

Sistemsko programiranje

Kopiranje datoteka u Python-u uz pomoć shutil modula

Kristijan Koščak

Osijek 2020.

Sadržaj:

1. Uvod:.....	1
2. Općenito o Pythonu:	2
3. Shutil modul	3
3.1. Funkcija copyfileobj(fsrc, fdst[, length]) i algoritam	3
3.2. Funkcija copyfile(src, dst, *, follow_symlinks=True) i algoritam.....	4
3.3. Funkcija copy(src, dst, *, follow_symlinks=True) i algoritam	6
3.4. Funkcija copy2(src, dst, *, follow_symlinks=True) i algoritam	8
4. Osvrt na laboratorijsku vježbu 1	9
4.1. Kreiranje datoteka	9
4.2. Kopiranje datoteka.....	9
4.3. Rezultati izvođenja	10
5. Zaključak.....	12
6. Literatura:	13

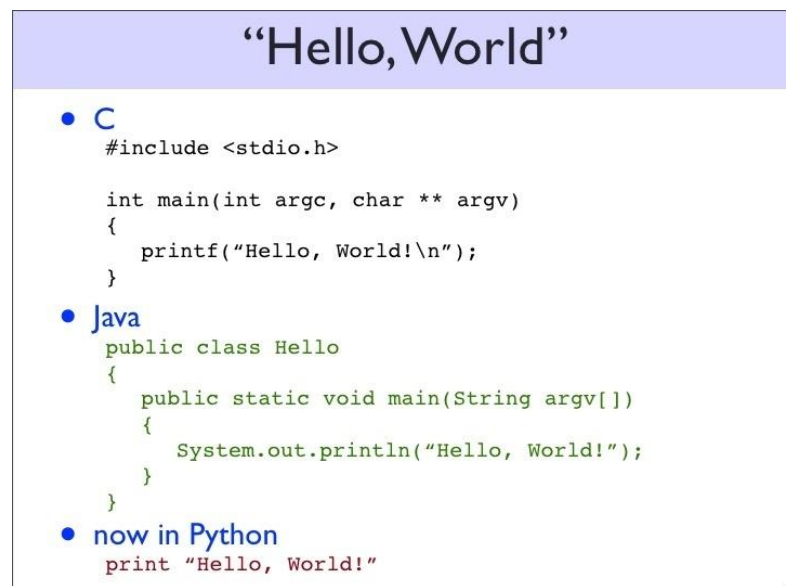
1. Uvod:

Cijeli operacijski sustav sastoji se mnogo datoteka. U računarstvu, datoteka predstavlja skup binarnih podataka pohranjenih na nekakvom mediju(npr. disku računala). Bilo kakve informacije mogu biti pohranjene u datotekama. Svaka datoteka posjeduje svoju veličinu koja je izražena u bajtovima, i to u obliku cijelih brojeva. Veličina ovisi o operacijskom sustavu ili o fizičkim svojstvima medija na kojem se datoteka nalazi. Svaka datoteka ima različitu svrhu pa prema tome postoji nekoliko različitih tipova datoteka. U datoteku možemo pohraniti sliku, tekstualnu poruku, video, računalni program itd. Raznolikost vrsta podataka je velika. Koristeći razne programe, korisnici mogu otvarati, čitati, izmjenjivati, spremati i zatvoriti datoteke. Osim navedenog korisnici mogu kopirati i brisati datoteke. Kopiranje datoteka jedna je od osnovnih radnji koje podržava svaki operacijski sustav. Datoteke su organizirane u datotečnom sustavu koji služi kako bi vodio brigu o lokaciji datoteka na disku i omogućio korisniku pristup istima.

Ideja ovog seminara je približiti postupak kopiranja datoteka na Windows OS-u koristeći Python i modul Shutil. Također, osvrnut ćemo se na laboratorijsku vježbu 1 u kojoj smo kopirali datoteke različitih veličina, koristeći različite veličine spremnika(engl. *buff size*) i različite algoritme.

2. Općenito o Pythonu:

Python je popularni programski jezik. Primjenjuje se u razvoju internetskih aplikacija uz pomoć raznih razvojnih okruženja, znanstvenim i numeričkim računanjima, u edukacijskim svrhama, razvoju grafičkih korisničkih sučelja, razvoju softvera i poslovnim aplikacijama. Kao što možemo vidjeti, primjena je stvarno velika i to u područjima koja se konstantno razvijaju. Python, programerima dopušta koristiti različite stilove programiranja kao što su OOP, strukturalno i aspektno orijentirano programiranje. Budući da je on interpreterski jezik, programi se vrše malo sporije u odnosu na jezike kao što su C, C++ i slični. Često se uspoređuje sa Javom jer su oboje interpreterski jezici te su znatno slabiji u području podrške za višejezrovno izvođenje programa. Oboje koriste samo jednu procesorsku jezgru. Brzina izvođenja programa navedenih dvoje je približno jednaka, ovisi o programu. Mogućnosti Pythona su izuzetno velike. Bitna stvar koju treba spomenuti jer je vezana za projekt su upravljanje datotekama. Također, podržava rad s velikim datotekama. Sintaksa je približena engleskom jeziku. Osim toga, ona je jedna od bitnijih razlika u odnosu na spomenute jezike jer ne koristi vitičaste zagrade i ključne riječi za razlikovanje programskih blokova. Umjesto toga koristi uvlačenje koje predstavlja novi, ugnježđeni blok, a smanjenje kraj bloka. To dovodi do pisanja programa s nekoliko linija koda manje u odnosu na ostale jezike. Napisani kod može biti izvršen trenutno jer radi na interpreterskom sustavu. Važno je spomenuti kako Python radi na različitim platformama kao što su Windows, Mac, Linux itd.



Slika 1 - Primjer istog programa u različitim jezicima. Literatura: [6]

3. Shutil modul

U mnogim programskim jezicima koriste se različite biblioteke unutar kojih se nalaze različite funkcije. Njihova uloga je olakšati korisniku pisanje programa na način da korisnik uključivanjem biblioteke može koristiti te funkcije umjesto da funkcionalnosti pojedinih funkcija piše sam. Shutil modul je ubiti biblioteka koja pruža razne mogućnosti upravljanja datotekama i direktorijima. Konkretno, mogućnosti koje uključuje su kopiranje i brisanje datoteka i direktorija. U ovom seminaru bazirat ćemo se na kopiranju datoteka budući da su one ujedno i ideja ovog seminara. Shutil nudi nekoliko različitih načina kopiranja datoteka koje ćemo navesti u sljedećim poglavljima. Kako bi ga koristiti potrebno je u kod uključiti Shutil i to na način da prvo unesemo ključnu riječ „import“ te nakon toga ime biblioteke.

```
# importing shutil module
import shutil
```

Slika 2 - Uključivanje shutil modula u programski kod.

3.1. Funkcija `copyfileobj(fsrc, fdst[, length])` i algoritam

Prva funkcija koju ćemo koristiti zove se `copyfileobj` i prima dva parametra uz mogućnost trećeg. Prva dva parametra (`fsrc` i `fdst`) predstavljaju datoteke. Treći parametar (`length`) predstavlja veličinu spremnika. Ukoliko se veličina ne preda vrijednost joj je 0. Nadalje, proći ćemo kroz algoritam navedene funkcije.

```
def copyfileobj(fsrc, fdst, length=0):
    """copy data from file-like object fsrc to file-like object fdst"""
    # Localize variable access to minimize overhead.
    if not length:
        length = COPY_BUFSIZE
    fsrc_read = fsrc.read
    fdst_write = fdst.write
    while True:
        buf = fsrc_read(length)
        if not buf:
            break
        fdst_write(buf)
```

Slika 3 - Algoritam `copyfileobj()` funkcije. Literatura: [8]

Navedeni algoritam prvo provjerava predanu dužinu spremnika. Ukoliko je nismo predali onda je 0 što se još može gledati i kao `false`. U prvom `if` uvjetu, `not false(0)` je isto što i `true` pa dužinu postavljamo na konstantu `COPY_BUFSIZE` koja je definirana kao `1024*1024` bajta ukoliko je u pitanju Windows, inače je `64*1024` bajta. Kod Python-a funkcije su first-class objekti što znači da mogu biti tretirane kao vrijednost, odnosno

možemo ih pridružiti varijablama. Gore navedeni kod pridružuje funkcije datoteka za čitanje i pisanje varijablama `fsrc_read` i `fdst_write`. Funkcija za čitanje, `read([size])` može primiti jedan parametar koji predstavlja broj bajtova (znakova) koji se vraća iz datoteke za čitanje. Ukoliko ga ne navedemo, on je -1 te funkcija vraća cijelu datoteku. Funkcija za pisanje, `write(text)` prima također jedan parametar koji predstavlja što zapisujemo u datoteku. Nadalje, dokle god možemo čitati iz datoteke dohvatiti ćemo točan broj bajtova koji je definiran veličinom spremnika tj. varijablom `length` i zapisat ih u novu datoteku.

3.2. Funkcija `copyfile(src, dst, *, follow_symlinks=True)` i algoritam

Druga funkcija zove se `copyfile` i koristi se za kopiranje sadržaja datoteka bez meta podataka. Meta podaci su podaci koji sadrže podatke o autoru dokumenta, zadnjoj izmjeni, verziji itd. Ono što predajemo navedenoj funkciji su putanja dokumenta koji kopiramo i putanja dokumenta u koji ćemo kopirati. Često, upotreba ove funkcije dovodi do greške ukoliko su te dvije putanje iste ili ukoliko putanja na datoteku u koju kopiramo nije namijenjena za pisanje. Također, mogući je neočekivani ishod ukoliko je prvi link prečac na neku drugu datoteku ili direktorij i `follow_symlinks` je `false`. U tom slučaju, kreira se nova datoteka sa imenom `scr` datoteke, umjesto kopiranja datoteke. Algoritam je malo duži u odnosu na prethodni.

```
231 def copyfile(src, dst, *, follow_symlinks=True):
232     """Copy data from src to dst in the most efficient way possible.
233
234     If follow_symlinks is not set and src is a symbolic link, a new
235     symlink will be created instead of copying the file it points to.
236
237     """
238     sys.audit("shutil.copyfile", src, dst)
239
240     if _samefile(src, dst):
241         raise SameFileError("{!r} and {!r} are the same file".format(src, dst))
242
243     file_size = 0
244     for i, fn in enumerate([src, dst]):
245         try:
246             st = _stat(fn)
247         except OSError:
248             # File most likely does not exist
249             pass
250         else:
251             # XXX What about other special files? (sockets, devices...)
252             if stat.S_ISFIFO(st.st_mode):
253                 fn = fn.path if isinstance(fn, os.DirEntry) else fn
254                 raise SpecialFileError("`%s` is a named pipe" % fn)
255             if _WINDOWS and i == 0:
256                 file_size = st.st_size
257
```

Slika 4 - Algoritam `copyfile()` funkcije. Literatura: [8]

Treći parametar (*) upućuje da su svi sljedeći parametri(u našem slučaju `follow_symlinks`), parametri ključnih riječi i mogu se dati koristeći njihovo ime, a ne kao pozicijski parametar. Drugim riječima, za 4. parametar nećemo predavati `True` ili `False` već `follow_symlinks = True`, odnosno `follow_symlinks = False`. Četvri parametar je uobičajeno `True`, odnosno istina. Ukoliko nije, ne kopira se datoteka. Prvo se provjerava jesu li datoteke iste i ako jesu, vraća se greška. Nadalje, iterira se kroz listu svih predmeta niti koji se trenutno odvijaju. Pri tome, dohvaćamo podatke o dokumentu kao što su zaštitni bitovi, uređaj, veličina datoteke(u bajtovima), vrijeme najčešćeg pristupa itd. Koristimo jedan od vraćenih podataka(`st_mode`), točnije zaštitne bitove kako bi provjerili radi li se o specijalnoj datoteci. Ako je, dobiti ćemo grešku, a ako nije onda dohvaćamo veličinu datoteke. Sljedeće se provjerava datoteka koju kopiramo. Ako je prečac, kreira novu datoteku ,ukoliko nije vrši se kopiranje na odgovarajućem operacijskom sustavu(mogućí macOS, Linux, Windows).

```

258     if not follow_symlinks and _islink(src):
259         os.symlink(os.readlink(src), dst)
260     else:
261         with open(src, 'rb') as fsrc, open(dst, 'wb') as fdst:
262             # macOS
263             if _HAS_FCOPYFILE:
264                 try:
265                     _fastcopy_fcopyfile(fsrc, fdst, posix._COPYFILE_DATA)
266                     return dst
267                 except _GiveupOnFastCopy:
268                     pass
269             # Linux
270             elif _USE_CP_SENDFILE:
271                 try:
272                     _fastcopy_sendfile(fsrc, fdst)
273                     return dst
274                 except _GiveupOnFastCopy:
275                     pass
276             # Windows, see:
277             # https://github.com/python/cpython/pull/7160#discussion_r195405230
278             elif WINDOWS and file_size > 0:
279                 _copyfileobj_readinto(fsrc, fdst, min(file_size, COPY_BUFSIZE))
280                 return dst
281
282         copyfileobj(fsrc, fdst)
283
284     return dst

```

Slika 5 - Algoritam `copyfileobj()` funkcije. Literatura: [8]

Kopiranje je slično kao i kod prethodne metode uz male razlike. Koristimo funkciju `memoryview()` koja vraća objekt memorijskog pogleda danog argumenta. Protokol spremnika pruža način pristupa unutrašnjim podacima objekta. Unutrašnji podaci su memorijsko polje ili spremnik. Stoga, memorijski pogled predstavlja siguran način za

otkrivanje protokola spremnika u Pythonu. Ovo je bitno zato što svakim izvođenjem nekakve akcije na objektu (npr. poziv neke metode objekta) Python treba kreirati kopiju tog objekta. Ukoliko se radi o velikim datotekama, nepotrebno kreiramo kopije velikih komada podataka koje nam ne koriste. Koristeći protokol spremnika, možemo dopustiti drugom objektu korištenje i modificiranje velikih podataka bez kopiranja istih. To dovodi do korištenja manje memorije i povećanja brzine izvođenja.

```
175 def _copyfileobj_readinto(fsrc, fdst, length=COPY_BUFSIZE):
176     """readinto()/memoryview() based variant of copyfileobj().
177     *fsrc* must support readinto() method and both files must be
178     open in binary mode.
179     """
180     # Localize variable access to minimize overhead.
181     fsrc_readinto = fsrc.readinto
182     fdst_write = fdst.write
183     with memoryview(bytearray(length)) as mv:
184         while True:
185             n = fsrc_readinto(mv)
186             if not n:
187                 break
188             elif n < length:
189                 with mv[:n] as smv:
190                     fdst.write(smv)
191             else:
192                 fdst_write(mv)
```

Slika 6 - Algoritam `_copyfileobj_readinto()` funkcije. Literatura: [8]

Zbog toga pozivamo funkciju za kreiranje objekta memorijskog pogleda i funkciji predajemo kao argument polje bajtova dužine `length`. Ono što smo dobili je objekt memorijskog pogleda na polje bajtova navedene dužine, odnosno spremnik. Dokle god je moguće, metodom `readinto(mv)` čitamo bajtove u unaprijed alocirani objekt i vraćamo broj pročitanih bajtova. Ukoliko je broj pročitanih bajtova manji od dužine spremnika, u novu datoteku upisujemo samo te bajtove umjesto cijele dužine spremnika jer bi onda stvarali nepotrebne praznine. U suprotnom, upisujemo cijeli spremnik. Na kraju, vraćamo putanju novonastale datoteke.

3.3. Funkcija `copy(src, dst, *, follow_symlinks=True)` i algoritam

Navedena metoda kopira datoteku `src` u datoteku ili direktorij `dst`. Oba parametra trebaju biti putanje do datoteka. Ako `dst` predstavlja direktorij, datoteka će biti kopirana u njega s nazivom datoteke `src`. Metoda `copy` vraća putanju do novonastale datoteke. Kod ove

metode također vrijedi ,ako je prvi link prečac na neku drugu datoteku i follow_symlinks je false, neće doći do kopiranja već će se stvoriti nova datoteka tj.prečac. Ako je follow_symlinks true i src je prečac dst će biti kopija datoteke na koju src upućuje.

Kod kopiranja ovom metodom, način dozvole će se kopirati, a ostali meta podaci kao što su vrijeme kreiranja i modificiranja neće.

```
401 def copy(src, dst, *, follow_symlinks=True):
402     """Copy data and mode bits ("cp src dst"). Return the file's destination.
403
404     The destination may be a directory.
405
406     If follow_symlinks is false, symlinks won't be followed. This
407     resembles GNU's "cp -P src dst".
408
409     If source and destination are the same file, a SameFileError will be
410     raised.
411
412     """
413     if os.path.isdir(dst):
414         dst = os.path.join(dst, os.path.basename(src))
415     copyfile(src, dst, follow_symlinks=follow_symlinks)
416     copymode(src, dst, follow_symlinks=follow_symlinks)
417     return dst
```

Slika 7 - Algoritam copy() funkcije. Literatura: [8]

Kod algoritma, prvo se provjerava je li određena putanja direktorij. Ako je, njoj pridružujemo ime datoteke izvora iz kojeg se kopira. Zatim koristimo prethodno objašnjenu metodu za kopiranje i metodu copymode() kako bi ostvarili kopiranje načina dozvole. Dohvaćamo podatke datoteke koju kopiramo i zatim određenoj datoteci pridružujemo zaštitne bitove.

3.4. Funkcija `copy2(src, dst, *, follow_symlinks=True)` i algoritam

Metoda `copy2` je gotovo identična metodi `copy`. Razlika je u tome što ova metoda nastoji kopirati sve meta podatke. Kada je `follow_symlinks` `false` i `src` je prečac, nastoje se kopirati svi meta podaci iz `src` prečaca u novi `dst` prečac. Ova funkcionalnost ne radi na svim platformama.

```
419 def copy2(src, dst, *, follow_symlinks=True):
420     """Copy data and metadata. Return the file's destination.
421
422     Metadata is copied with copystat(). Please see the copystat function
423     for more information.
424
425     The destination may be a directory.
426
427     If follow_symlinks is false, symlinks won't be followed. This
428     resembles GNU's "cp -P src dst".
429     """
430     if os.path.isdir(dst):
431         dst = os.path.join(dst, os.path.basename(src))
432     copyfile(src, dst, follow_symlinks=follow_symlinks)
433     copystat(src, dst, follow_symlinks=follow_symlinks)
434     return dst
```

Slika 8 - Algoritam `copy2()` funkcije. Literatura: [8]

Kod je gotovo identičan prethodnoj metodi uz male razlike. Vidimo kako se u ovoj metodi kopiranja koristi `copystat()` metoda za meta podatke. Njena svrha je kao što je već navedeno kopirati bitove dopuštenja, vrijeme zadnjeg pristupa dokumentu, izmjene itd.

4. Osvrt na laboratorijsku vježbu 1

U laboratorijskoj vježbi jedan smo kopirali datoteke koristeći različite algoritme kopiranja na Windows OS-u. Ovim seminarom, pokušat ćemo se približiti laboratorijskoj vježbi koristeći shutil modul za kopiranje datoteka te usporediti vrijeme izvođenja.




4.1. Kreiranje datoteka

Radi jednostavnosti, za kreiranje datoteka koristili smo metodu `truncate(size)`. Navedena metoda koristi se za skraćivanje datoteke na zadanu veličinu ukoliko je ona predana. U našem slučaju poslužit ćemo se navedenom metodom kako bi kreirali datoteku odgovarajuće veličine.

```
#size 1024*1024 = 1MB
size = 1024*1024
with open("file1MB.txt", "wb") as file:
    file.truncate(size)
```

Slika 9 – Kod za kreiranje datoteka određenih veličina.

Navedeni kod koristi se za kreiranje datoteke veličine 1MB. Za kreiranje datoteka preostalih veličina navedenu varijablu `size` množimo dodatno sa 1024 kako bi dobili datoteku veličine 1GB te sa $1024*5$ za datoteku veličine 5GB.

 file1GB.txt	5/25/2020 4:24 PM	Text Document	1,048,576 KB
 file1MB.txt	5/25/2020 4:33 PM	Text Document	1,024 KB
 file5GB.txt	5/25/2020 4:26 PM	Text Document	5,242,880 KB

Slika 10 – Prikaz kreiranih datoteka.

Gornjom slikom prikazane su datoteke odgovarajućih veličina, a koje su nama potrebne za izvođenje ove vježbe. Svaka novonastala, kopirana datoteka imati će ime kao i datoteka koju kopiramo uz nastavak „-copy“.

4.2. Kopiranje datoteka

Za kopiranje datoteka, koristiti ćemo metodu `copyfileobj`, budući da ona jedina ima mogućnost mijenjati veličinu spremnika. Prvo smo uključili potrebne module, tj. biblioteke. Uključili smo `Path` kako bi mogli provjeriti veličinu datoteke koju kopiramo i veličinu novonastale datoteke. Također, uključili smo `time` biblioteku, koja nam je potrebna kako bi mjerili vrijeme izvođenja kopiranja. Budući da već imamo kreirane datoteke odgovarajućih veličina, kod kopiranja datoteka mijenjat ćemo `source` i `destination` varijable te ćemo kod metode `copyfileobj` treći parametar mijenjat prema zadanim veličinama spremnika. Na slici ispod, nalazi se kod koji koristimo prilikom kopiranja datoteka.

```

import shutil
from pathlib import Path
import time

source = 'file1MB.txt'
fileSrc = open(source, 'r')
destination = 'file1MB-copy.txt'
fileDst = open(destination, 'w')

start = time.time()
shutil.copyfileobj(fileSrc, fileDst, 256)
end = time.time()
print("Spent time(buff-256): {} ms.".format((end-start)*1000))

fileSrc.close()
fileDst.close()
print("Size of source file: {} KB.".format(Path(source).stat().st_size/1024))
print("Size of destination file: {} KB.".format(Path(destination).stat().st_size/1024))

```

Slika 11 – Kod za kopiranje datoteka.

4.3. Rezultati izvođenja

Datoteka veličine 1MB

<pre> ===== RESTART: C:/Users/Kosco/De Spent time(buff-256): 29.891014099121094 ms. Size of source file: 1024.0 KB. Size of destination file: 1024.0 KB. </pre>	<pre> ===== RESTART: C:/Users/Kosco/Desk Spent time(buff-2048): 31.162261962890625 ms. Size of source file: 1024.0 KB. Size of destination file: 1024.0 KB. ... </pre>
<pre> ===== RESTART: C:/Users/Kosco/De Spent time(buff-8192): 39.93988037109375 ms. Size of source file: 1024.0 KB. Size of destination file: 1024.0 KB. >>> </pre>	

Datoteka veličine 1GB:

<pre> ===== RESTART: C:/Users/Kosco/D Spent time(buff-256): 19622.00379371643 ms. Size of source file: 1048576.0 KB. Size of destination file: 1048576.0 KB. </pre>	<pre> ===== RESTART: C:/Users/Kosco/De Spent time(buff-2048): 16520.34306526184 ms. Size of source file: 1048576.0 KB. Size of destination file: 1048576.0 KB. ... </pre>
<pre> ===== RESTART: C:/Users/Kosco/Des Spent time(buff-8192): 14296.761989593506 ms. Size of source file: 1048576.0 KB. Size of destination file: 1048576.0 KB. </pre>	

Datoteka veličine 5GB:

<pre> ===== RESTART: C:/Users/Kosco/De Spent time(buff-256): 183670.18342018127 ms. Size of source file: 5242880.0 KB. Size of destination file: 5242880.0 KB. >>> </pre>	<pre> ===== RESTART: C:/Users/Kosco/Des Spent time(buff-2056): 142418.45083236694 ms. Size of source file: 5242880.0 KB. Size of destination file: 5242880.0 KB. ... </pre>
<pre> ===== RESTART: C:/Users/Kosco/Des Spent time(buff-8192): 140418.6191558838 ms. Size of source file: 5242880.0 KB. Size of destination file: 5242880.0 KB. </pre>	

Tablica rezultata laboratorijske vježbe:

Windows		Veličina datoteke		
Program	BUF_SIZE	1MB	1GB	5GB
UW1	256	596	92108	311718
	2048	594	47704	227359
	8192	547	52144	225705
UW2	256	54	36427	131847
	2048	319	25964	132202
	8192	82	27099	127706
W1	256	108	72480	292044
	2048	120	26689	136960
	8192	269	28484	132362
W2	256	29	25636	113671
	2048	194	24799	127207
	8192	191	23359	117682
W3	256	450	62679	333442
	2048	605	34706	183652
	8192	515	36122	170308

Tablica rezultata koristeći Python i shutil:

Windows		Veličina datoteke		
Metoda	BUF_SIZE	1MB	1GB	5GB
copyfileobj	256	29.891	19622.003	183680.183
	2048	31.162	16520.343	142418.450
	8192	39.939	14296.761	140418.619

5. Zaključak

Prvenstveno, vrijeme izvođenja dosta ovisi o memorijskom spremniku, veličini datoteka, ali i o specifikacijama osobnog računala. Budući da su laboratorijske vježbe i ovaj pokus izvođene na istom računalu, možemo reći da odstupanja nema. Ukoliko promotrimo tablice, vidimo kako shutil-ova metoda copyfileobj malo odstupa od programa W2 koji je u laboratorijskoj vježbi bio najbolji po vremenskom izvođenju u odnosu na ostale programe. Vidimo kako se za datoteku veličine 1MB povećanjem spremnika, povećavalo i trajanje kopiranja, ali i dalje je brže u odnosu na program W2. Kod datoteke veličine 1GB, naš pokus i dalje ima najbolji rezultat u odnosu na ostale programe. Kod većih datoteka, možemo uočiti kako veličina spremnika utječe na trajanje kopiranja. Što je spremnik veći, to je trajanje kopiranja manje. Zadnja datoteka je imala veličinu 5GB. W2 Program u odnosu na naš pokus ima znatno brže izvođenje. Vrijeme trajanja kopiranja u našem slučaju se također ponašalo kao i kod datoteke veličine 1GB odnosno povećanjem spremnika, trajanje se smanjilo. Možemo reći kako je za datoteke male veličine naš program nešto bolji u odnosu na ostale, koje smo koristili u prvoj laboratorijskoj vježbi. Mogli bi zaključiti kako nekakvu granicu do koje se naš program brže izvodi predstavlja 4GB, odnosno velike datoteke. Nakon te veličine, program znatno usporava, ali se i dalje nalazi u nekakvoj sredini između programa sa laboratorijske vježbe.

6. Literatura:

- [1] <https://hr.wikipedia.org/wiki/Datoteka>
- [2] https://en.wikipedia.org/wiki/Computer_file
- [3] <https://www.python.org/about/>
- [4] [https://hr.wikipedia.org/wiki/Python_\(programski_jezik\)](https://hr.wikipedia.org/wiki/Python_(programski_jezik))
- [5] <https://wiki.python.org/moin/>
- [6] slika - <https://qph.fs.quoracdn.net/main-qimg-b4d8dfe1de991041fdaf15df8549de93-c>
- [7] <https://docs.python.org/3/library/shutil.html>
- [8] Shutil modul(source kod) - <https://github.com/python/cpython/blob/3.8/Lib/shutil.py>
- [9] MemoryView - <https://www.programiz.com/python-programming/methods/built-in/memoryview>