**Hapat që të arrish te skripti dhe ta ekzekutosh me time:**

1. **Hap terminalin në Ubuntu/WSL**  
   (jo PowerShell ose CMD – duhet të jesh në mjedisin Linux të WSL)
2. **Navigo te folderi me skriptin:**

bash

CopyEdit

cd /mnt/c/Users/Amorii7.AMORII7/Desktop/master\_task

1. **Kontrollo që skripti ndodhet aty:**

bash

CopyEdit

ls

Duhet të shfaqet edhe process\_file.py.

1. **Ekzekuto dhe mat kohën:**

bash

CopyEdit

time python3 process\_file.py

**Si ta interpretoj këtë rezultat:**

* Execution Time: 0.00579... seconds – kjo është koha që ti ke matur brenda vetë skriptit (time.time() në Python).
* real 0m0.059s – koha e përgjithshme që sistemi mori për të përfunduar komandën (përfshin gjithçka, edhe kohën që kalon në pritje, I/O, etj.).
* user 0m0.036s – koha që CPU shpenzoi në ekzekutimin e kodit të Python-it.
* sys 0m0.012s – koha që CPU kaloi në operacione të sistemit (p.sh. hapje skedari, lexime, etj.).

Kur ti ekzekuton komandën top në terminalin e **Ubuntu/WSL**, do të shfaqet një pamje **e gjallë (live)** e përdorimit të burimeve të sistemit në kohë reale.

### ✅ Ja çfarë do të shohësh në top:

📌 **Seksionet kryesore në ekranin e top:**

#### 1. **Lart: Përmbledhje e sistemit**

yaml

CopyEdit

top - 15:43:01 up 1:23, 1 user, load average: 0.11, 0.05, 0.01

Tasks: 123 total, 1 running, 122 sleeping, 0 stopped, 0 zombie

%Cpu(s): 2.3 us, 0.7 sy, 0.0 ni, 96.5 id, 0.3 wa, 0.0 hi, 0.2 si, 0.0 st

MiB Mem : 7987.6 total, 2123.7 free, 1452.9 used, 4411.0 buff/cache

MiB Swap: 2048.0 total, 2048.0 free, 0.0 used. 5879.0 avail Mem

* **%Cpu(s)** – përqindja e përdorimit të CPU:
  + us = user space (kod Python, etj.)
  + sy = system (operacione të sistemit)
  + id = idle (kohë e papërdorur, e lirë)
* **Mem / Swap** – përdorimi i RAM dhe hapësira e rezervuar për RAM të tejkaluar.

#### 2. **Poshtë: Lista e proceseve**

yaml

CopyEdit

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND

1532 testuser 20 0 23412 4120 3088 R 98.3 0.1 0:03.45 python3

* **%CPU** – tregon se sa CPU po përdor një proces (p.sh. python3)
* **%MEM** – tregon sa RAM po përdor
* **COMMAND** – emri i procesit (mund të jetë python3, bash, etj.)

### ⚠️ Si ta ndalosh top?

Shtyp **q** për ta mbyllur.

### 💡 Si ta përdorësh për matje?

1. **Run top**
2. Në një dritare tjetër, run python3 process\_file.py
3. Shiko në top se si rritet përdorimi i CPU dhe RAM gjatë ekzekutimit.

Nëse dëshiron, mund të përdorim edhe **htop** (një version vizual më i bukur). Mund ta instalosh me:

bash

CopyEdit

sudo apt update

sudo apt install htop

Pastaj e ekzekuton me:

bash

CopyEdit

htop

Më thuaj nëse do ndihmë ta interpretojmë një rezultat konkret që ke parë nga top.

4o

##### You said:

top - 01:27:36 up 22 min, 1 user, load average: 0.00, 0.00, 0.00 Tasks: 26 total, 1 running, 25 sleeping, 0 stopped, 0 zombie %Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 99.9 id, 0.1 wa, 0.0 hi, 0.0 si, 0.0 st MiB Mem : 3842.5 total, 3288.0 free, 585.2 used, 166.9 buff/cache MiB Swap: 1024.0 total, 1024.0 free, 0.0 used. 3257.2 avail Mem PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND 55 root 19 -1 66840 19704 18584 S 0.3 0.5 0:01.24 systemd-journal 1 root 20 0 21664 12944 9672 S 0.0 0.3 0:02.47 systemd 2 root 20 0 2476 1436 1320 S 0.0 0.0 0:00.05 init-systemd(Ub 7 root 20 0 2492 132 132 S 0.0 0.0 0:00.00 init 103 root 20 0 24492 6636 4848 S 0.0 0.2 0:01.29 systemd-udevd 157 systemd+ 20 0 21452 12072 9876 S 0.0 0.3 0:00.47 systemd-resolve 158 systemd+ 20 0 91020 6532 5684 S 0.0 0.2 0:00.67 systemd-timesyn 170 root 20 0 4236 2652 2408 S 0.0 0.1 0:00.06 cron 171 message+ 20 0 9576 5148 4552 S 0.0 0.1 0:00.19 dbus-daemon 189 root 20 0 17976 8256 7236 S 0.0 0.2 0:00.27 systemd-logind 196 root 20 0 1756096 16096 9500 S 0.0 0.4 0:00.33 wsl-pro-service 201 root 20 0 3160 1232 1144 S 0.0 0.0 0:00.01 agetty 206 root 20 0 3116 1080 992 S 0.0 0.0 0:00.01 agetty 209 syslog 20 0 222508 7076 4500 S 0.0 0.2 0:00.17 rsyslogd 240 root 20 0 107008 22868 13036 S 0.0 0.6 0:00.24 unattended-upgr 290 root 20 0 2492 116 0 S 0.0 0.0 0:00.00 SessionLeader 291 root 20 0 2492 124 0 S 0.0 0.0 0:00.00 Relay(292) 292 test 20 0 6068 5348 3620 S 0.0 0.1 0:00.22 bash 293 root 20 0 6820 4596 3812 S 0.0 0.1 0:00.01 login 337 test 20 0 20056 11252 9364 S 0.0 0.3 0:00.51 systemd 353 test 20 0 21144 1724 0 S 0.0 0.0 0:00.00 (sd-pam) 378 test 20 0 6056 4940 3436 S 0.0 0.1 0:00.03 bash 409 root 20 0 2492 120 0 S 0.0 0.0 0:00.00 SessionLeader 410 root 20 0 2492 128 0 S 0.0 0.0 0:00.13 Relay(411) 411 test 20 0 6068 5340 3620 S 0.0 0.1 0:00.44 bash 471 test 20 0 9276 5340 3196 R 0.0 0.1 0:00.19 top

###### ChatGPT said:

Faleminderit për këtë rezultat — është shumë i qartë dhe i dobishëm për analizë.

### ✅ Çka tregon ky rezultat nga komanda top?

#### 🔹 **1. Gjendja e përgjithshme e sistemit:**

scss

CopyEdit

%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 99.9 id, 0.1 wa, 0.0 hi, 0.0 si, 0.0 st

* 99.9 id ➜ CPU pothuajse është **totalisht e lirë** (idle).
* 0.0 us ➜ Nuk ka aktivitete nga përdoruesi (si skripta python).
* Do të thotë: **asnjë proces nuk po e ngarkon sistemin në atë moment.**

#### 🔹 **2. Përdorimi i RAM:**

yaml

CopyEdit

MiB Mem : 3842.5 total, 3288.0 free, 585.2 used

* Pothuajse **85% e RAM-it është e lirë**, që është **shumë mirë**.
* Asnjë proces nuk po konsumon RAM në mënyrë të dukshme.

#### 🔹 **3. Proceset aktuale:**

bash

CopyEdit

PID USER %CPU %MEM COMMAND

471 test 0.0 0.1 top

292 test 0.0 0.1 bash

...

* **Procesi top vetë** është aktiv (PID 471), por po përdor 0.0% CPU.
* **Nuk ka procese python3** që do të thotë se skripta process\_file.py nuk ishte duke u ekzekutuar në këtë moment.

### ✅ A është kjo normale?

Po, është **shumë normale** nëse:

* Ke **ekzekutuar skriptin më herët**, dhe ai është përfunduar shpejt.
* Skripta ka qenë **shumë e lehtë**, kështu që përfundohet për pak milisekonda dhe nuk shihet më në top.

### 🧪 Si ta testosh ndikimin real të një skripte në top?

Nëse dëshiron të shohësh ndikimin real në CPU dhe RAM:

#### 1. Redakto skriptën process\_file.py që të zgjasë pak më shumë:

python

CopyEdit

import time

print("Processing...")

time.sleep(5) # qëndron 5 sekonda aktiv

print("Done")

#### 2. Në një terminal, ekzekuto top

#### 3. Në një tjetër terminal, ekzekuto:

bash

CopyEdit

python3 process\_file.py

#### 4. Në top, do të shfaqet një rresht me COMMAND = python3 dhe %CPU do jetë më i lartë për disa sekonda.

Nëse dëshiron, mund ta konfigurojmë skriptën që të përdorë më shumë CPU qëllimisht për testim.

Dëshiron një shembull për ta testuar më mirë ndikimin e CPU

Mund të krijosh një skriptë që i ngarkon CPU për 30 sekonda:

python

CopyEdit

import time

print("Running CPU intensive task for 30 seconds...")

start\_time = time.time()

while time.time() - start\_time < 30:

# Bëj një llogaritje të thjeshtë intensive për CPU

x = 0

for i in range(10\_000\_000):

x += i\*i

print("Done.")

Ruaje këtë si cpu\_load.py dhe ekzekuto me:

bash

CopyEdit

python3 cpu\_load.py

Ndërkohë, në një terminal tjetër hap top dhe kërko për procesin python3, duhet të shohësh përdorim të lartë CPU.

Nëse sërish nuk shikon përdorim, më trego:

* Saktësisht përmbajtjen e skriptës që po përdor (cpu\_load.py),
* Nëse je në WSL ose Linux natyral,
* Si e ekzekuton skriptën.

Kështu të ndihmoj më mirë.

4.1-mini

##### You said:

dmth une duhet ti hap dy dritare te ngjashme ubuntu edhe ne njeren duhet te beje run time dhe ne tjetren top?

###### ChatGPT said:

Po, saktë!

Në një dritare terminali (ose tab) ekzekuton skriptën Python që bën ngarkesë CPU (p.sh. python3 cpu\_load.py),  
dhe në një dritare tjetër terminali hap top për të monitoruar përdorimin e CPU në kohë reale.

Kështu, ndërsa skripta po punon, ti sheh në top se sa CPU po përdoret.

Nëse i ke të dy në të njëjtin terminal, nuk mund të shohësh ngarkesën e CPU në kohë reale bashkë me skriptën, prandaj duhen dy dritare (ose tab) terminali.

Nëse je në Windows dhe po përdor WSL, mund të hapësh dy dritare Ubuntu (ose dy tab-e në terminalin që përdor) dhe të bësh këtë eksperimente.

Në rregull?

4.1-mini

##### You said:

test@Amorii7:~$ cd /mnt/c/Users/Amorii7.AMORII7/Desktop/master\_task test@Amorii7:/mnt/c/Users/Amorii7.AMORII7/Desktop/master\_task$ top top - 01:45:54 up 38 min, 1 user, load average: 0.44, 0.17, 0.06 Tasks: 31 total, 2 running, 29 sleeping, 0 stopped, 0 zombie %Cpu(s): 16.7 us, 0.0 sy, 0.0 ni, 83.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st MiB Mem : 3842.5 total, 3276.9 free, 597.3 used, 165.0 buff/cache MiB Swap: 1024.0 total, 1024.0 free, 0.0 used. 3245.2 avail Mem PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND 543 test 20 0 15332 10848 6212 R 100.0 0.3 0:11.53 python3 471 test 20 0 9276 5340 3196 S 0.3 0.1 0:00.72 top 1 root 20 0 21664 12944 9672 S 0.0 0.3 0:02.53 systemd 2 root 20 0 2476 1436 1320 S 0.0 0.0 0:00.16 init-systemd(Ub 7 root 20 0 2492 132 132 S 0.0 0.0 0:00.00 init 55 root 19 -1 66840 19744 18624 S 0.0 0.5 0:01.36 systemd-journal 103 root 20 0 24492 6636 4848 S 0.0 0.2 0:01.53 systemd-udevd 157 systemd+ 20 0 21452 12072 9876 S 0.0 0.3 0:00.56 systemd-resolve 158 systemd+ 20 0 91020 6532 5684 S 0.0 0.2 0:00.70 systemd-timesyn 170 root 20 0 4236 2652 2408 S 0.0 0.1 0:00.06 cron 171 message+ 20 0 9576 5148 4552 S 0.0 0.1 0:00.23 dbus-daemon 189 root 20 0 17976 8256 7236 S 0.0 0.2 0:00.38 systemd-logind 196 root 20 0 1756096 16096 9500 S 0.0 0.4 0:00.49 wsl-pro-service 201 root 20 0 3160 1232 1144 S 0.0 0.0 0:00.01 agetty 206 root 20 0 3116 1080 992 S 0.0 0.0 0:00.01 agetty 209 syslog 20 0 222508 7076 4500 S 0.0 0.2 0:00.19 rsyslogd 240 root 20 0 107008 22868 13036 S 0.0 0.6 0:00.24 unattended-upgr 293 root 20 0 6820 4596 3812 S 0.0 0.1 0:00.01 login 337 test 20 0 20056 11252 9364 S 0.0 0.3 0:00.51 systemd 353 test 20 0 21144 1724 0 S 0.0 0.0 0:00.00 (sd-pam) 378 test 20 0 6056 4940 3436 S 0.0 0.1 0:00.03 bash 409 root 20 0 2492 120 0 S 0.0 0.0 0:00.00 SessionLeader

###### ChatGPT said:

Perfekt, kjo tregon se gjithçka po funksionon siç duhet! ✅

Nga output-i i komandës top, shohim këtë rresht:

bash

CopyEdit

543 test ... R 100.0 0.3 0:11.53 python3

Kjo do të thotë se skripta jote Python po përdor **100% të një CPU-je**, që është pikërisht ajo që prisnim. Ti ke:

* Hapur skriptën që krijon ngarkesë CPU
* Dhe po e monitoron atë me top

Për të përmbledhur:

* PID 543 është procesi i Python që po ekzekuton skriptën tënde (cpu\_load.py)
* %CPU = 100.0 tregon se po përdor maksimalisht një bërthamë të CPU-së

do të përgatis një përmbledhje të saktë për përdorimin e curl në skriptin tënd cpu\_load.py brenda Ubuntu/WSL. Kjo do të përfshijë si ta ekzekutosh skriptin për të gjeneruar ngarkesë me curl, si ta monitorosh me top/htop, dhe si të mbledhësh të dhëna për analizë në temën tënde të masterit. Do të të njoftoj sapo të jetë gati për rishikim.

Research cancelled

##### You said:

a mund te me ndihmosh?

###### ChatGPT said:

Po, patjetër që mund të të ndihmoj! Ja udhëzimi i plotë për të përdorur një skriptë Python (cpu\_load.py) në Ubuntu/WSL që gjeneron ngarkesë në CPU duke bërë kërkesa HTTP me curl, dhe për të monitoruar këtë me top ose htop.

## ✅ **1. Krijimi i skriptës** cpu\_load.py **që përdor** curl

Kjo skriptë do të përdorë multiprocessing për të krijuar disa procese që ekzekutojnë curl në një URL të caktuar.

python

CopyEdit

import sys

import subprocess

from multiprocessing import Process

def stress\_with\_curl(url: str):

while True:

subprocess.run(["curl", "-s", url], stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL)

if \_\_name\_\_ == "\_\_main\_\_":

if len(sys.argv) < 2:

print("Përdorimi: python3 cpu\_load.py <URL> [numri\_i\_proceseve]")

sys.exit(1)

url = sys.argv[1]

num\_processes = int(sys.argv[2]) if len(sys.argv) > 2 else 4 # default 4 procese

processes = []

for \_ in range(num\_processes):

p = Process(target=stress\_with\_curl, args=(url,))

p.start()

processes.append(p)

for p in processes:

p.join()

## ✅ **2. Si ta ekzekutosh në Ubuntu/WSL**

1. **Shko në dosjen ku është skripta:**

bash

CopyEdit

cd /mnt/c/Users/Amorii7.AMORII7/Desktop/master\_task

1. **Ekzekutoje skriptën me një URL aktive:**

bash

CopyEdit

python3 cpu\_load.py http://localhost:3000 4

Kjo do të gjenerojë trafik ndaj localhost:3000 me 4 procese.

## ✅ **3. Hap një terminal tjetër dhe monitoro me** top **ose** htop

### Me top:

bash

CopyEdit

top

Shiko për proceset python3 që përdorin shumë CPU.

### Me htop:

bash

CopyEdit

htop

Në htop është më vizual — mund të përdorësh F6 për të sortuar sipas %CPU.

## ✅ **4. Si ta ndalosh skriptën**

Në terminalin ku e ke lëshuar cpu\_load.py, shtyp:

objectivec

CopyEdit

CTRL + C

## ✅ **5. Këshilla për testim të saktë për temën tënde të masterit**

* Provo të ndryshosh numrin e proceseve: 2, 4, 8, 16 për të parë efektin në load average.
* Mund të përdorësh endpoint-e të thjeshta që japin përgjigje të menjëhershme (ping API).
* Monitoro si ndikohet përdorimi i CPU-së dhe shëno të dhënat për krahasime midis ambienteve të ndryshme.

Nëse dëshiron, mund të përfshij edhe:

* Ruajtje automatike e rezultateve në një file .csv.
* Version me asyncio në vend të multiprocessing.
* Apo mënyrën si mund të bësh testime të ngjashme në cloud.

Më thuaj si dëshiron të vazhdojmë 😊

4o

##### You said:

test@Amorii7:/mnt/c/Users/Amorii7.AMORII7/Desktop/master\_task$ htop internal error, please report: running "htop" failed: cannot find installed snap "htop" at revision 4853: missing file /snap/htop/4853/meta/snap.yaml test@Amorii7:/mnt/c/Users/Amorii7.AMORII7/Desktop/master\_task$

###### ChatGPT said:

Ky gabim ndodh sepse htop është instaluar përmes **snap**, por instalimi është i dëmtuar ose i paplotë. Ja si mund ta rregullosh këtë problem në WSL (Ubuntu):

### ✅ ****Zgjidhja më e shpejtë: Çinstalo**** htop ****dhe instaloje me**** apt

bash

CopyEdit

sudo snap remove htop

sudo apt update

sudo apt install htop

Kjo do ta instalojë versionin klasik të htop që nuk varet nga Snap, dhe funksionon mirë në WSL.

### 🟡 ****Nëse snap është problem në WSL (që ndodh shpesh)****

Mund të shmangësh Snap totalisht dhe të përdorësh vetëm apt.

Nëse snap jep gjithmonë gabime, mund ta çaktivizosh apo heqësh fare, por në rastin tënd, instalimi i htop me apt është zgjidhje perfekte për të gjitha nevojat e monitorimit të ngarkesës në CPU.

### ✅ ****Pasi të instalohet suksesshëm**** htop

Thjesht ekzekuto:

bash

CopyEdit

htop

Dhe do të shohësh pamje të ngjashme me këtë:

perl

CopyEdit

PID USER ... %CPU %MEM COMMAND

1234 test 100 0.5 python3 cpu\_load.py http://localhost:3000

...

a mund te me shkruash nje dokumentim te thjeshte ne kete pjese qe sapo e kompletuam, import argparse import csv import subprocess import time import psutil from multiprocessing import Pool def curl\_request(url): start = time.time() # Ekzekutoni curl dhe merrni kohën e përgjigjes result = subprocess.run( ["curl", "-o", "/dev/null", "-s", "-w", "%{time\_total}", url], capture\_output=True, text=True ) response\_time = float(result.stdout.strip()) cpu = psutil.cpu\_percent(interval=None) timestamp = time.strftime("%Y-%m-%d %H:%M:%S") return (timestamp, cpu, response\_time) def worker(args): url, filename = args data = curl\_request(url) with open(filename, "a", newline="") as f: writer = csv.writer(f) writer.writerow(data) return data def main(): parser = argparse.ArgumentParser(description="Load test with curl and CPU usage logging") parser.add\_argument("--url", required=True, help="URL to test") parser.add\_argument("--processes", type=int, default=4, help="Number of parallel processes") parser.add\_argument("--file", default="results.csv", help="CSV file to save results") args = parser.parse\_args() # Shto header në csv (vetëm në fillim) with open(args.file, "w", newline="") as f: writer = csv.writer(f) writer.writerow(["timestamp", "cpu\_percent", "response\_time\_sec"]) pool = Pool(args.processes) print(f"Starting load test on {args.url} with {args.processes} processes. Saving to {args.file}") try: while True: pool.map(worker, [(args.url, args.file)] \* args.processes) time.sleep(1) # Pushim 1 sekondë mes batch-eve except KeyboardInterrupt: print("Test stopped by user.") if \_\_name\_\_ == "\_\_main\_\_": main()

###### ChatGPT said:

Sigurisht! Ja një dokumentim i thjeshtë në gjuhën shqipe për kodin që sapo ke kompletuar:

# Dokumentacion i Skriptit për Testim Ngarkese dhe Matjen e CPU-së

Ky skript Python kryen testim ngarkese në një URL të dhënë duke përdorur curl, dhe njëkohësisht regjistron përdorimin e CPU-së dhe kohën e përgjigjes së serverit në një skedar CSV.

## Funksionaliteti Kryesor

* **curl\_request(url)**  
  Ekzekuton komandën curl për të marrë kohën totale të përgjigjes nga URL-ja e dhënë. Gjithashtu, lexon përdorimin aktual të CPU-së në sistem dhe shtyp kohën e ekzekutimit.
* **worker(args)**  
  Funksion që thërret curl\_request dhe shkruan rezultatin në një skedar CSV. Merr URL-në dhe emrin e skedarit si argument.
* **main()**  
  Përgjigjet për marrjen e argumenteve nga komandat (URL, numri i proceseve, dhe emri i skedarit të rezultatit). Krijon një Pool me procese paralelë që ekzekutojnë testin vazhdimisht me interval 1 sekondë, derisa përdoruesi të ndalojë ekzekutimin (CTRL+C).

## Si përdoret

bash

CopyEdit

python3 cpu\_load2.py --url http://localhost:3000 --processes 4 --file results.csv

* --url: URL-ja që do të testohet (e detyrueshme)
* --processes: Numri i proceseve paralelë që do të ekzekutohen (default: 4)
* --file: Emri i skedarit CSV ku ruhen rezultatet (default: results.csv)

## Output

Skedari CSV përmban kolonat:

* timestamp: Koha kur është regjistruar testi (formati YYYY-MM-DD HH:MM:SS)
* cpu\_percent: Përqindja e përdorimit të CPU-së në momentin e testit
* response\_time\_sec: Koha në sekonda që mori përgjigja nga serveri përmes curl

## Vërejtje

* Skripti ekzekutohet në mënyrë të vazhdueshme derisa përdoruesi të ndërpresë me CTRL+C.
* Është i nevojshëm që curl dhe paketa psutil të jenë të instaluara në sistem.
* Testi kryhet paralelisht me numrin e proceseve që specifikohet për të simuluar ngarkesë.

2025-05-18 02:58:31,0.7,0.000141 2025-05-18 02:58:31,0.7,0.000167 2025-05-18 02:58:31,0.7,0.000157

Ky është një fragment të dhënash në format CSV që përmban tre kolona:

* **timestamp** — data dhe ora e matjes (p.sh., "2025-05-18 02:57:35")
* **cpu\_percent** — përqindja e përdorimit të CPU-së në atë moment (p.sh., 18.2%)
* **response\_time\_sec** — koha e përgjigjes në sekonda (p.sh., 0.001532 sekonda)

Çdo rresht tregon një regjistrim të veçantë me kohën, sa CPU ishte në përdorim dhe sa shpejt erdhi përgjigja për atë moment. Të dhënat duken si të regjistruara në intervale shumë të shpejta, për të monitoruar performancën në kohë reale.