mnogo i bili su vrlo specifični za određeni računalni sustav, odnosno pojedino *mainframe* računalo. Tipično, svaki put kad bi proizvođač računala izbacio novi model, morao bi razviti i novi operacijski sustav za taj model, a samim tim i prilagoditi sve programe.

Uvođenjem **višezadačnosti** (eng. *multitasking*) i **višekorisničkog** (eng. *multiuser workload*) rada početkom 70-ih godina 20. stoljeća, operacijski sustavi su postali sposobni istovremeno izvršavati više zadataka, što je danas normalno u modernom računarstvu.

Definicija operacijskog sustava značajno je evoluirala. Naime, nekad su operacijski sustavi predstavljali isključivo osnovni skup programa nužnih za funkcioniranje računala, dok su danas postali kompleksni sustavi koji uključuju niz dodatnih programa i alata.

Možemo to usporediti s automobilima – rani modeli nisu imali brzinomjer, radio ili klimu, ali danas su ti dodaci standardni i nezamislivo je kupiti novi automobil bez njih. Slično tome, moderni operacijski sustavi više nisu samo temeljni alati za upravljanje računalom, već dolaze s nizom dodatnih programa, poput grafičkog sučelja, web preglednika, uređivača teksta, datotečnog sustava pa i razno-raznih *third party* aplikacija.

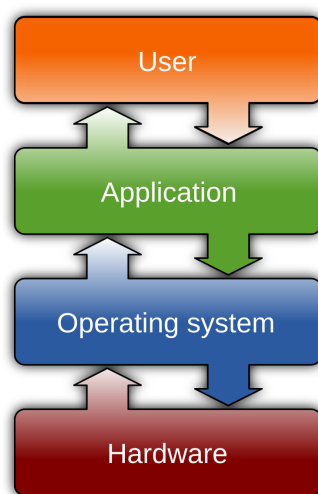
Zbog toga se operacijski sustav danas često doživljava kao **platforma** – cjeloviti korisnički sustav koji uključuje integrirano sučelje, datotečni sustav, uređivače teksta, tablica, slika i druge korisne alate. Više nije riječ samo o skupu osnovnih programa, već o ukupnom okruženju u kojem korisnici mogu obavljati razne zadatke na računalu.



Evolucija računala, od *mainframe* računala do modernih prijenosnih računala i *all-in-one* računala

Svaki računalni sustav moguće je prikazati kroz hijerarhijsku strukturu koja se sastoji od četiri razine (1 - najapstraktnija, 4 - najkonkretnija):

1. **Korisnička razina (User Layer)** - najviša razina na kojoj se nalaze programi koje koriste krajnji korisnici
2. **Razina primjenskih programa (Application Layer)** - razina na kojoj se nalaze programi koji koriste usluge operacijskog sustava
3. **Operacijski sustav (Operating system)** - posrednik između fizičkih komponenti računala i ostalog softvera
4. **Razina sklopovlja (Hardware Layer)** - najniža razina na kojoj se nalaze fizičke komponente računala



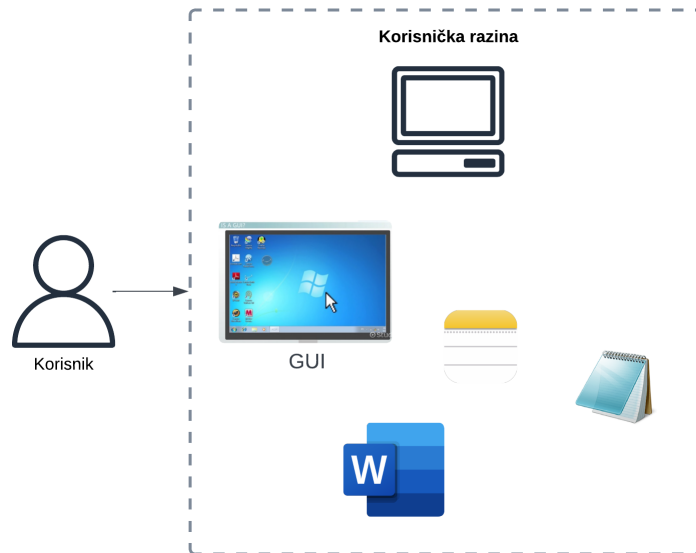
4 razine računalnog sustava i njihova međusobna povezanost

Pokazat ćemo primjer kako ovo izgleda u praksi na primjeru otvaranja tekstualne datoteke.

1. Korisnička razina

Na najvišoj razini, korisnik pokreće program za obradu teksta (npr. Microsoft Word). Ovdje se odvija **interakcija s grafičkim korisničkim sučeljem** (eng. *GUI*) na način da korisnik klikne na ikonu programa ili otvori datoteku iz nekog direktorija.

Aplikacija na ovoj razini ne zna kako se podaci pohranjuju na disku niti kako pristupiti disku, ali zna kako prikazati podatke korisniku jednom kad ih dobije.

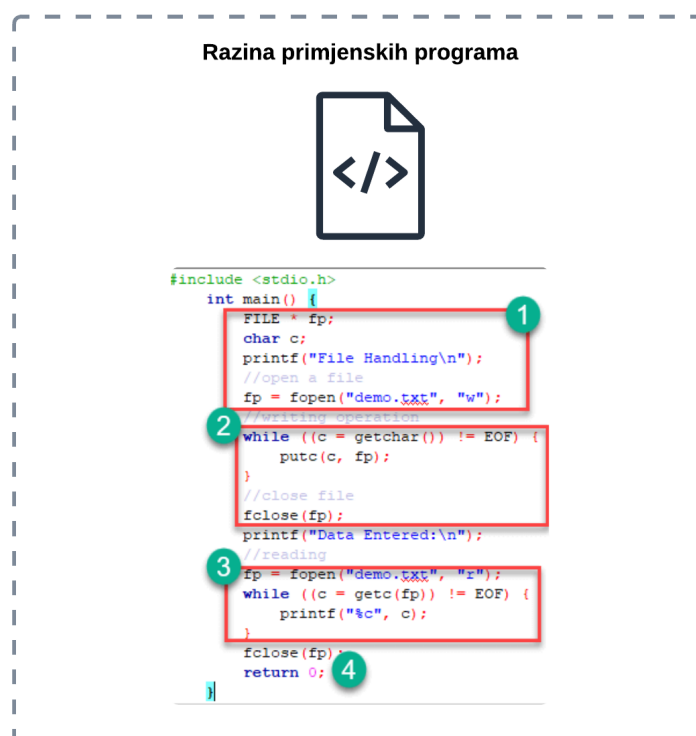



 Korisnička razina: korisnik pokreće program za obradu teksta kroz GUI

2. Razina primjenskih programa

Nakon što korisnik pokrene tekstualni uređivač na korisničkom sloju, program koristi niz **aplikacijskih programskih sučelja** (eng. *Application programming interface*, skraćeno *API*) za komunikaciju s operacijskim sustavom. Recimo: `C` jezik nudi funkciju `fopen()` za čitanje datoteke, u Pythonu bi to bila funkcija `open()`, u Javi je to `FileReader` sučelje itd.

U ovom kontekstu, zadaća primjenskih programa je da **koriste sučelje koje nudi operacijski sustav** kako bi pristupili datotekama, memoriji, mreži i ostalim resursima i osiguraju siguran pristup i učinkovit rad s tim resursima. Na neki način, oni **apstrahiraju složenost pristupa resursima i omogućuju programerima da se fokusiraju na rješavanje problema**.

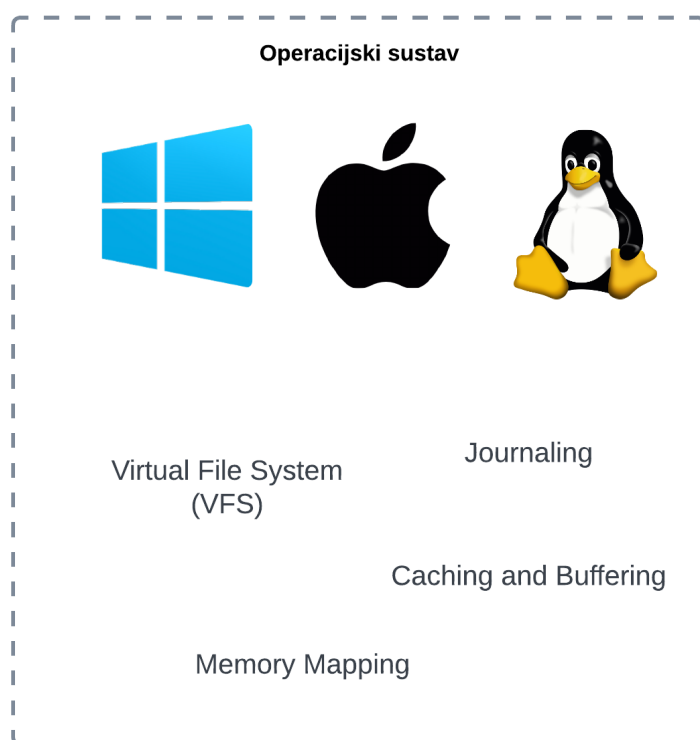


 Razina primjenskih programa: primjenski program `fopen` koristi API operacijskog sustava za pristup resursima

3. Operacijski sustav

Operacijski sustav prima zahtjev preko API-ja i obavlja niz operacija kako bi zadovoljio zahtjev primjenskog programa. Na primjer, ako je primjenski program zatražio otvaranje tekstualne datoteke, OS mora odraditi sljedeće operacije (redoslijed operacija u ovom slučaju nije nužno ovakav):

- **učitati datoteku:** OS mora pronaći odgovarajuću datoteku na disku
- **provjeriti dozvole pristupa:** OS mora provjeriti dozvole pristupa za odgovarajuću datoteku
- **upravljati radnom memorijom:** OS mora osigurati dovoljno radne memorije za učitavanje datoteke
- **rukovati potencijalnim greškama i iznimkama:** OS mora obraditi greške koje se mogu pojaviti tijekom operacije
- **vratiti potrebne informacije** natrag prema primjenskom programu
- **osloboditi resurse** nakon završetka operacije



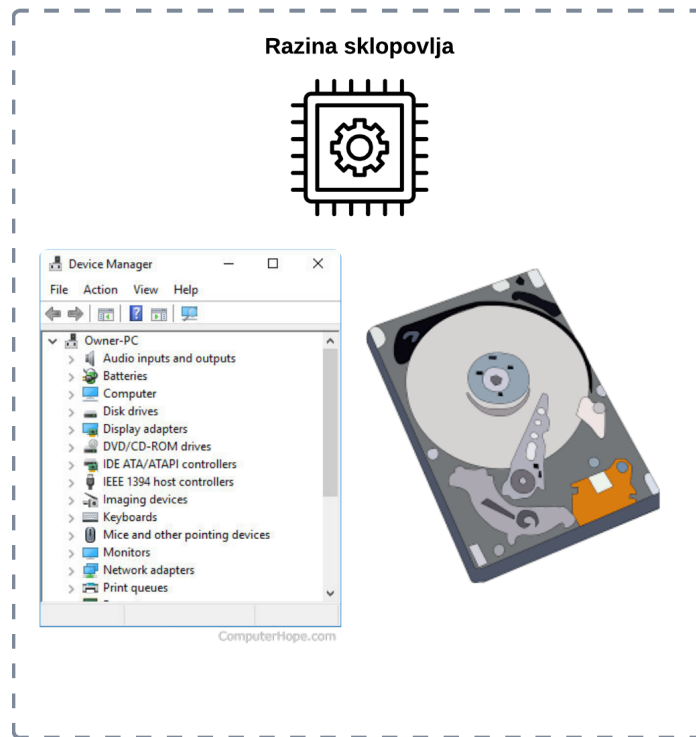
Operacijski sustav: Upravljanje resursima i povratna komunikacija s primjenskim programima


4. Razina sklopovlja

Na najnižoj razini, operacijski sustav koristi upravljačke programe (*eng. Device drivers*) kako bi komunicirao s fizičkim komponentama računala.

Upravljački programi su posebni programi koji omogućuju operacijskom sustavu da komunicira s hardverom, kao što su diskovi, memorija, grafičke kartice, mrežne kartice itd. Upravljačke programe u pravilu razvijaju proizvođači hardvera, a možemo ih zamisliti kao posrednike između operacijskog sustava i *firmware* softvera koji je programiran u samom hardveru.

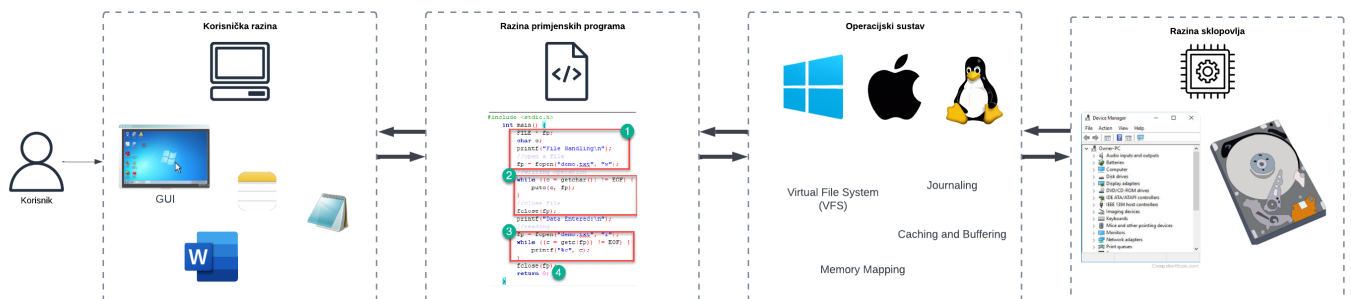
Bez odgovarajućeg upravljačkog programa, operacijski sustav ne može ispravno koristiti povezani hardver. Ovo je najkonkretnija razina na kojoj se odvija komunikacija s fizičkim komponentama računala.




 Razina sklopovlja: upravljački programi omogućuju komunikaciju s fizičkim komponentama računala

Naravno, kao što je i vidljivo na ilustraciji iznad, komunikacija između ovih razina je **dvosmjerna** (\leftrightarrow).

U nastavku je prikazana **ukupna komunikacija između različitih razina računalnog sustava** na opisanom primjeru otvaranja tekstualne datoteke:



 Ukupni prikaz dvosmjerne komunikacije između različitih razina računalnog sustava na opisanom primjeru otvaranja tekstualne datoteke

TL;DR

- **Korisnička razina:** korisnik pokreće program za obradu teksta
- **Razina primjenskih programa:** primjenski program koristi API operacijskog sustava za pristup resursima
- **Operacijski sustav:** OS obavlja operacije koje zadovoljavaju zahtjev primjenskog programa
- **Razina sklopovlja:** upravljački programi omogućuju komunikaciju s fizičkim komponentama računala u svrhu izvršavanja stvarne operacije

Pitanje za razmišljanje:

Nakon ovog kratkog uvoda u operacijske sustave, na sljedećih nekoliko vježbi ćemo se ustvari baviti učenjem **sučelja naredbenog retka** (eng. *Command Line Interface*, skraćeno *CLI*).

U koji od navedenih dijelova računalnog sustava uopće spada naredbeni redak? Argumentirajte odgovor.

► Spoiler alert! Odgovor na pitanje

Naredbeni redak možemo svrstati u korisničku razinu računalnog sustava. Premda nam omogućuje pristup resursima operacijskog sustava i interakciju s njime, **naredbeni redak je prije svega sučelje koje koristi krajnji korisnik** za upravljanje računalom.

Štoviše, kada koristimo naredbeni redak, u pozadini često rade brojni primjenski programi (kao i u slučaju GUI-a) koji nam omogućuju pristup operacijskom sustavu, a često nismo ni svjesni njihove prisutnosti.

2. Naredbeni redak (CLI)

Naredbeni redak (eng. *Command Line Interface*, skraćeno *CLI*) je tekstualno sučelje koje omogućava korisnicima unos i izvršavanje naredbi u operacijskom sustavu. Za razliku od grafičkog korisničkog sučelja (eng. *Graphical User Interface*, skraćeno *GUI*), koje koristi miš i vizualne elemente za interakciju s korisnikom, naredbeni redak **koristi isključivo tekstualne naredbe**.

[Prvi interaktivni naredbeni reci pojavili su se još 60-ih godina prošlog stoljeća](#), a sredinom 70-ih i početkom 80-ih godina počinju se **standardizirati**.

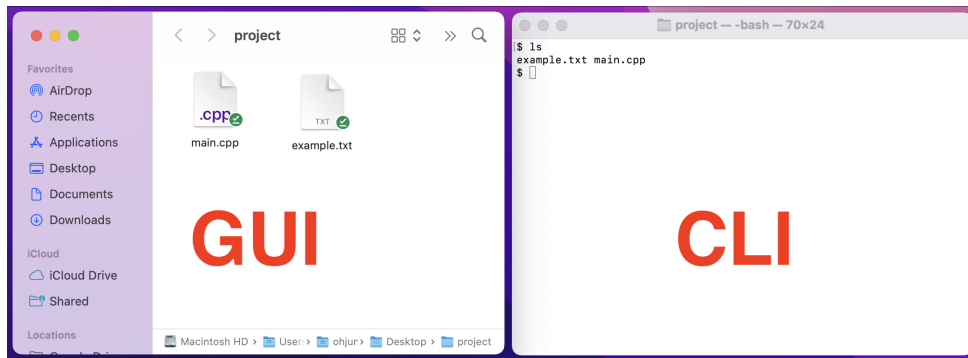
Premda se većina današnjih računalnih sustava oslanja na GUI umjesto CLI, mnogi aplikacijski i pomoćni programi nemaju korespondirajući GUI, već se koriste isključivo kroz CLI.

CLI ima mnogo sinonima i manjih varijacija, međutim **za početak je dovoljno znati da se radi o tekstualnom sučelju koji se bazira na unosu naredbi i prikazu rezultata izvršavanja tih naredbi**.

Za CLI se često koriste i sljedeći "sinonimi" (premda značenje nije potpuno isto, više u nastavku!):

- Terminal
- Shell
- Console
- Command prompt
- Command line
- PowerShell
- Interpreter
- i mnogi drugi...

U nastavku više o razlikama između ovih pojmova.



GUI vs CLI

Kao generalni pojam, u ovim skriptama najčešće će se upotrebljavati termin **naredbeni redak**, odnosno skraćena verzija: **CLI**.

2.1 Bash - Bourne Again Shell

Rekli smo da se termini CLI, terminal i shell često upotrebljavaju kao sinonimi, različiti nazivi za istu stvar. Međutim, ako malo zagrebemo ispod površine, možemo definirati neke razlike između ovih pojmova:

Terminal je **program** koji pruža korisniku tekstualno sučelje/okruženje za interakciju s računalom.

- dopušta korisniku unos naredbi i prikazuje rezultate izvršavanja tih naredbi
- možemo ga zamisliti kao **omotač** (eng. *wrapper*) oko *shell*a koji nam omogućava da unosimo naredbe
- ne obrađuje naredbe, već služi kao posrednik (medij)

Moderni terminali su: **GNOME Terminal**, **Konsole**, **iTerm2**, **Alacritty**, **Windows Terminal**...

 **Analogija:** Terminal je alat, poput ekrana i tipkovnice, koji koristimo za komunikaciju s računalom.

CLI je generalni **koncept** koji se odnosi na bilo koje tekstualno sučelje koje omogućava korisnicima interakciju s računalom putem naredbi.

- CLI radi unutar terminala
- **nije program, već način interakcije s računalom** (kao što je i GUI način interakcije)
- mnogi programi imaju svoje vlastito CLI sučelje, npr. **git**, **npm**, **docker**, **python**, **aws**...

 **Analogija:** CLI je poput jezika (npr. Engleski, Hrvatski) koji koristimo za svakodnevni razgovor. To je metoda interakcije.

Shell (hrv. *ljuska*) se odnosi na **specifični program** koji interpretira naredbe koje korisnik unosi unutar terminala.

- služi kao **posrednik između korisnika i operacijskog sustava**
- **omogućava interakciju s OS-om**
- može biti **interaktivan** (unosom tekstualnih naredbi) ili **skriptni** (izvršavanje napisanih shell skripti)

Primjeri poznatih shellova:

- **Bourne shell** (*sh*) - jedna od prvih i najpoznatijih shell implementacija (70-ih godina)
- **Bash** (*Bourne Again Shell*) - najpopularniji shell za Unix i *Unix-like* sustave
- **Zsh** (*Z shell*) - moderna alternativa bashu
- **Fish** (*Friendly Interactive Shell*) - još jedna moderna alternativa bashu
- **tcsh** (*Tenex C Shell*) - starija alternativa bashu
- **CMD** (*Command Prompt*) - stari shell za Windows OS
- **PowerShell** - novi shell za Windows OS
- i mnogi drugi...

📌 **Analogija:** Shell je poput prevoditelja koji interpretira/prevodi naš jezik i sudjeluje u konverzaciji.

Pitanje:

U gornjim opisima *shellova* spominje se pojam Unix i *Unix-like* sustavi. Što je to Unix i što znači pojam *Unix-like* sustavi?

► Spoiler alert! Odgovor na pitanje

[Unix](#) je operacijski sustav koji je razvijan u [Bell Labsu](#) sredinom 60-ih i početkom 70-ih godina. Unix je bio jedan od prvih operacijskih sustava koji je podržavao višekorisnički i višezadačni rad, a njegova arhitektura i dizajn bili su temelj za mnoge moderne operacijske sustave.

Unix je bio zatvorenog koda, ali je kasnije razvijena otvorena verzija pod nazivom BSD ([Berkeley Software Distribution](#)) koja je postala temelj za mnoge *Unix-like* operacijske sustave.

Unix-like sustavi su operacijski sustavi koji su dizajnirani na temelju Unixa, ali nisu nužno kompatibilni s njim. To znači da su *Unix-like* sustavi inspirirani Unixom, ali su razvijeni odvojeno i ne dijele kod s Unixom.

Najpoznatiji *Unix-like* sustavi su **Linux** i **macOS**

Za lakše razumijevanje, spomenute razlike sažete su u tablici:

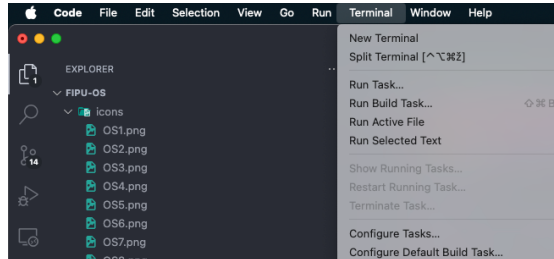
Ključne razlike između terminala, CLI-ja i shella


| Koncept | Objašnjenje | Primjeri |
|--|---|---|
| Terminal | Program koji pruža tekstualno sučelje, ali ne obrađuje naredbe. | Windows Terminal, Command Prompt, GNOME Terminal, Konsole, iTerm2 |
| CLI (<i>hrv. Naredbeni redak</i>) | Način interakcije sa sustavom kroz naredbe, suprotnost je GUI. | Git CLI, Docker CLI, Linux CLI |
| Shell (<i>hrv. ljuška</i>) | Program koji interpretira naredbe i vrši interakciju s operacijskim sustavom. | bash, zsh, PowerShell |

Napomena: neke programe možemo interpretirati i kao CLI i kao shell, ali i kao terminale. Primjer takvog programa je `Command Prompt (cmd.exe)` u Windows sustavu. On je prvenstveno shell jer interpretira naredbe, a kao sučelje može koristiti `Windows Terminal`, standalone (vlastiti terminal), `PowerShell` pa i `VS Code terminal`. Isto vrijedi i za mnoge druge *shellove*.

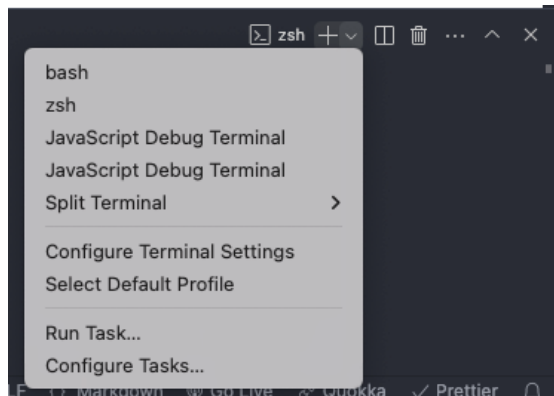
Primjer za lakše razumijevanje ovih razlika:

Ako otvorite VS Code okruženje, možete pokrenuti novi **Terminal**.



 VS Code → New Terminal

Unutar tog terminala možete pokrenuti **shell** (npr. `bash`) i koristiti ga kao **CLI** za izvršavanje naredbi.



 Terminal → bash

Međutim, uočite i opcije poput: `JavaScript Debug Terminal`, `Python Debug Console`. Ovo su također terminali, ali su specifični za određene programske jezike i koriste se za **debugiranje**, ne kao CLI za interakciju s OS-om ili nekim drugim programom.

Zaključno: Unutar istog terminala možemo koristiti različite *shellove* i CLI-eve, ovisno o potrebama i preferencijama.

Mi ćemo na vježbama učiti **bash** (*Bourne Again Shell*), najpopularniji shell za Unix i *Unix-like* sustave.



 bash - Bourne Again Shell, najpopularniji shell za Unix i *Unix-like* sustave

2.2 Kako pokrenuti bash?

Bash (*Bourne Again Shell*) je popularni Unix shell za Unix i *Unix-like* sustave. Razvio ga je **Brian Fox** 1989. godine kao dio [GNU projekta](#), s ciljem pružanja poboljšane i slobodne alternative originalnom Bourne shellu (*sh*). GNU projekt, koji je započeo Richard Stallman 1983. godine, razvio je niz alata za operacijski sustav GNU. Iako GNU nije imao vlastiti kernel, kombiniranjem njegovih komponenti s Linux kernelom, koji je 1991. razvio Linus Torvalds, nastao je **GNU/Linux operacijski sustav** (poznat kao Linux).

Bash shell dolazi unaprijed instaliran na većini *Unix-like* sustava, **uključujući većinu distribucija Linuxa**. Na macOS-u je bio zadani shell do verzije 10.15 (Catalina), nakon čega ga je Apple zamijenio **Z shellom** (*zsh*).

Unix-like OS

Ako koristite *Unix-like* OS, jednostavno otvorite novi terminal i unesite naredbu `bash`:

```
→ bash
```

Ovo će pokrenuti bash shell unutar aktivnog terminala.

```
bash-3.2$
```

Windows OS

Ako koristite Windows OS, postoji više načina za pokretanje bash *shell*a.

Svakako se preporučuje korištenje novog [Windows Terminala](#), umjesto starijeg cmd-a.

Preporučeni način rada je koristeći **Windows Subsystem for Linux (WSL)**. [WSL](#) je Linux okruženje koje se može instalirati na Windows OS-u i omogućuje pokretanje Linux distribucija unutar Windowsa, samim time i korištenje bash *shell*a.

Otvorite `Windows Terminal` ili `PowerShell` **kao administrator** i unesite sljedeću naredbu kako biste instalirali WSL:

```
→ wsl --install
```

Ako dobivate grešku, moguće da je već instaliran. U tom slučaju, izlistajte se dostupne distribucije:

```
→ wsl --list --verbose
```

Trebali biste dobiti popis instaliranih Linux distribucija, primjerice:

| NAME | STATE | VERSION |
|----------|---------|---------|
| * Ubuntu | Stopped | 2 |

Zvezdica `*` označava zadanu distribuciju

Ako slučajno nemate niti jednu distribuciju, možete ručno instalirati `Ubuntu` distribuciju:

```
→ wsl --install -d Ubuntu
```

Ako ih imate više, postavite zadanu na `Ubuntu`:

```
→ wsl --set-default Ubuntu
```

Nakon prve instalacije, možda ćete morati ponovno pokrenuti računalo.

Napokon, pokrenite WSL:

```
→ wsl
```

To je to! Kako biste se uvjerali da ste u bash shellu, unesite naredbu:

```
→ echo $0
```

Ako se ispiše `bash`, znači da ste u bash shellu i možete početi s radom. 😎

3. Osnovne bash naredbe

U ovom dijelu ćemo naučiti nekoliko osnovnih bash naredbi koje će nam pomoći u svakodnevnom radu s naredbenim retkom. Za sada se nećemo baviti složenim naredbama (niti zastavicama).

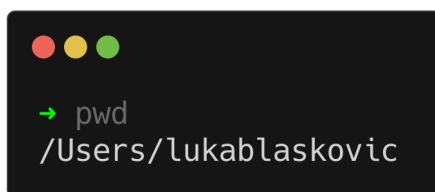
Svaki put kad otvorimo terminal, zapravo se nalazimo u određenom direktoriju na računalu. Direktorij u kojem se trenutno nalazimo naziva se **radni direktorij** (eng. *working directory*).

Prema tome, nije loše započeti upravo od toga, saznati gdje se trenutno nalazimo!

pwd

pwd (zapamti kao: *print working directory*) → **ispisuje trenutni radni direktorij kao apsolutnu putanju**

→ pwd



 Naredba **pwd** ispisuje apsolutnu putanju radnog direktorija

Apsolutna putanja (eng. *Absolute Path*) je putanja koja počinje od korijenskog direktorija (`/` na *Unix-like* sustavima) i vodi do trenutnog direktorija. Na primjer, `/home/<user>/Desktop`.

- **Apsolutna putanja je neovisna o radnom direktoriju.**

Apsolutna putanja na Windowsu počinje s simbolom diska, npr. `C:\Users\<vaš_username>\Desktop`. Međutim, obzirom da koristimo WSL, dobivamo apsolutnu putanju u Unix stilu s prefiksom `/mnt/` koji označava *mount point* za pristup datotečnom sustavu Windowsa unutar Linux okruženja.

Primjeri apsolutnih putanji:

```
/mnt/c/Users/Marko # Windows (WSL)
/Users/Marko # macOS
/home/Marko # Linux
C:\Users\Marko # Windows
```

- `/mnt/c` je prefiks koji označava `c:` disk na Windowsu, ali *mountan* kao `/` u WSL-u
- **Važno!** Uočite obrnuti redoslijed separatora (`\` na Windowsu, `/` na *Unix-like* sustavima)!

Relativna putanja (eng. *Relative Path*) je putanja koja se odnosi na trenutni radni direktorij. Na primjer, ako se nalazimo u direktoriju `/home/<user>`, relativna putanja do direktorija `Desktop` je `Desktop`.

Primjeri relativnih putanji:

Desktop # relativna putanja do direktorija Desktop ako se nalazimo u /home/<user> odnosno /mnt/c/Users/<user>
marko/Desktop # relativna putanja do direktorija Desktop ako se nalazimo u /home odnosno /mnt/c/Users

ls

ls (zapamti kao: *list*) → **ispisuje sadržaj direktorija (direktoriji + datoteke)**

Sintaksa:

→ **ls**

Primjeri:

→ **ls** # ispisuje sadržaj trenutnog radnog direktorija

ili

→ **ls** <relativna_putanja_do_direktorija>

Primjer: ispisuje sadržaj direktorija "Desktop"

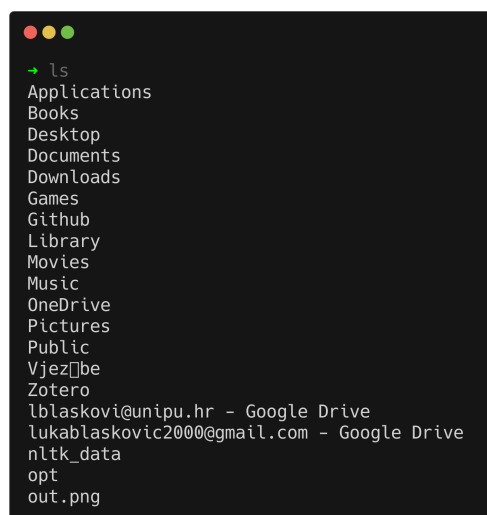
→ **ls** Desktop

ili

→ **ls** <apsolutna_putanja_do_direktorija>

Primjer: ispisuje sadržaj direktorija "Pictures"

→ **ls** /mnt/c/Users/<user>/Pictures




```
→ ls
Applications
Books
Desktop
Documents
Downloads
Games
Github
Library
Movies
Music
OneDrive
Pictures
Public
Vježbe
Zotero
lblaskovi@unipu.hr - Google Drive
lukablaskovic2000@gmail.com - Google Drive
nlTK_data
opt
out.png
```



Ako koristite **ls** bez argumenata, ispisat će se sadržaj trenutnog radnog direktorija.

```
→ ls /Users/lukablaskovic
Applications
Books
Desktop
Documents
Downloads
Games
Github
Library
Movies
Music
OneDrive
Pictures
Public
Vjezbe
Zotero
laskovi@unipu.hr - Google Drive
lukablaskovic2000@gmail.com - Google Drive
nlk_data
opt
```

 Ako koristite `ls` s relativnom ili apsolutnom putanjom, ispisat će se sadržaj direktorija na toj putanji.

cd

`cd` (zapamti kao: *change directory*) → **mijenja trenutni radni direktorij**

Sintaksa:

```
cd <putanja>
```

Primjeri:

```
→ cd <relativna_putanja_do_direktorija>
# Primjer: prebaci se u direktorij Desktop (uz pretpostavku da se nalazimo u
/mnt/c/Users/<user>)
→ cd Desktop

# ili

→ cd <apsolutna_putanja_do_direktorija>
# Primjer: prebaci se u direktorij Desktop (Windows) prema apsolutnoj putanji
→ cd /mnt/c/Users/<user>/Desktop
```

Ako direktorij ne postoji, dobivamo grešku:

```
→ cd: no such file or directory: "direktorij"
```

 Hint: Korisno je raditi česte provjere radnog direktorija koristeći `pwd` i `ls` naredbe.

Kako bismo se vratili u prethodni direktorij, koristimo posebnu oznaku `..` koja označava roditeljski direktorij (eng. *parent directory*):

```
→ cd ..
```

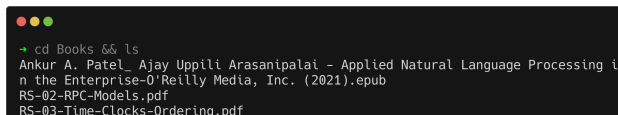
Možemo kombinirati više Bash naredbi koristeći AND operator `&&` (ovo je korisnije kod skriptiranja)

Na primjer, želimo se premjestiti u direktorij `Books` i ispisati njegov sadržaj, sve u jednoj naredbi:

```
→ cd Books && ls
```

ipak, bolje napisati samo naredbu s argumentom

```
→ ls Books
```



```
→ cd Books && ls
Ankur A. Patel_ Ajay Upplil Arasanipalal - Applied Natural Language Processing i
n the Enterprise-O'Reilly Media, Inc. (2021).epub
RS-02-RPC-Models.pdf
RS-03-Time-Clocks-Ordering.pdf
```

 Ispisivanje sadržaja direktorija "Books"

clear

Sad smo si nakrcali terminal sa svim ovim naredbama i ispisima. Kako bismo ga očistili, koristimo naredbu

`clear`:

`clear` → briše sadržaj terminala

Napomena: `clear` naredba ne briše povijest naredbi, već samo trenutni sadržaj terminala. Također, naredba ne vraća korisnika u početni položaj odnosno putanja i povijest naredbi ostaju nepromijenjeni.

💡 Hint: Povijest naredbi možemo pregledavati koristeći tipku `↑` (strelica prema gore) i `↓` (strelica prema dolje) na tipkovnici.

Sintaksa:

```
→ clear
```

echo

Prethodno smo već koristili `echo` naredbu u kontekstu provjere trenutnog *shell*a. Općenito, `echo` naredba se koristi za ispisivanje teksta u terminalu.

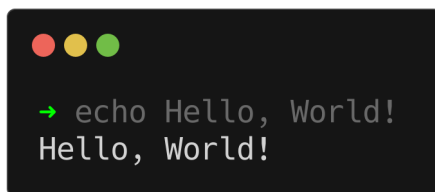
`echo` → ispisuje tekst u terminalu

Sintaksa:

```
→ echo '<tekst>'
```

Primjer:


```
→ echo 'Hello, World!'
```



```
→ echo Hello, World!  
Hello, World!
```



Ispis pišemo unutar navodnika ili bez navodnika, naredbom `echo`

Moguće je koristiti jednostruke i dvostruke navodnike, ali budite oprezni s dvostrukim navodnicima jer omogućuju interpretaciju posebnih znakova poput `$`, `\`, `!` i drugih, što može dovesti do neočekivanog ponašanja.

```
→ echo "Hello, World!" # greška
```

- Greška! Zbog specijalnog znaka `!` dolazi do greške:

```
bash: !": event not found
```

Naredbu je moguće koristiti i bez navodnika, ali je preporučljivo koristiti ih kako bi se izbjegle eventualne greške.

Bash je programski jezik za obradu naredbi, što znači da možemo koristiti varijable, petlje, uvjetne izraze, funkcije i sve druge konstrukte programskih jezika (više u nastavku vježbi).

Vrijednosti varijabli možemo ispisivati koristeći naredbu `echo` i sintaksu `$<naziv_varijable>`.

Primjer:

```
→ echo $0
```

Ova naredba će ispisati naziv aktivnog *shella*.

Postoji mnoštvo ugrađenih (već pohranjenih) varijabli sa specijalnim nazivima koje možemo ispisivati:

- `$USER` - trenutni korisnik
- `$HOME` - putanja do korisničkog direktorija
- `$PWD` - trenutni radni direktorij
- `$HOSTNAME` - ime računala (domaćina)

Moguće je i kombinirati tekst i varijable, **ali samo unutar dvostrukih navodnika**:

```
→ echo 'Trenutni korisnik je: $USER'
# Ispisuje: Trenutni korisnik je: $USER
→ echo "Trenutni radni direktorij je: $PWD"
#Ispisuje: Trenutni radni direktorij je: koji već je...
```

3.1 Osnovne manipulacije datotekama/direktorijima

Pokazali smo nekoliko osnovnih naredbi za navigaciju po direktorijima i ispisivanje teksta. Sada ćemo naučiti nekoliko naredbi za manipulaciju datotekama odnosno direktorijima.

mkdir

mkdir (zapamti kao: *make directory*) → **stvara novi direktorij na danoj putanji**

Sintaksa:

```
→ mkdir <naziv_direktorija>
```

Primjeri:

```
# Primjer: stvara direktorij "new_dir" u trenutnom radnom direktoriju
→ mkdir new_dir

# ili

→ mkdir <relativna_putanja_do_novog_direktorija>
# Primjer: stvara direktorij "new_dir" unutar Desktop direktorija
→ mkdir Desktop/new_dir

# ili

→ mkdir <apsolutna_putanja_do_novog_direktorija>
# Primjer: stvara direktorij "new_dir" na danoj apsolutnoj putanji, bez obzira gdje se
trenutno nalazimo
→ mkdir /home/user/Documents/new_dir
```

Napravit ćemo novi radni direktorij `files_manipulation` i unutar njega direktorij `test`:

```
→ mkdir files_manipulation
→ cd files_manipulation
→ mkdir test

# ali

# ne možemo napraviti direktorij unutar nepostojećeg direktorija (odnosno 2 ili više
direktorija odjednom)
→ mkdir files_manipulation/test
```

💡 Hint: Kada pišete naziv postojećeg direktorija, možete započeti upisivati te upotrijebiti tipku `Tab` za automatsko dovršavanje naziva.

npr. `cd fi` + `Tab` = `cd files_manipulation`

`rmdir`

`rmdir` (zapamti kao: *remove directory*) → **briše direktorij na putanji, ako je prazan**

Sintaksa:

```
→ rmdir <naziv_direktorija>
```

Primjeri:

```
# Primjer: briše direktorij "some_dir", ako je prazan
→ rmdir some_dir

# ili

→ rmdir <relativna_putanja_do_direktorija>
# Primjer: briše direktorij "some_dir" unutar "Desktop" direktorija, ako je prazan
→ rmdir Desktop/some_dir

# ili

→ rmdir <apsolutna_putanja_do_novog_direktorija>
# Primjer: briše direktorij "some_dir" na danoj apsolutnoj putanji, ako je prazan (bez
obzira gdje se trenutno nalazimo)
→ rmdir /home/user/Documents/some_dir
```

`touch`

`touch` → **stvara novu praznu datoteku na danoj putanji**

Sintaksa:

```
→ touch <naziv_datoteke>.<ekstenzija>
```

- stvara praznu datoteku s navedenim imenom i ekstenzijom na danoj putanji
- npr. ako želimo stvoriti tekstualnu datoteku, koristimo ekstenziju `.txt`, ako želimo novu JavaScript datoteku, koristimo ekstenziju `.js`
- može se koristiti i bez ekstenzije, u tom slučaju se jednostavno stvara datoteka bez ekstenzije

Primjeri:

```
→ touch test.txt
→ touch index.html
→ touch script.js
→ touch main.py
```

Na isti način kao i kod `mkdir`, možemo koristiti `touch` naredbu za stvaranje datoteka u drugim direktorijima:

```
# Primjer: stvara datoteku "test.txt" unutar "Desktop" direktorija (uz pretpostavku da se
nalazimo u /mnt/c/Users/<user>)
→ touch Desktop/test.txt

# Primjer: stvara datoteku "test.txt" na danoj apsolutnoj putanji, bez obzira gdje se
trenutno nalazimo
→ touch /mnt/c/Users/<user>/Desktop/test.txt
```

cp

cp (zapamti kao: *copy*) → **kopira datoteku iz mjesta** `<izvor>` **u neko** `<odredište>`

Sintaksa:

```
→ cp <izvor> <odredište>
```

- gdje izvor i odredište mogu biti relativne ili apsolutne putanje

Primjeri:

```
→ cp test.txt subfolder # kopira datoteku "test.txt" u direktorij "subfolder". Nova
datoteka je istog naziva

→ cp text.txt text_copy.txt # kopira datoteku "text.txt" u istom direktoriju i naziva je
"text_copy.txt"

→ cp README.md /mnt/c/Users/<user>/Documents # kopira datoteku "README.md" u Documents
direktorij prema apsolutnoj putanji. Nova datoteka je istog naziva.
```

Primjer s kombiniranjem naredbi:

```
→ cp test.txt test_copy.txt && ls # kopira datoteku "test.txt" u radni direktorij i naziva
je "test_copy.txt", a zatim ispisuje sadržaj direktorija
```

```
→ cp test.txt test_copy.txt && ls  
test.txt test_copy.txt
```



Primjer kombiniranja naredbi `cp` i `ls` (kopiraj pa listaj)

Ova naredba će kopirati datoteku `test.txt` i napraviti novu datoteku `test_copy.txt` u trenutnom radnom direktoriju. Ako bismo htjeli novu datoteku kopirati u neki drugi direktorij, **jednostavno koristimo putanju do te datoteke** kao `<odredište>`.

Recimo da se nalazimo u direktoriju `files_manipulation` i želimo kopirati datoteku `file.txt` u direktorij `test`:

1. radimo novi direktorij
2. radimo novu datoteku u trenutnom direktoriju
3. kopiramo datoteku u novoizrađeni direktorij

```
→ pwd # /Users/lukablaskovic/Documents/files_manipulation  
→ mkdir test  
→ touch file.txt  
→ cp file.txt test  
→ ls test
```

Odnosno listanjem naredbi s AND operatorom:

```
→ mkdir test && touch file.txt && cp file.txt test && ls test
```

```
→ mkdir test && touch file.txt && cp file.txt test && ls test  
file.txt
```



Primjer stvaranja direktorija pa datoteke, kopiranja datoteke u novi direktorij i ispisivanja sadržaja direktorija

mv

mv (zapamti kao: *move*) → premješta **datoteku** ili **direktorij** iz mjesta `<izvor>` u neko `<odredište>` ili preimenuje datoteku/direktorij ako ne postoji `<odredište>`

Sintaksa:

```
→ mv <izvor> <odredište>
```

- gdje izvor i odredište mogu biti relativne ili apsolutne putanje

Primjeri premještanja:

```
# Primjer: premješta datoteku "test_copy.txt" u direktorij `test`  
→ mv test.txt test  
  
# ili  
  
# Primjer: premješta datoteku "test" u Documents direktorij prema apsolutnoj putanji  
→ mv test.txt /mnt/c/Users/<user>/Documents  
  
# ili  
  
# Primjer: premješta direktorij "dir_1" u direktorij "dir_2"  
→ mv dir_1 dir_2
```

Primjer preimenovanja:

```
# Primjer: preimenuje datoteku "test.txt" u "test_copy.txt"  
→ mv test.txt test_copy.txt  
  
# ili  
  
# Primjer: preimenuje direktorij "dir_1" u "dir_2", samo ako "dir_2" ne postoji  
→ mv dir_1 dir_2
```

rm

rm (remove) → briše **datoteku** na navedenoj putanji

Sintaksa:

```
→ rm <naziv_datoteke>
```

- gdje izvor i odredište mogu biti relativne ili apsolutne putanje

Primjer:

```
→ rm test.txt
```

- briše datoteku `test.txt`, ako postoji u trenutnom radnom direktoriju

3.2 Tablica osnovnih bash naredbi

| Naredba | Sintaksa | Objašnjenje | Primjer |
|------------------|---|--|------------------|
| <code>pwd</code> | <code>pwd</code> | Ispisuje apsolutnu putanju trenutnog radnog direktorija. | <code>pwd</code> |
| | <code>ls, ls <rel_putanja>, ls</code> | Ispisuje sadržaj direktorija na putanji (radni | |

| | | | |
|-------|---|---|----------------------------------|
| ls | <abs_putanja> | direktorij, relativna ili apsolutna) | ls |
| cd | cd <rel_putanja>, cd <abs_putanja> | Mijenja trenutni radni direktorij. | cd Documents |
| cd .. | cd .. | Vraća se u roditeljski direktorij. | cd .. |
| clear | clear | Briše sadržaj terminala, ali ne i povijest naredbi. | clear |
| echo | echo "<tekst>", echo \$varijabla | Ispisuje tekst u terminalu. Može ispisivati i varijable simbolom \$ | echo "Hello, World!", echo \$VAR |
| mkdir | mkdir <naziv>, mkdir <rel_putanja_i_naziv>, mkdir <abs_putanja_i_naziv> | Stvara novi direktorij na danoj putanji. | mkdir test |
| rmdir | rmdir <naziv>, rmdir <rel_putanja_i_naziv>, rmdir <abs_putanja_i_naziv> | Briše prazan direktorij na danoj putanji. | rmdir test |
| touch | touch <naziv>, touch <rel_putanja_i_naziv>, touch <abs_putanja_i_naziv> | Stvara novu praznu datoteku. Argument se može upotpuniti putanjom do nove datoteke. | touch test.txt |
| cp | cp <izvor> <odredište> | Kopira datoteku s nekog izvora u odredište. | cp test.txt test_copy.txt |
| mv | mv <izvor> <odredište> | Premješta datoteku/direktorij u neko odredište. Ako ne postoji, preimenuje. | mv test_copy.txt test |
| rm | rm <naziv>, rm <rel_putanja_i_naziv>, rm <abs_putanja_i_naziv> | Briše datoteku na danoj putanji. | rm test.txt |

Zadaci za Vježbu 1

Instalirajte WSL na Windows OS-u i pokrenite bash shell, ili ga direktno pokrenite u terminalu ako koristite *Unix-like* OS.

Za svaki zadatak, koristite bash naredbe koje smo prošli u ovoj skripti.

Napomena: Za svaku točku zadatka pišite naredbe koje se od vas traže, ako se traži više naredba pod istom točkom (npr. `cd` i `ls`), pišite ih u jednoj liniji odvojene operatorom `&&`.

Zadatak 1

1. Provjerite trenutni radni direktorij
2. Izlistajte sadržaj trenutnog radnog direktorija
3. Napravite novi direktorij `vjezba1` i prebacite se u njega
4. Unutar direktorija `vjezba1` napravite novu datoteku `README.md`
5. Vratite se u početni radni direktorij

Zadatak 2

1. Napravite praznu datoteku `file.txt` unutar direktorija `vjezba2`
2. Kopirajte datoteku `file.txt` u direktorij `vjezba2` i nazovite ju `file_copy.txt`
3. Ispišite sve datoteke unutar direktorija `vjezba2`
4. Obrišite datoteku `file.txt` i vratite se u početni radni direktorij
5. Pokušajte izbrisati direktorij `vjezba2`. Zašto ne možete?

Zadatak 3

1. Napravite novi direktorij `vjezba3` i unutar njega direktorij `backup`
2. Unutar direktorija `vjezba3` napravite 3 datoteke: `notes.txt`, `todo.txt` i `script.sh`
3. Kopirajte sve datoteke iz direktorija `vjezba3` u direktorij `backup`
4. Izbrišite samo datoteku `script.sh` iz direktorija `vjezba3` i ispišite sve datoteke
5. U backup dodajte još jedan direktorij koju ćete imenovati `USER` varijablom
6. Premjestite sve datoteke iz direktorija `backup` u direktorij nazvan varijablom `USER`

Zadatak 4

1. Napravite novi direktorij `vjezba4` i unutar njega direktorij `subfolder`
2. Unutar direktorija `vjezba4` napravite datoteku prema nazivu varijable `HOSTNAME`
3. Preimenujte novoizrađenu datoteku prema nazivu varijable `USER`
4. Premjestite datoteku `USER` u direktorij `subfolder`
5. Izbrišite datoteku `USER` koristeći apsolutnu putanju

Zadaću predajete prema [uputama na Merlin platformi](#).