

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

## **Punë Laboratori nr. 2**

**Lënda:** Sisteme Operative

**Grupi:** III-B

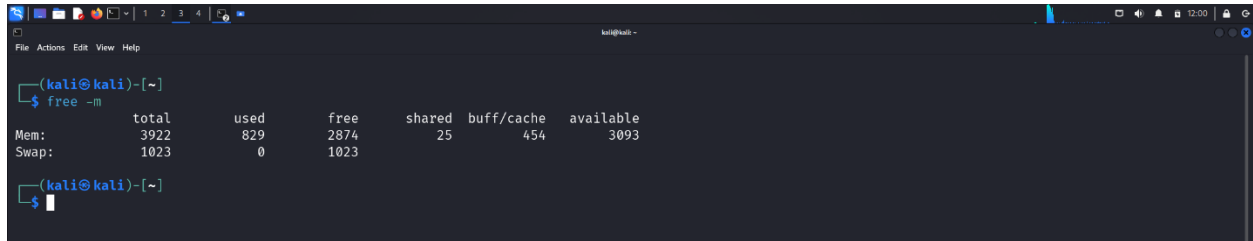
**Tema:** Ambjentimi dhe përdorimi i tools-ave në *Linux* për të kuptuar koncepte të menaxhimit të proceseve dhe memorjes.

**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.



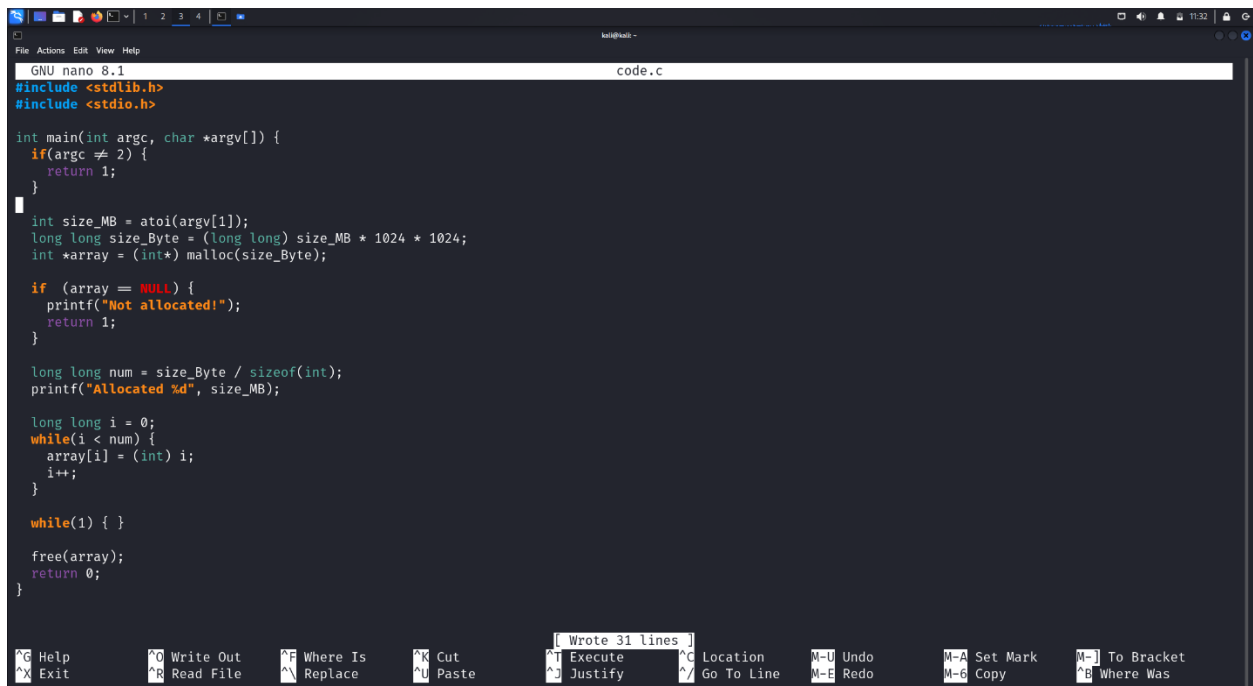
```
(kali@kali)-[~]
$ free -m
             total        used        free      shared  buff/cache   available
Mem:        3922         829        2874         25         454        3093
Swap:        1023           0         1023          0           0           0

(kali@kali)-[~]
$
```

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:*



```
GNU nano 8.1 code.c
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[]) {
    if(argc < 2) {
        return 1;
    }

    int size_MB = atoi(argv[1]);
    long long size_Byte = (long long) size_MB * 1024 * 1024;
    int *array = (int*) malloc(size_Byte);

    if (array == NULL) {
        printf("Not allocated!");
        return 1;
    }

    long long num = size_Byte / sizeof(int);
    printf("Allocated %d", size_MB);

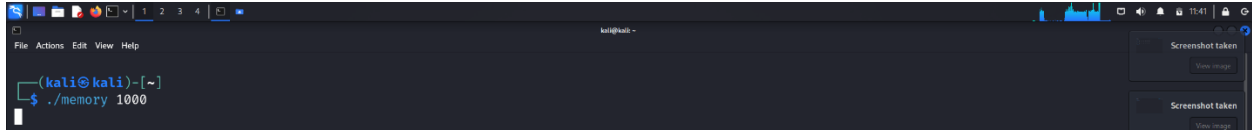
    long long i = 0;
    while(i < num) {
        array[i] = (int) i;
        i++;
    }

    while(1) { }

    free(array);
    return 0;
}
```

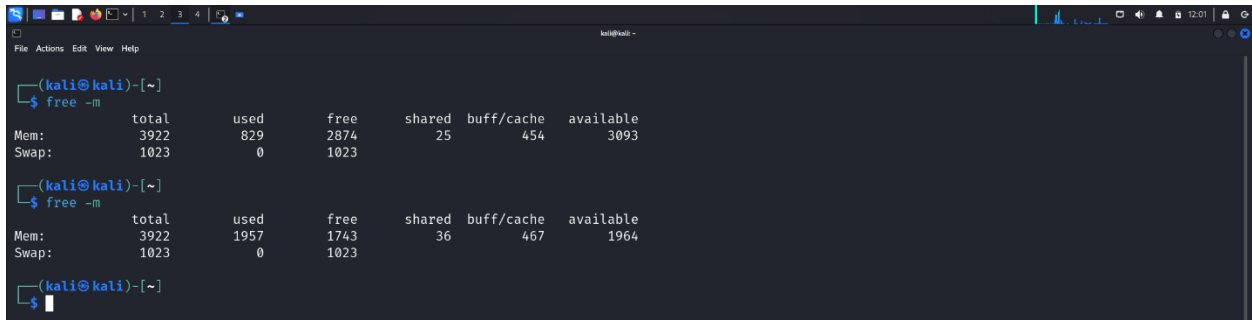
Pasim marrim argumentat nga përdoruesi, bëjmë alokimin në memorje me anë të funksionit *malloc()*. Krijojmë ciklin e pafund me *while(1)* për aksesimin e çdo elementi të këtij vektori. Në përfundit të ciklit çlirojmë memorje, gjë kjo që nuk arrihet kurrë.

Tashmë le të ekzekutojmë kodin:



```
(kali@kali)-[~]  
$ ./memory 1000
```

- 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*”.

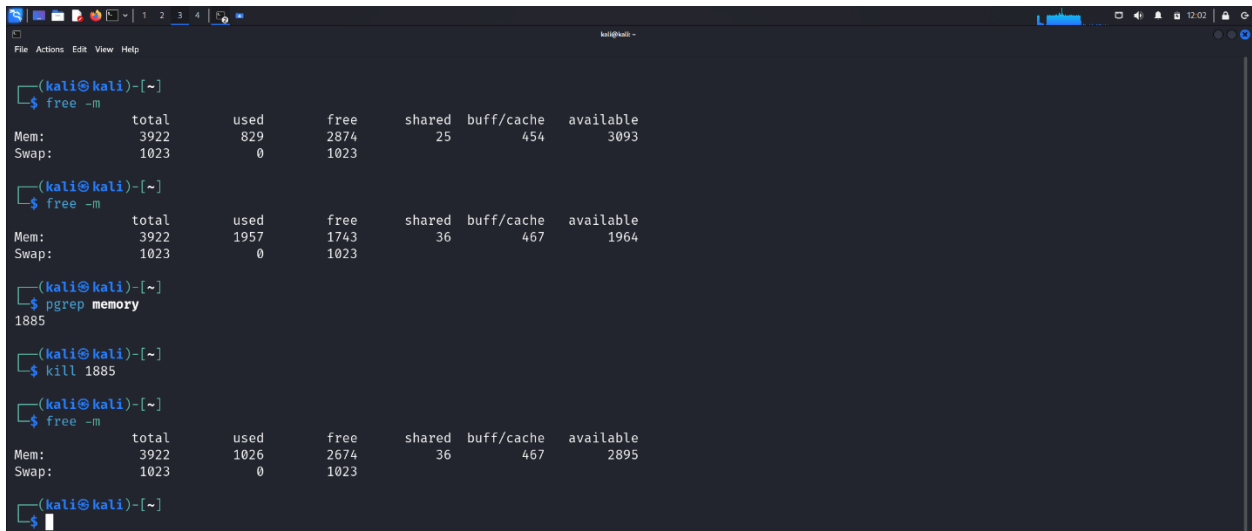


```
(kali@kali)-[~]  
$ free -m  
Mem:      total        used        free      shared  buff/cache   available  
Swap:      1023            0       1023           25         454         3093  
  
(kali@kali)-[~]  
$ free -m  
Mem:      total        used        free      shared  buff/cache   available  
Swap:      1023            0       1743           36         467         1964  
  
(kali@kali)-[~]  
$
```

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**?

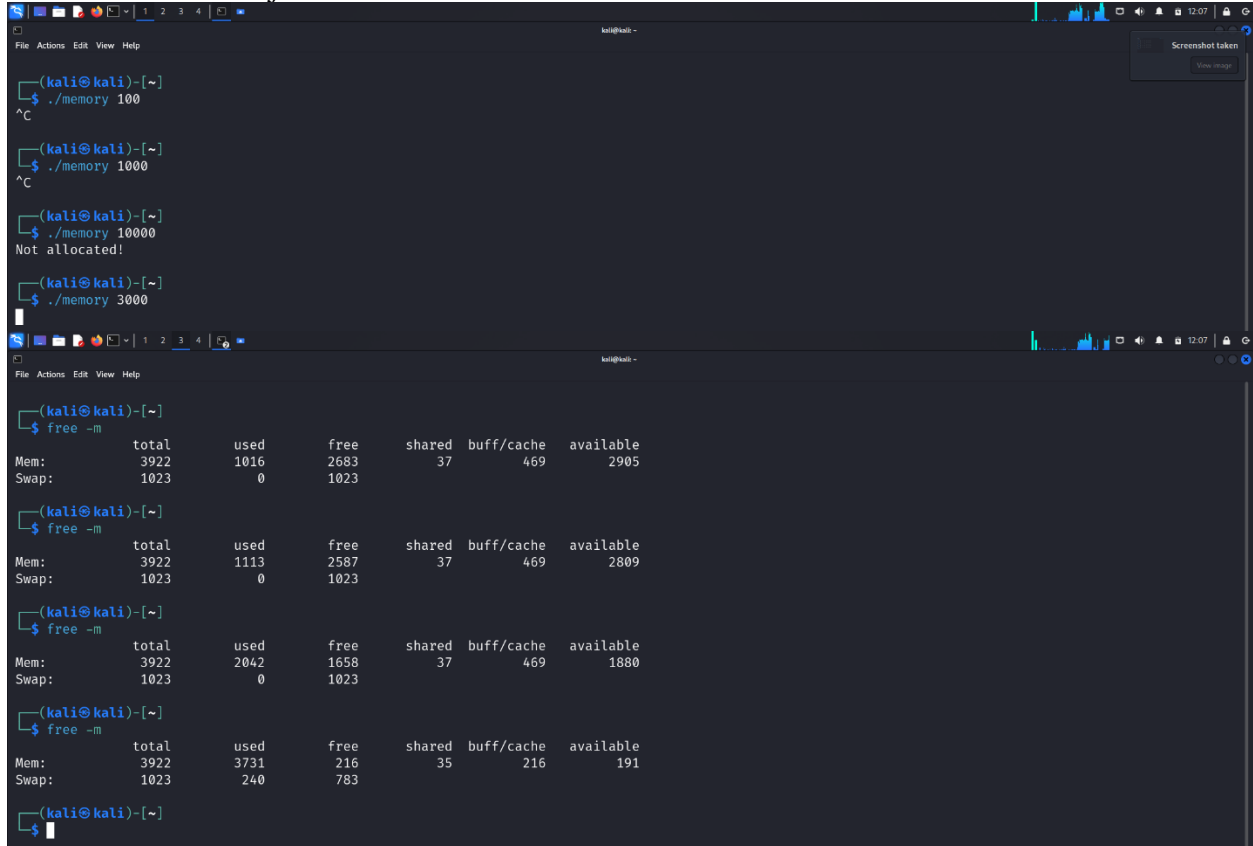
Gjejmë si fillim id-në e procesit në CPU me anë të komandës **pgrep** dhe i japim **kill**.



```
(kali@kali)-[~]  
$ free -m  
Mem:      total        used        free      shared  buff/cache   available  
Swap:      1023            0       2874           25         454         3093  
  
(kali@kali)-[~]  
$ free -m  
Mem:      total        used        free      shared  buff/cache   available  
Swap:      1023            0       1743           36         467         1964  
  
(kali@kali)-[~]  
$ pgrep memory  
1885  
  
(kali@kali)-[~]  
$ kill 1885  
  
(kali@kali)-[~]  
$ free -m  
Mem:      total        used        free      shared  buff/cache   available  
Swap:      1023            0       2674           36         467         2895  
  
(kali@kali)-[~]  
$
```

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?



```
(kali@kali)-[~]
$ ./memory 100
^C

(kali@kali)-[~]
$ ./memory 1000
^C

(kali@kali)-[~]
$ ./memory 10000
Not allocated!

(kali@kali)-[~]
$ ./memory 3000
```

```
(kali@kali)-[~]
$ free -m
total        used        free      shared  buff/cache   available
Mem:      3922        1016        2683         37         469        2905
Swap:      1023           0        1023

(kali@kali)-[~]
$ free -m
total        used        free      shared  buff/cache   available
Mem:      3922        1113        2587         37         469        2809
Swap:      1023           0        1023

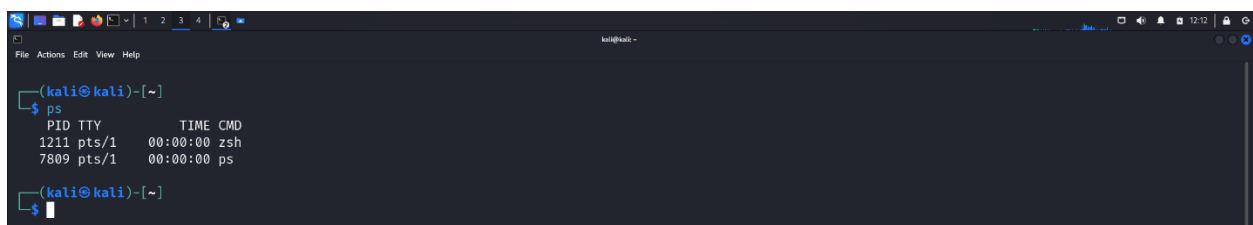
(kali@kali)-[~]
$ free -m
total        used        free      shared  buff/cache   available
Mem:      3922       2042        1658         37         469        1880
Swap:      1023           0        1023

(kali@kali)-[~]
$ free -m
total        used        free      shared  buff/cache   available
Mem:      3922       3731         216         35         216         191
Swap:      1023        240         783
```

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.



```
(kali@kali)-[~]
$ ps
PID TTY          TIME CMD
 1211 pts/1    00:00:00 zsh
 7809 pts/1    00:00:00 ps
```

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





```
top - 12:28:00 up 28 min, 2 users, load average: 0.39, 0.35, 0.25
Tasks: 186 total, 2 running, 184 sleeping, 0 stopped, 0 zombie
%Cpu(s): 10.6 us, 12.1 sy, 0.0 ni, 75.1 id, 0.9 wa, 0.0 hi, 1.3 si, 0.0 st
MiB Mem : 3922.8 total, 2101.1 free, 1359.6 used, 749.3 buff/cache
MiB Swap: 1024.0 total, 811.8 free, 212.2 used, 2563.2 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
  675 root        20   0 655936 357160 107020 S 32.2   8.9   1:34.05 Xorg
 12211 kali      20   0  10.8g 389416 188652 S 12.0   9.7   0:14.98 firefox-esr
 13761 _apt      20   0  23716 10240  9216 S  1.7   0.3   0:03.84 http
   983 kali     20   0 969076 90800 60836 R  1.3   2.3   0:06.45 xfwm4
  1023 kali     20   0 533668 32860 27004 S  1.0   0.8   0:02.49 xfce4-panel
  1057 kali     20   0 383220 27732 23336 S  0.7   0.7   0:00.57 panel-18-power-
  1753 kali     20   0 461956 83856 76312 S  0.7   2.1   0:01.43 qterminal
 12481 kali     20   0 2497508 165504 89356 S  0.7   4.1   0:01.29 Isolated Web Co
  157 root       0 -20   0      0      0 I  0.3   0.0   0:00.65 kworker/1:1H-kblockd
  489 message+ 20   0  8156  5376  3840 S  0.3   0.1   0:00.50 dbus-daemon
  1050 kali     20   0 285908 21920 17664 S  0.3   0.5   0:02.75 panel-13-cpugra
  1052 kali     20   0 337692 19688 16112 S  0.3   0.5   0:02.10 panel-15-genmon
 12349 kali     20   0 2435916 118160 91392 S  0.3   2.9   0:00.63 Privileged Cont
    1 root       20   0  22600 13840 10240 S  0.0   0.3   0:00.67 systemd
    2 root       20   0      0      0 S  0.0   0.0   0:00.00 kthreadd
    3 root       20   0      0      0 S  0.0   0.0   0:00.00 pool_workqueue_release
    4 root       0 -20   0      0      0 I  0.0   0.0   0:00.00 kworker/R-rcu_g
    5 root       0 -20   0      0      0 I  0.0   0.0   0:00.00 kworker/R-rcu_p
    6 root       0 -20   0      0      0 I  0.0   0.0   0:00.00 kworker/R-slub_
    7 root       0 -20   0      0      0 I  0.0   0.0   0:00.00 kworker/R-netns
   10 root       0 -20   0      0      0 I  0.0   0.0   0:00.00 kworker/0:0H-events_highpri
   11 root       20   0      0      0 I  0.0   0.0   0:00.00 kworker/u4:0-ext4-rsv-conversion
   12 root       0 -20   0      0      0 I  0.0   0.0   0:00.00 kworker/R-mm_pe
   13 root       20   0      0      0 I  0.0   0.0   0:00.00 rcu_tasks_kthread
   14 root       20   0      0      0 I  0.0   0.0   0:00.00 rcu_tasks_rude_kthread
   15 root       20   0      0      0 I  0.0   0.0   0:00.00 rcu_tasks_trace_kthread
   16 root       20   0      0      0 S  0.0   0.0   0:00.16 ksoftirqd/0
   17 root       20   0      0      0 I  0.0   0.0   0:00.27 rcu_preempt
   18 root       rt   0      0      0 S  0.0   0.0   0:00.01 migration/0
   19 root      -51   0      0      0 S  0.0   0.0   0:00.00 idle_inject/0
```

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 një *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***.

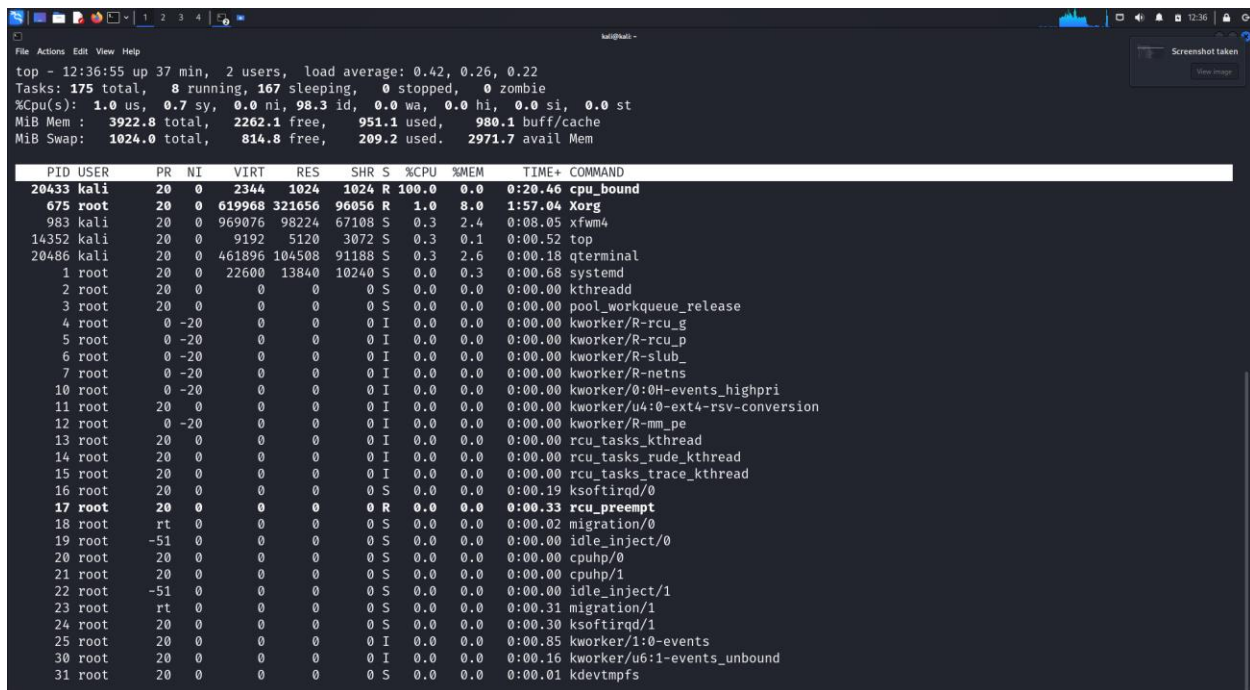
*File-i CPU-bound*



```
GNU nano 8.1 cpu_bound.c
#include <stdio.h>

int main(){
    int i = 0;
    while(1){
        i++;
    }
    return 0;
}
```

Ekzekutimi dhe vrojtimi në *htop*:



```
top - 12:36:55 up 37 min, 2 users, load average: 0.42, 0.26, 0.22
Tasks: 175 total, 8 running, 167 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.0 us, 0.7 sy, 0.0 ni, 98.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 3922.8 total, 2262.1 free, 951.1 used, 980.1 buff/cache
MiB Swap: 1024.0 total, 814.8 free, 209.2 used, 2971.7 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
20433	kali	20	0	2344	1024	1024	R	100.0	0.0	0:20.46	cpu_bound
675	root	20	0	619968	321656	96056	R	1.0	8.0	1:57.04	Xorg
983	kali	20	0	969076	98224	67108	S	0.3	2.4	0:08.05	xfwm4
14352	kali	20	0	9192	5120	3072	S	0.3	0.1	0:00.52	top
20486	kali	20	0	461896	104508	91188	S	0.3	2.6	0:00.18	qterminal
1	root	20	0	22600	13840	10240	S	0.0	0.3	0:00.68	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pool_workqueue_release
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_g
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_p
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-slub_
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-netns
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
11	root	20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/u4:0-ext4-rsv-conversion
12	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-mm_pe
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_kthread
14	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_rude_kthread
15	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_trace_kthread
16	root	20	0	0	0	0	S	0.0	0.0	0:00.19	ksoftirqd/0
17	root	20	0	0	0	0	R	0.0	0.0	0:00.33	rcu_preempt
18	root	rt	0	0	0	0	S	0.0	0.0	0:00.02	migration/0
19	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
21	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
22	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/1
23	root	rt	0	0	0	0	S	0.0	0.0	0:00.31	migration/1
24	root	20	0	0	0	0	S	0.0	0.0	0:00.30	ksoftirqd/1
25	root	20	0	0	0	0	I	0.0	0.0	0:00.85	kworker/1:0-events
30	root	20	0	0	0	0	I	0.0	0.0	0:00.16	kworker/u6:1-events_unbound
31	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kdevtmpfs

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



## File-i i I/O-bound

```
GNU nano 8.1 cpu_bound.c
#include <stdio.h>

int main(){
    int i = 0;
    while(1){
        i++;
    }
    return 0;
}
```

Read 9 lines

Help Exit Write Out Read File Where Is Replace Cut Paste Execute Justify Location Go To Line Undo Redo Set Mark Copy To Bracket Where Was

Ekzekutimi dhe vrojtimi në *top*:

```
top - 12:42:05 up 42 min, 2 users, load average: 1.00, 0.55, 0.37
Tasks: 175 total, 5 running, 170 sleeping, 0 stopped, 0 zombie
%Cpu(s): 16.8 us, 38.5 sy, 0.0 ni, 44.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 3922.8 total, 119.1 free, 2947.9 used, 3102.6 buff/cache
MiB Swap: 1024.0 total, 795.6 free, 228.4 used, 974.9 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1108	kali	20	0	463464	88452	81964	R	97.6	2.2	0:43.84	qterminal
22776	kali	20	0	2476	1152	1152	R	6.6	0.0	0:04.97	io_bound
675	root	20	0	683952	383304	105828	S	5.4	9.5	2:17.68	Xorg
37	root	20	0	0	0	0	I	1.2	0.0	0:00.75	kworker/u5:2-events_unbound
16130	root	20	0	0	0	0	I	1.2	0.0	0:00.97	kworker/u6:3-events_unbound
1050	kali	20	0	285908	22304	17664	S	0.6	0.6	0:03.98	panel-13-cpugra
1052	kali	20	0	337692	19560	15984	S	0.6	0.5	0:03.01	panel-15-genmon
22837	kali	20	0	461900	104500	91288	R	0.6	2.6	0:00.21	qterminal
22854	root	20	0	0	0	0	I	0.6	0.0	0:00.15	kworker/u5:3-events_unbound
22900	kali	20	0	9192	5120	3072	R	0.6	0.1	0:00.03	top
1	root	20	0	22600	13840	10240	S	0.0	0.3	0:00.69	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pool_workqueue_release
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_g
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_p
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-slub_
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-netns
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
12	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-mm_pe
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_kthread
14	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_rude_kthread
15	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_trace_kthread
16	root	20	0	0	0	0	S	0.0	0.0	0:00.22	ksoftirqd/0
17	root	20	0	0	0	0	I	0.0	0.0	0:00.36	rcu_preempt
18	root	rt	0	0	0	0	S	0.0	0.0	0:00.02	migration/0
19	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
21	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
22	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/1
23	root	rt	0	0	0	0	S	0.0	0.0	0:00.32	migration/1



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, memorijen e CPU-së (*meminfo* file), konfigurimet e sistemit tonë, na jep mundësi të bëjmë ndryshime në sistemin tonë duke modifikuar data structures që përmban sistemi.



```
(kali@kali)-[~]
$ iostat
Linux 6.8.11-amd64 (kali)      11/24/2024      _x86_64_      (2 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           3.03    0.02   2.64    0.20    0.00   94.11

Device            tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
sda                12.96       277.68       363.31         0.00     984083     1287545         0
sr0                0.01         0.02         0.00         0.00         76         0         0

(kali@kali)-[~]
$
```

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 komandën *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.

```
(kali@kali)-[~]
$ pgrep firefox
32510

(kali@kali)-[~]
$ sudo pmap -q 32510
[sudo] password for kali:
32510: /usr/lib/firefox-esr/firefox-esr
0000058af800000 1024K rw--- [ anon ]
000001352c100000 1024K rw--- [ anon ]
000005c898200000 1024K rw--- [ anon ]
000009552f000000 1024K rw--- [ anon ]
000009cd04600000 1024K rw--- [ anon ]
00000db507600000 1024K rw--- [ anon ]
000011fc7f200000 1024K rw--- [ anon ]
00001711dcb00000 1024K rw--- [ anon ]
00001b804a000000 1024K rw--- [ anon ]
00001bfbd7000000 1024K rw--- [ anon ]
00001d9bbf595000 56K r-x-- [ anon ]
00001d9bbf5a3000 8K r-x-- [ anon ]
00001d9bbf5a5000 192K ----- [ anon ]
00001d9bbf5d5000 64K ----- [ anon ]
00001d9bbf5e5000 56K r-x-- [ anon ]
00001d9bbf5f3000 8K r-x-- [ anon ]
00001d9bbf5f5000 256K ----- [ anon ]
00001d9bbf635000 64K ----- [ anon ]
00001d9bbf645000 64K ----- [ anon ]
00001d9bbf655000 64K ----- [ anon ]
00001d9bbf665000 128K ----- [ anon ]
00001d9bbf685000 64K ----- [ anon ]
00001d9bbf695000 64K ----- [ anon ]
00001d9bbf6a5000 2091968K ----- [ anon ]
00001e5cedc00000 1024K rw--- [ anon ]
0000206fa8d00000 1024K rw--- [ anon ]
000021426ae00000 1024K rw--- [ anon ]
0000326fec000000 1024K rw--- [ anon ]
```

```

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----- liblglpplibs.so
00007f8f1a6f2000   20K r-x--- liblglpplibs.so
00007f8f1a6f7000    8K r----- liblglpplibs.so
00007f8f1a6f9000    4K r----- liblglpplibs.so
00007f8f1a6fa000    4K rw----- liblglpplibs.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 stack-ut 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*’ (read/write permission). Kjo ndodh sepse shpeshherë këto 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 102400K 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   52K 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 ]

```

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