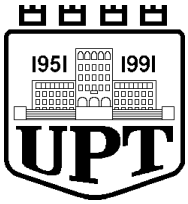


UNIVERSITETI POLITEKNIK I TIRANËS
FAKULTETI I TEKNOLOGJISË DHE INFORMACIONIT
DEPARTAMENTI I INXHINIERISË INFORMATIKE

Punë Laboratori nr. 2

Lënda: Sisteme Operative

Grupi: III-B

Tema: Ambjentimi dhe përdorimi i tools-ave në *Linux* për të kuptuar koncepte të menaxhimit të proceseve dhe memorjes.

Punoi:
Piro Gjikdhima

Pranoi:
MSc. Megi Tartari

Pjesa e parë: Përdorimi i tools-it ***free*** për të parë përdorimin/konsumin e memorjes në sistem

a) Futu në terminal dhe shtyp komandën ***$free -m*** për të parë memorjen e zënë dhe të lirë në sistem e shprehur në MB.

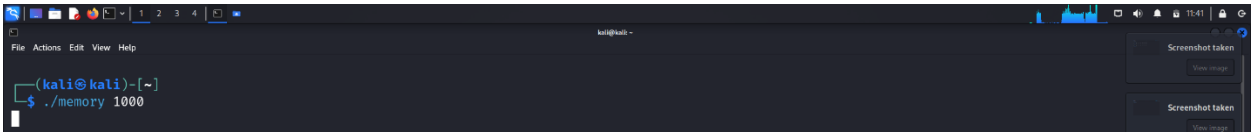
Në memorje kemi në total 3922 MB ku janë të përdorura 829 MB dhe të lira 2874 MB. Simboli *swap* i shënuar më poshtë është memorja virtuale, e cila përdoret kur memorja RAM është full.

b) Krijo në gjuhën C një program "*memory_use.c*" që alokon një array me përmasa të dhënë si argument në MB. Gjatë ekzekutimit programi duhet të skanojë këtë array duke aksesuar çdo element të saj në një loop infinit.

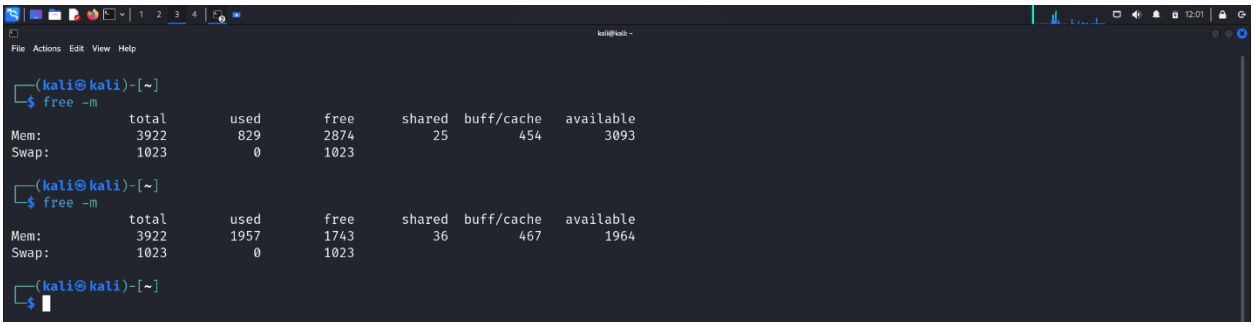
Kodi i shkruajtur në C:

Pasim marrim argumentat nga përdoruesi, bëjmë alokimin në memorje me anë të funksionit *malloc()*. Krijojmë ciklin e pafund me *while(1)* për aksesimin e çdo elementi të këtij vektori. Në përfundit të ciklit çlirojmë memorje, gjë kjo që nuk arrihet kurrë.

Tashmë le të ekzekutojmë kodin:



c) Logohu në një terminal tjetër dhe jep komandën ***$free -m*** për ta parë sa ndryshon përdorimi i memorjes gjatë kohës që ekzekutohet programi "*memory_use.c*".



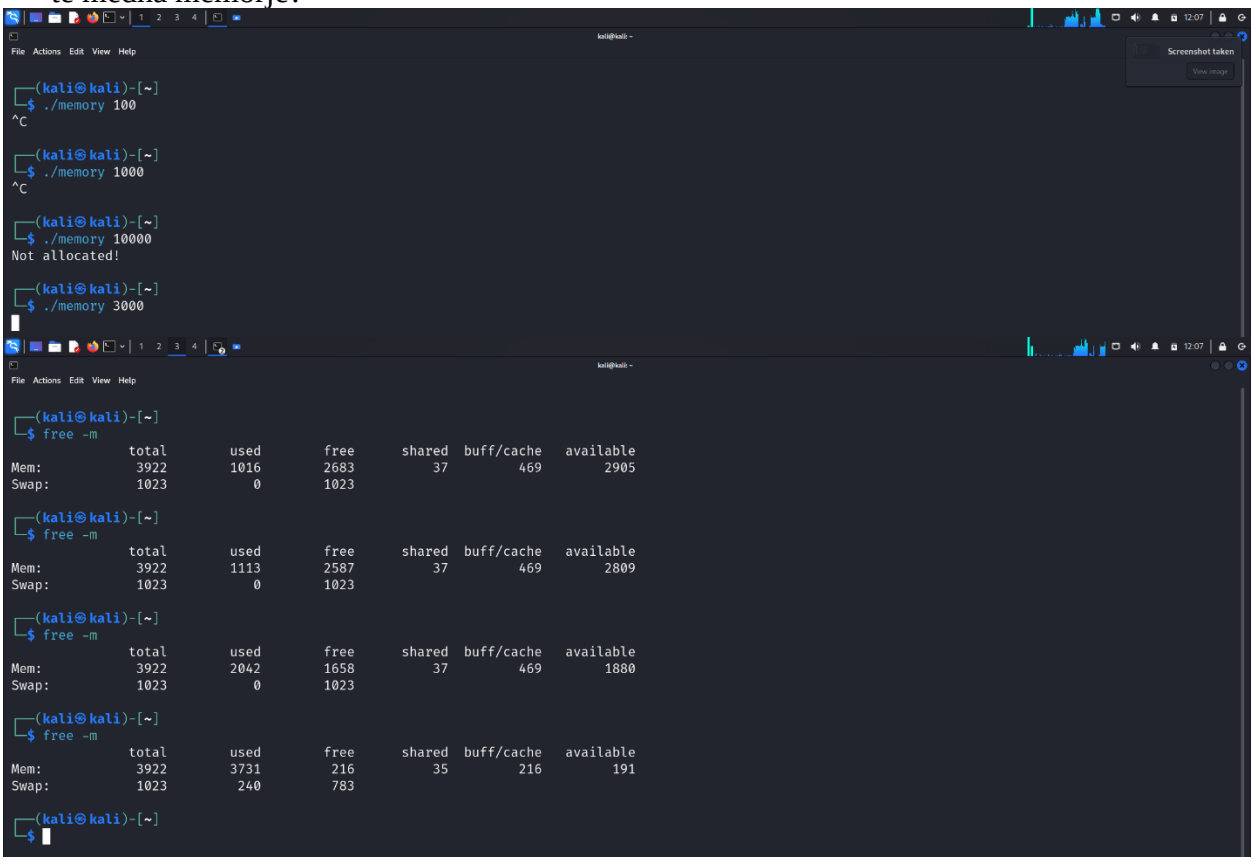
Shohim që nga 829 MB të zëna, tashtë janë bërë 1957 MB.

d) Si ndryshon përdorimi i memorjes nqs e përfundon programin me komandën ***kill***?

Gjejmë si fillim id-në e procesit në CPU me anë të komandës ***pgrep*** dhe i japim ***kill***.

Shohim që memorja u krye në gjendjen e mëparshme dhe *cache*-ja po lirohet.

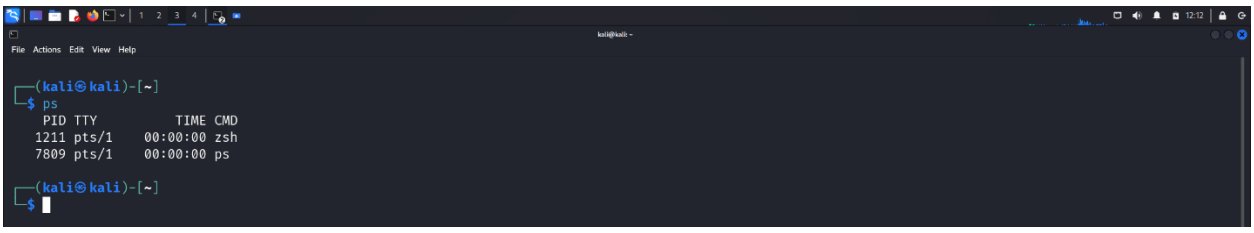
e) Provo hapat **c** dhe **d** për përmasa të ndryshme të array. Çfarë ndodh nqs programi alokon sasi të mëdha memorje?



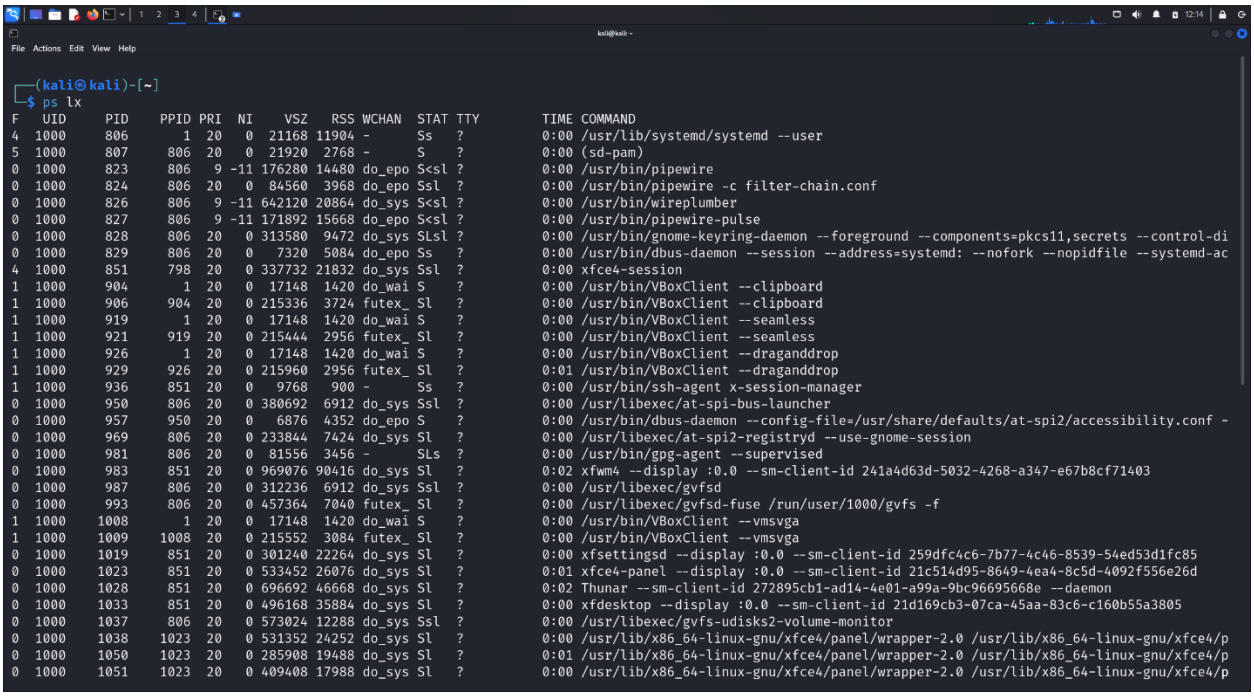
Për vlera të vogla, memorja nuk ndikohet shumë. Për vlera të mëdha fillon e zihet memorja, kurse për vlera si 10 000 MB si në rastin e parafundit kemi failure në alokimin e memorjes. Në rastin e fundit kemi kërkuar 3000 MB memorie dhe nje pjese e saj mbulohet nga Swap.

Pjesa e dytë: Përdorimi i tools-ave për menaxhimin e proceseve

a) Përdor komandat ***ps***, ***ps lx***, ***pstree***, ***ps -aux*** për të shfaqur informacione mbi proceset.

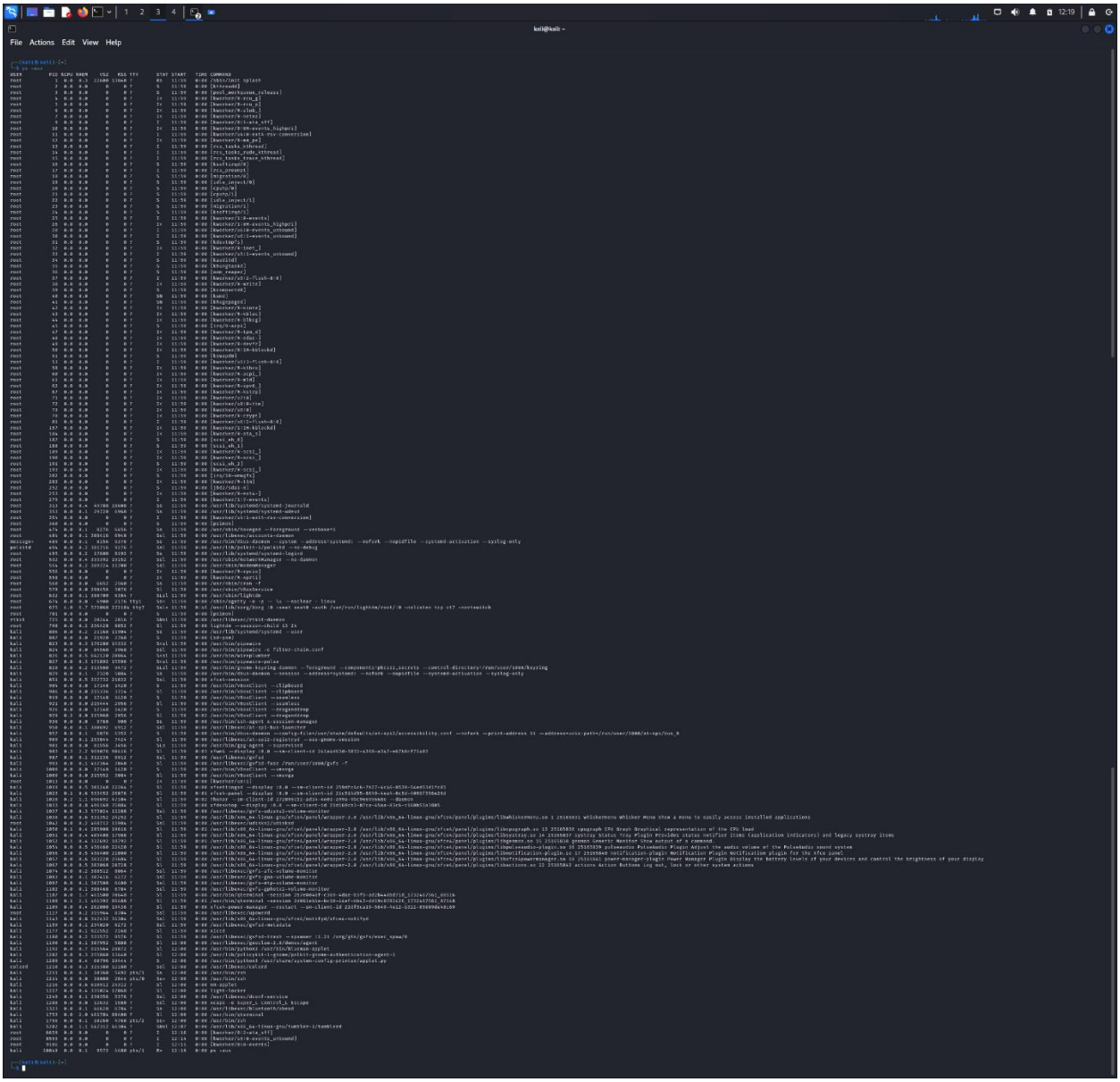


Komanda ***ps*** (process status) shërben për të treguar procesin aktual që është në atë moment në procesor.



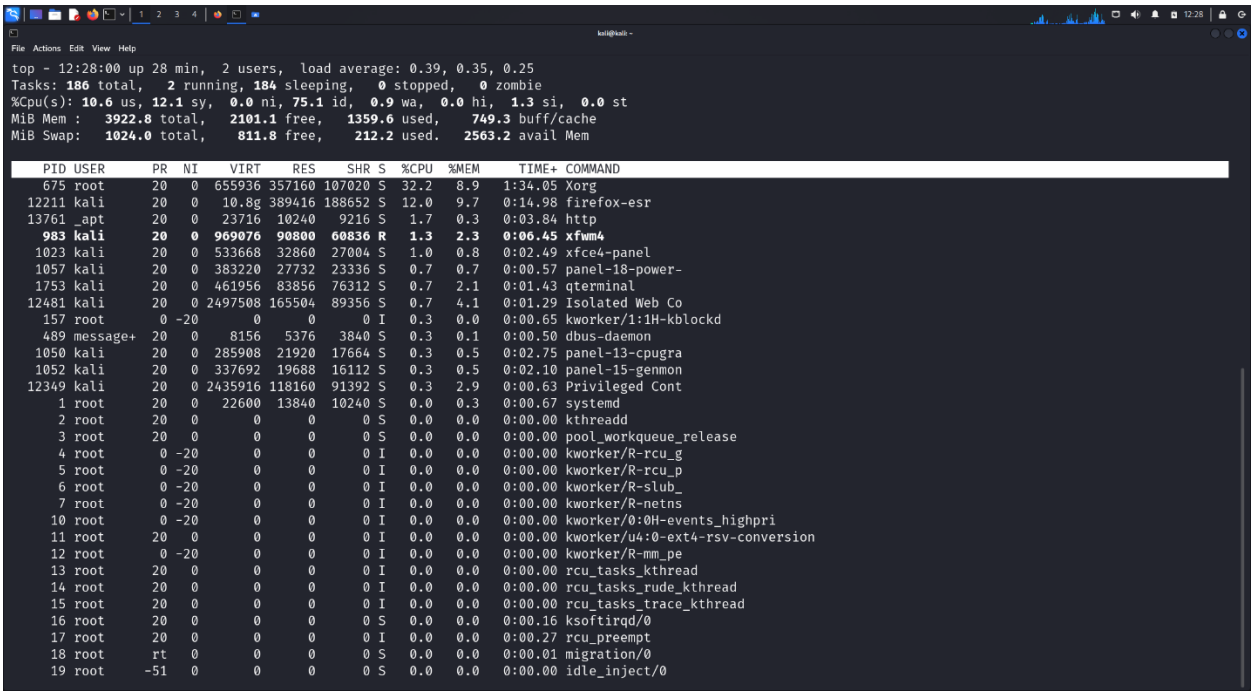
Komanda ***ps lx*** shërben për të shfaqur një listë të gjatë procesesh. *l* qëndron për longer dhe *x* qëndron për proceset që nuk kanë lidhje me terminalin.

Komanda **pstree** shfaq një strukturë të proceseve në një mënyrë hierarkike (si një pemë).



Komanda ***ps aux*** shërben për të shfaqur një listë të detajuar të të gjithë proceseve të sistemit. *a* qëndron për listën e proceseve të të gjithë përdoruesve, *u* qëndron për detaje më shumë të output-teve dhe *x* për procese që nuk kanë lidhje me terminalin.

b) Jep komadën ***top*** (shikoni dhe variantet e saj ***htop***, ***atop***, ***vtop***) ndërkohë që starton një web browser dhe shiko efektin tek shfaqja e komandës ***top***.



Në ***top*** tregohet si useri ka hapur *firefox* dhe sa % të memorjes po zë secili proces (afërsisht 9.7%).

- Sa memorje të lirë ka në sistem ?

Kemi 2101.1 MB të lira në memorje.

- Cili proces zë më shumë kohë CPU-je ?

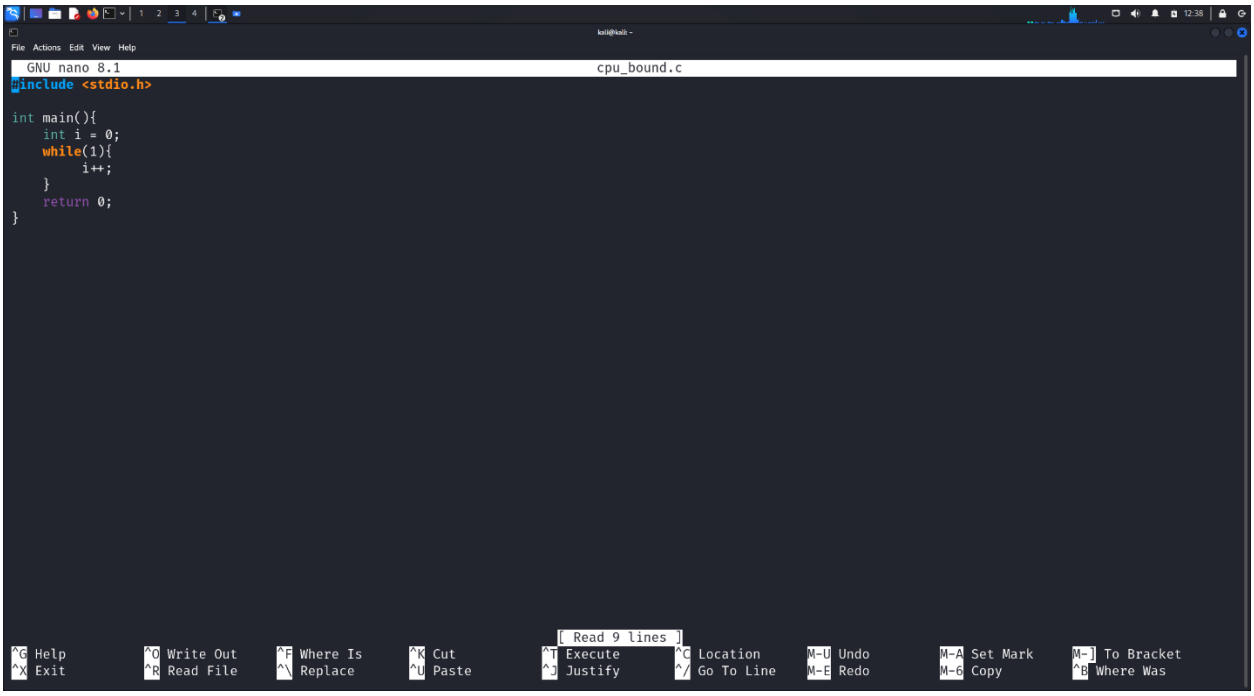
Në këtë rast procesi me id. 675 që është *Xorg*.

- Cili proces zë pjesën më të madhe të memorjes ?

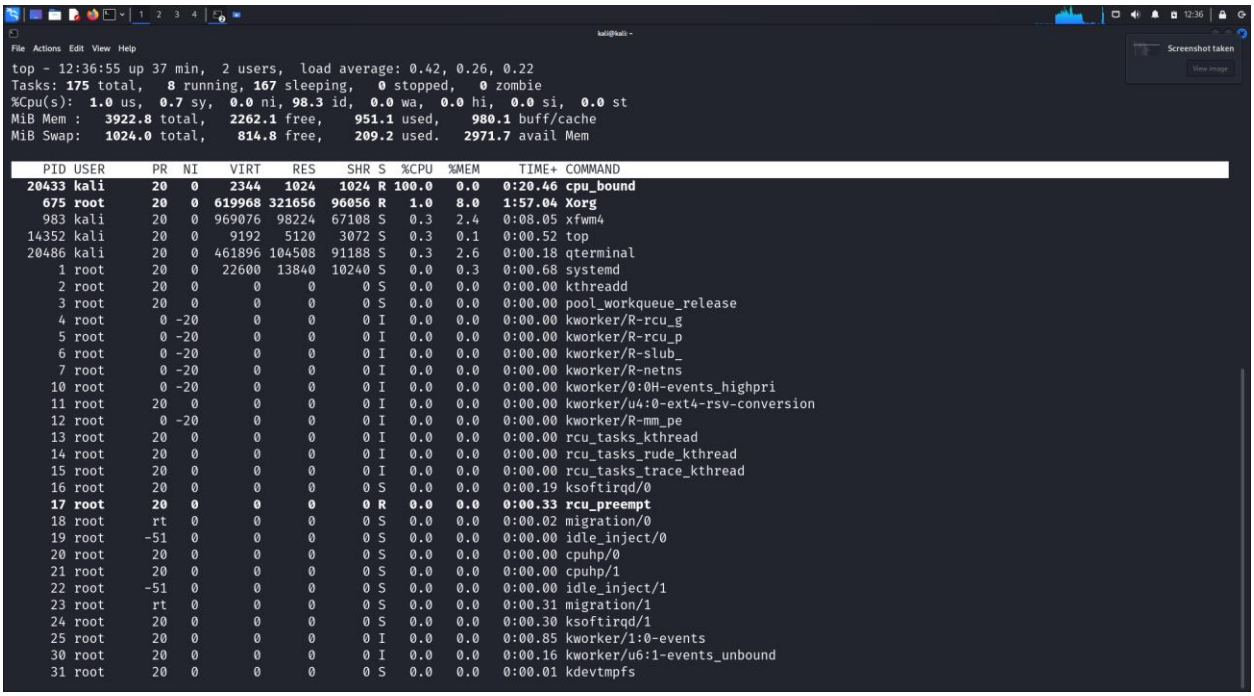
Është firefox që zë 9.7% dhe më pas Xorg me 8.9%.

c) Shkruaj dy programe në gjuhën C, një *CPU-bound* (printon diçka në ekran në një loop infinit) dhe nje *I/O bound* (ky i fundit të përdorë më shumë ***printf*** në një loop infinit). Ekzekutoji programet dhe shiko sa CPU përdorin nëpërmjet komadës ***top***.

File-i CPU-bound

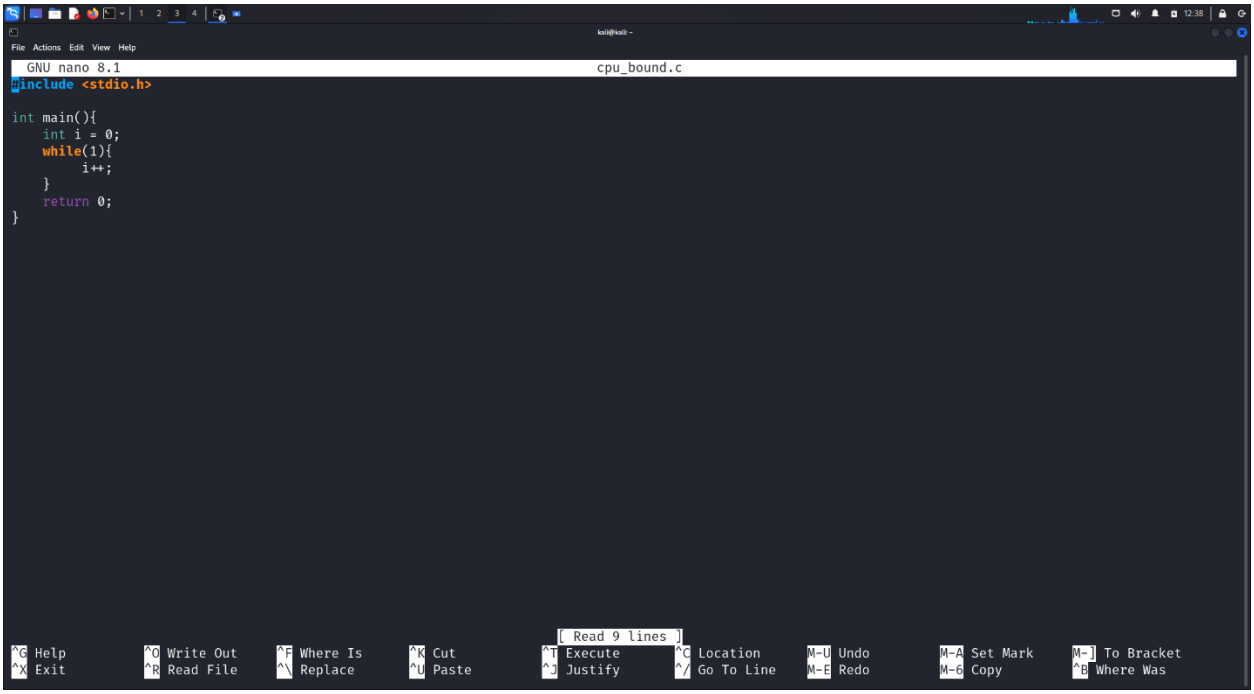


Ekzekutimi dhe vrojtimi në ***htop***:

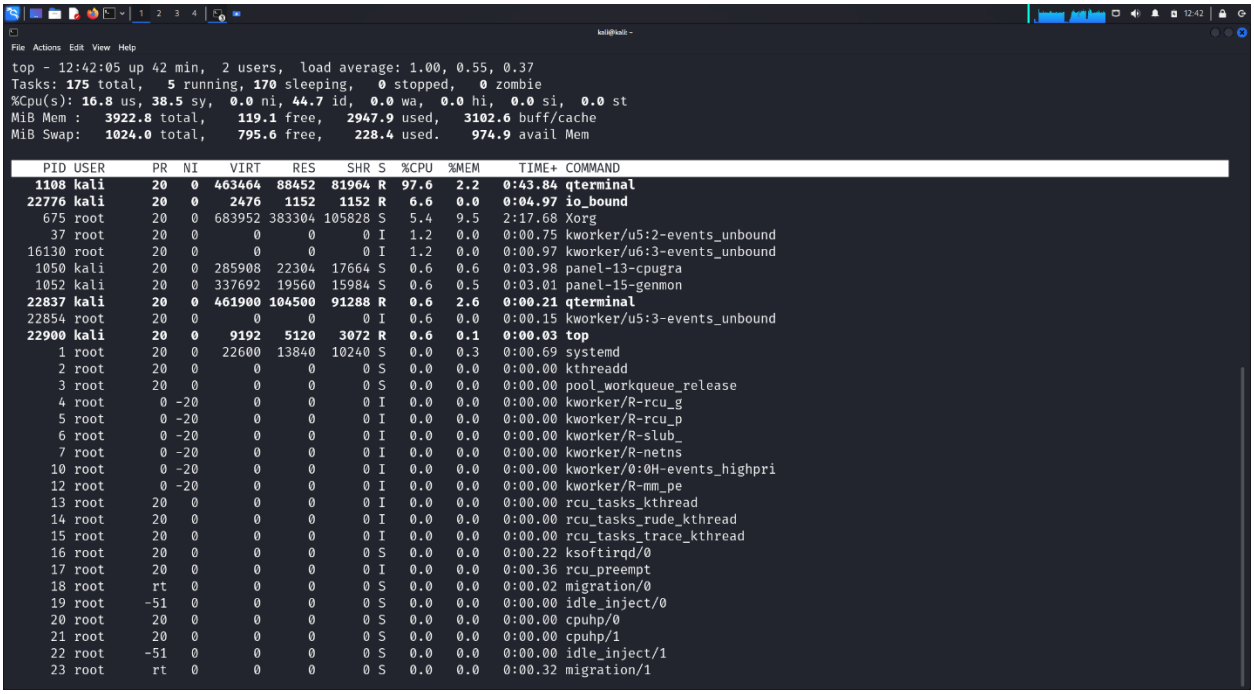


Del që ky program përdor 100% të CPU (e mbingarkon).

File-i i I/O-bound

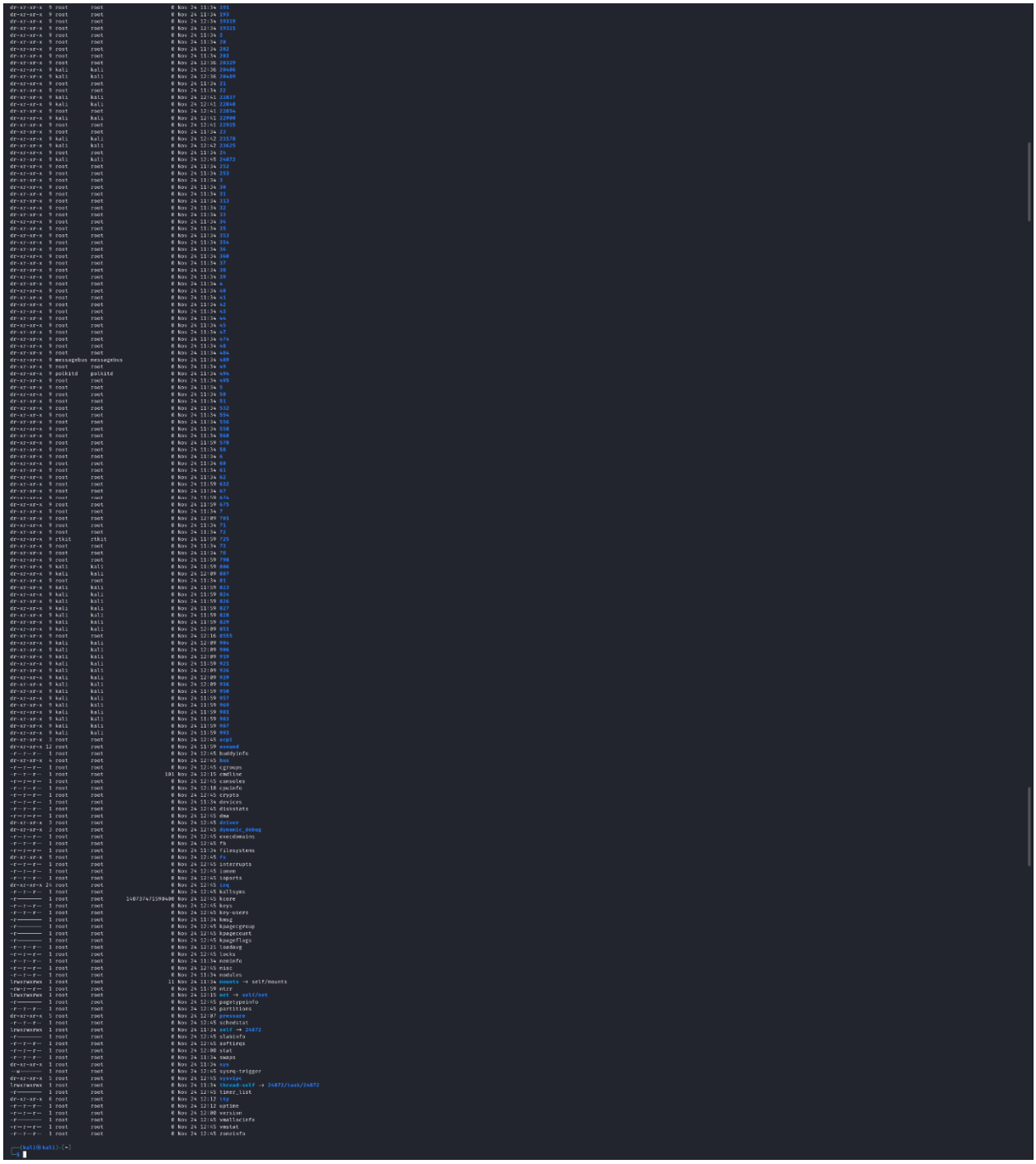


Ekzekutimi dhe vrojtimi në ***top***:

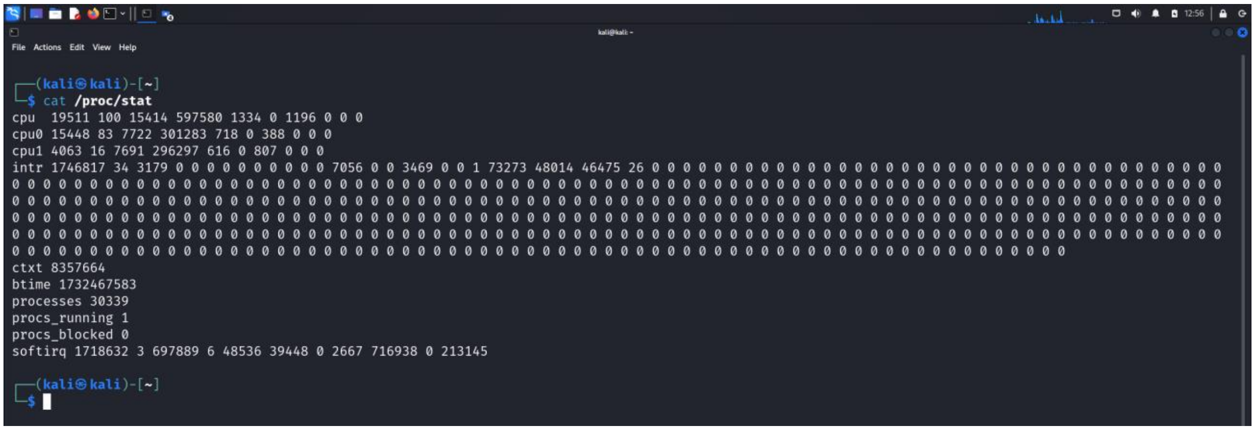


Vet si program I/O bound zë vetëm 6.6% të CPU-së. Por për faktin që detyra e këtij programi është të printoj pafundësisht në terminal, terminali tashmë zë 97.6% të CPU-së.

d) Shiko përmbajtjen e direktorisë ***/proc*** që është një ndërfaqe në formën e sistemit të file-ve në strukturat e të dhenave të *kernelit*. Shiko ***/proc/stat***.



Kjo është një pjesë e përmbajtjes së direktorisë */proc*. Kemi përdorur komandën ***ls -l*** për të parë të gjithë listën e direktorive dhe file-ve që ajo përmban. Kjo si direktori shërben për të mbajtur informacione rreth *kernel data structures* dhe për sistemin tonë. Kemi detaje për proceset që janë running dhe për id-të e tyre, kemi parametrat e kernelit në momentin që ne japim komandën në kohë reale siç tregohet dhe në datat dhe orët e treguara në figurë, memorjen e CPU-së (*meminfo* file), konfigurimet e sistemit tonë, na jep mundësi të bëjmë ndyshime në sistemin tonë duke modifikuar data structures që përmban sistemi.



```
┌──(kali㉿kali)-[~]
└─$ cat /proc/stat
cpu  19511 100 15414 597580 1334 0 1196 0 0 0
cpu0 15448 83 7722 301283 718 0 388 0 0 0
cpu1 4063 16 7691 296297 616 0 807 0 0 0
intr 1746817 34 3179 0 0 0 0 0 0 0 0 0 0 7056 0 0 3469 0 0 1 73273 48014 46475 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
ctxt 8357664
btime 1732467583
processes 30339
procs_running 1
procs_blocked 0
softirq 1718632 3 697889 6 48536 39448 0 2667 716938 0 213145

┌──(kali㉿kali)-[~]
└─$
```

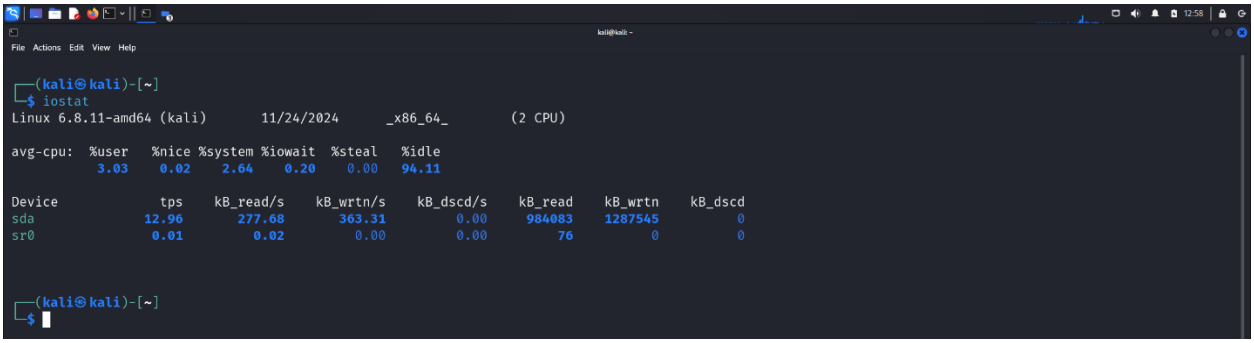
/proc/stat na jep të dhëna për CPU-në tonë si për kohën totale të përdorimit, kohën e përdorur në user mode, kernel/system mode, idle etj.. Ajo ekzekutohet duke përdorur komandën ***cat***. Kolonat në rrjeshtin ku lexon cpu përfshijnë kohët si user, nice, system, idle, iowait, irq, softirq etj.. kemi të dhënë numrin e proceseve totale që janë running (në këtë rast janë 30339), numrin e proceseve që janë running në terminal (në rastin tonë është 1) etj.

e) Përdor komandat ***df -h*** dhe ***iostat*** për të parë përdorimin e diskut. (Instalimi me ***apt-get install sysstat***)

```
┌──(kali㉿kali)-[~]
└─$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G     0  1.9G   0% /dev
tmpfs           393M  996K  392M   1% /run
/dev/sda1        79G   15G   60G  21% /
tmpfs           2.0G     0  2.0G   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           1.0M     0  1.0M   0% /run/credentials/systemd-journald.service
tmpfs           1.0M     0  1.0M   0% /run/credentials/systemd-udev-load-credentials.service
tmpfs           1.0M     0  1.0M   0% /run/credentials/systemd-sysctl.service
tmpfs           1.0M     0  1.0M   0% /run/credentials/systemd-tmpfiles-setup-dev-early.service
tmpfs           1.0M     0  1.0M   0% /run/credentials/systemd-tmpfiles-setup-dev.service
tmpfs           2.0G  552K  2.0G   1% /tmp
tmpfs           1.0M     0  1.0M   0% /run/credentials/systemd-tmpfiles-setup.service
kali            458G  296G  163G  65% /media/sf_kali
tmpfs           1.0M     0  1.0M   0% /run/credentials/getty@tty1.service
tmpfs           393M  124K  393M   1% /run/user/1000

┌──(kali㉿kali)-[~]
└─$
```

Komanda ***df -h*** na flet në një mënyrë më të lexueshme për njeriun për hapësirën e lirë të diskut tonë. Na jep numrin total të filesytem-eve tona, për madhësinë e secilit prej tyre, sa hapësirë është përdorur, hapësirën e lirë që ka ngelur për të dhënat e reja, përqindjen e hapësirës së zënë, dhe vendndodhjen e file-it në pemën e direktorisë.



Komanda ***iostat*** shërben për të raportuar përdorimin e CPU-së dhe statistikat e I/O të pajisjeve, partitions dhe sistemin e file-ve të rrjetit. Kjo komandë na ofron emrin e pajisjeve, the transaction për sekondë (tps), KB e lexuara për sekondë dhe ato të shkruara për sekondë, përqindjen e kohës së CPU-së ku shqyrtohen kërkesat e pajisjeve I/O. Kjo komandë na ndihmon për të optimizuar sistemin dhe për '*troubleshoot problems*'.

f) Përdor komandën ***pmap*** për të parë map-imin e memorjes së një procesi. Lexo faqen e manualit të ***pmap***. Jep komanden ***pmap*** me proces id-në e një procesi, psh: te *web browser*-it dhe parametra të ndryshëm për të parë informacione më të detajuara. Mund të dallosh zonën e kodit, stakut apo heap ? Mund ta përdorësh ***pmap*** me programin mëparshëm "*memory_use.c*" për alokime të ndryshme të memorjes. Çfarë vë re nga komanda ***pmap***?

Fillimisht përpara se të përdorim këtë komandë duhet të sigurohemi që po e përdorim si admonistrator me ***sudo*** përpara ose futemi në rrënjë me ***sudo su***. Gjithashtu le të gjejmë dhe id-në e procesit të *web browser*-it tonë me komandën ***pgrep***. Le dhe të përdorim ***-q*** për një format më konçiz.

```
00007f8f1a6b0000      4K r---- libmozsandbox.so
00007f8f1a6b1000      4K rw--- libmozsandbox.so
00007f8f1a6b2000     20K rw---   [ anon ]
00007f8f1a6b7000     16K r---- libgcc_s.so.1
00007f8f1a6bb000    140K r-x-- libgcc_s.so.1
00007f8f1a6de000     16K r---- libgcc_s.so.1
00007f8f1a6e2000      4K r---- libgcc_s.so.1
00007f8f1a6e3000      4K rw--- libgcc_s.so.1
00007f8f1a6e4000      4K r---- gtk30.mo
00007f8f1a6e5000      4K r---- libwayland-egl.so.1.22.0
00007f8f1a6e6000      4K r-x-- libwayland-egl.so.1.22.0
00007f8f1a6e7000      4K r---- libwayland-egl.so.1.22.0
00007f8f1a6e8000      4K r---- libwayland-egl.so.1.22.0
00007f8f1a6e9000      4K rw--- libwayland-egl.so.1.22.0
00007f8f1a6ea000      4K r---- libmozgtk.so
00007f8f1a6eb000      4K r-x-- libmozgtk.so
00007f8f1a6ec000      4K r---- libmozgtk.so
00007f8f1a6ed000      4K r---- libmozgtk.so
00007f8f1a6ee000      4K rw--- libmozgtk.so
00007f8f1a6ef000     12K r---- liblgpllibs.so
00007f8f1a6f2000     20K r-x-- liblgpllibs.so
00007f8f1a6f7000      8K r---- liblgpllibs.so
00007f8f1a6f9000      4K r---- liblgpllibs.so
00007f8f1a6fa000      4K rw--- liblgpllibs.so
00007f8f1a6fb000      8K rw---   [ anon ]
00007f8f1a6fd000      4K r---- ld-linux-x86-64.so.2
00007f8f1a6fe000    148K r-x-- ld-linux-x86-64.so.2
00007f8f1a723000     40K r---- ld-linux-x86-64.so.2
00007f8f1a72d000      8K r---- ld-linux-x86-64.so.2
00007f8f1a72f000      8K rw---- ld-linux-x86-64.so.2
00007ffe46069000    132K rw---   [ stack ]
00007ffe4608a000      4K rw---   [ anon ]
00007ffe460f2000     16K r----   [ anon ]
00007ffe460f6000      8K r-x--   [ anon ]

┌──(kali㉿kali)-[~]
└─$
```

Në këtë raport është e lehtë të dallojmë zonën e stack-ut, heap-it dhe të kodit. Zona e stakut zakonisht përcaktohet me *[stack]*. Zona e kodit është e përcaktuar me inicialet '*r-x*'(readable and executable). Kurse zona e heap-it përcaktohet me [heap], por në rastet që nuk e gjejmë dot, te elementet *[anon]* (anonymous) do të shohim ato që i kanë inicialet '*rx-*' (reand/write permission). Kjo ndodh sepse shpeshherë kto elemente të heap-it kanë madhësi të madhe.

Le ta përdorim këtë komandë për programin që krijuam më parë *memory_use.c*:

```
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~]
└─$ pgrep memory
35527

┌──(kali㉿kali)-[~]
└─$ sudo pmap -q 35527
[sudo] password for kali:
35527:   ./memory 1000
000055a92a10e000      4K r---- memory
000055a92a10f000      4K r-x-- memory
000055a92a110000      4K r---- memory
000055a92a111000      4K r---- memory
000055a92a112000      4K rw--- memory
000055a92a5cc000    132K rw---   [ anon ]
00007fb1bf600000 1024004K rw---   [ anon ]
00007fb1fde03000     12K rw---   [ anon ]
00007fb1fde06000    152K r---- libc.so.6
00007fb1fde2c000   1372K r-x-- libc.so.6
00007fb1fdf83000    340K r---- libc.so.6
00007fb1fdfd8000     16K r---- libc.so.6
00007fb1fdfdc000      8K rw---- libc.so.6
00007fb1fdfde000     52K rw---   [ anon ]
00007fb1fe002000      8K rw---   [ anon ]
00007fb1fe004000      4K r---- ld-linux-x86-64.so.2
00007fb1fe005000    148K r-x-- ld-linux-x86-64.so.2
00007fb1fe02a000     40K r---- ld-linux-x86-64.so.2
00007fb1fe034000      8K r---- ld-linux-x86-64.so.2
00007fb1fe036000      8K rw---- ld-linux-x86-64.so.2
00007ffd63dcc000    132K rw---   [ stack ]
00007ffd63df0000     16K r----   [ anon ]
00007ffd63df4000      8K r-x--   [ anon ]

┌──(kali㉿kali)-[~]
└─$
```

Nga komanda ***pmap*** vëmë re një informacion të detajuar të elementëve të një procesi, adresat e tyre në memorje, sa vend zënë, nëse kto elemente jan readable, writable dhe executable si dhe nëse janë elemente të kodit, stack-ut apo heap-it.