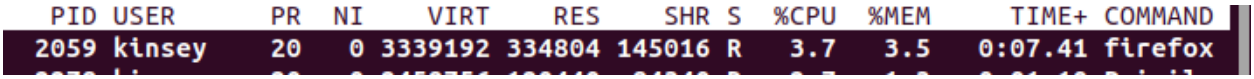
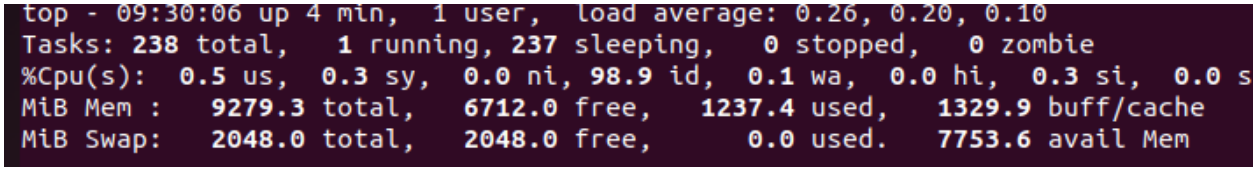
1. The browser consumes the following.



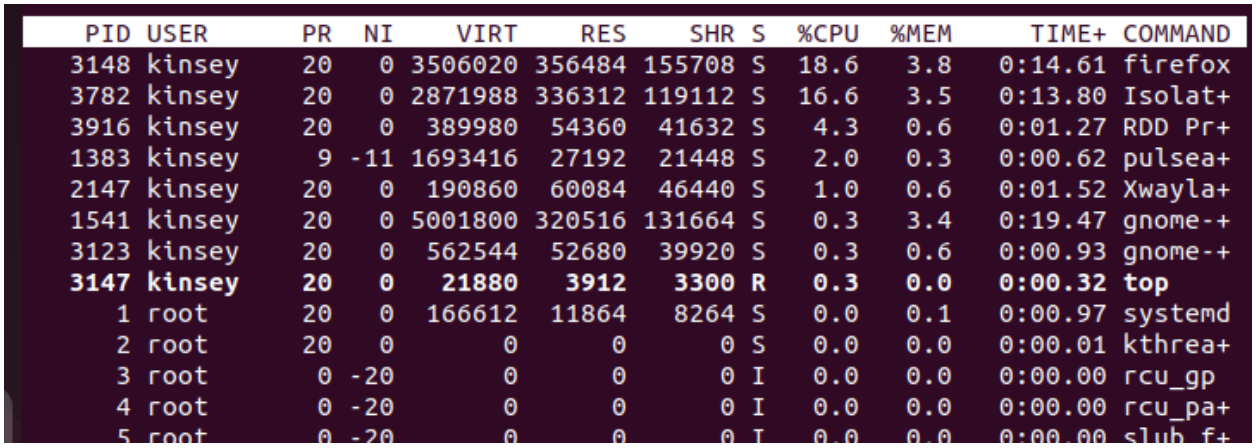
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2059	kinsey	20	0	3339192	334804	145016	R	3.7	3.5	0:07.41	firefox

2. There is 7753.6 available memory



```
top - 09:30:06 up 4 min,  1 user,  load average: 0.26, 0.20, 0.10
Tasks: 238 total,   1 running, 237 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.5 us,  0.3 sy,  0.0 ni, 98.9 id,  0.1 wa,  0.0 hi,  0.3 si,  0.0 s
MiB Mem :   9279.3 total,   6712.0 free,   1237.4 used,   1329.9 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   7753.6 avail Mem
```

3. Isolat+ is consuming the most CPU (16.6%)
4. Firebox is consuming the most Memory (3.8%)



PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3148	kinsey	20	0	3506020	356484	155708	S	18.6	3.8	0:14.61	firefox
3782	kinsey	20	0	2871988	336312	119112	S	16.6	3.5	0:13.80	Isolat+
3916	kinsey	20	0	389980	54360	41632	S	4.3	0.6	0:01.27	RDD Pr+
1383	kinsey	9	-11	1693416	27192	21448	S	2.0	0.3	0:00.62	pulsea+
2147	kinsey	20	0	190860	60084	46440	S	1.0	0.6	0:01.52	Xwayla+
1541	kinsey	20	0	5001800	320516	131664	S	0.3	3.4	0:19.47	gnome-+
3123	kinsey	20	0	562544	52680	39920	S	0.3	0.6	0:00.93	gnome-+
3147	**kinsey**	**20**	**0**	**21880**	**3912**	**3300**	**R**	**0.3**	**0.0**	**0:00.32**	**top**
1	root	20	0	166612	11864	8264	S	0.0	0.1	0:00.97	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthrea+
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_pa+
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	slub_f+

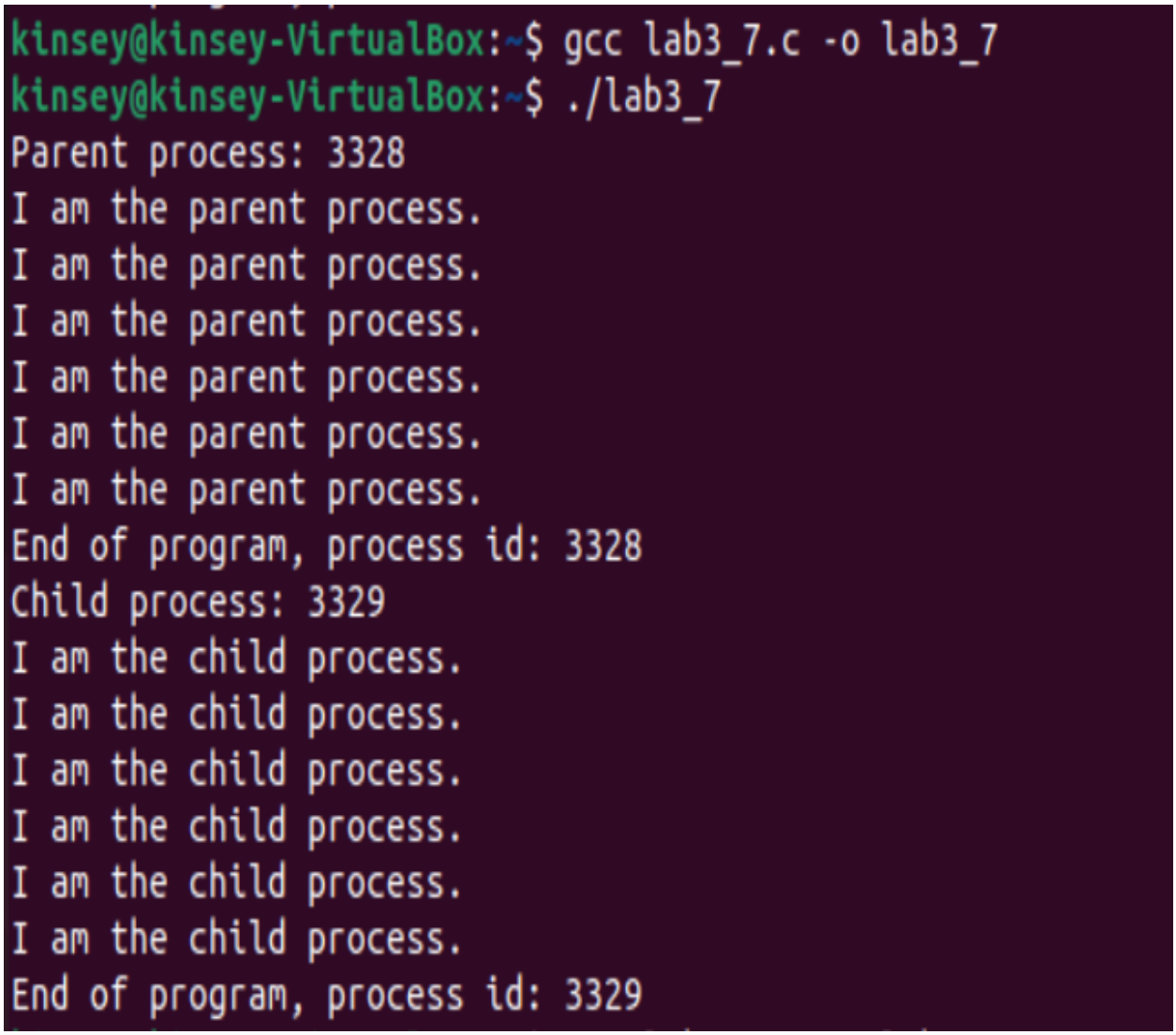
6. **apt-get**: This command is for accessing the Advanced Package Tool library letting you access it to search, install, manage, update, and remove software.
yum: This command is essentially a package manager and is used to make system updates, install packages, remove packages, or examine available and installed packages.
wget: This command is a tool that is a part of the APT library. It's used to collect content and files from web servers.
gzip: This command is used to compress files into a .gz file and deletes the original file.
tar: This command is used for creating, extracting, compressing, updating, and viewing Archive files.
rar: This command is a file format used for data compressing and archiving. It's a tool that is a part of the APT library.

7. Here is my program:

```c
//Question 7
// Write a a program that will generate a child process. In a loop, the child process writes "I am a child process" 200 times and the parent process repeatedly prints "I am a parent process" in a loop.
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(){
        //Step 2: Declare the variables pid, pid1, pid2
        pid_t pid, pid1, pid2;


        //Step 3: Call fork() system call to create process
        pid = fork();

        //IF pid is -1 exit
        if(pid == -1){
                fprintf(stderr, "Error: Unable to create child process.\n");
                exit(1);
        }

        //If pid is NOT 0 the print parent process
        if(pid != 0){
                pid1 = getpid();
                printf("Parent process: %d\n", pid1);
```

```c
                printf("Parent process: %d\n", pid1);
                int i;
                while(i < 200){
                        printf("I am the parent process.\n");
                        i++;
                }
        }else{
                pid2 = getpid();
                printf("Child process: %d\n", pid2);
                int j;
                while(j < 200){
                        fprintf(stderr, "%s %s %s %s %s","I", "am", "the", "child", "process.\n");
                        j++;
                }


        }

        //Step 7: Stop the program
        printf("End of program, process id: %d\n", getpid());
        return 0;
}
```

Here is my question 7 program running (I modified it to just print 5 times for each process so that it would fit in a single screenshot)



8. Here is my program:

```c
//Lab3: Question 8

#include<stdio.h>
#include<sys/wait.h>
#include<unistd.h>
#include <dirent.h>

int main(){

        pid_t pid = fork();

        if(pid < 0){
                fprintf(stderr, "Fork fail.");
                return 1;
        }else if (pid == 0){
                printf("Child Process...\n");
        }else{
                printf("Parent Process is waiting for Child Process to end..
\n");
                //parent waits for child to complete
                wait(NULL);
                printf("Child has ended.\n");

                //Once complete print contents of current directory
                // Pointer for directory entry
                    // Pointer for directory entry
                    struct dirent *de;

                    // opendir() returns a pointer of DIR type.
                    DIR *dr = opendir(".");
                    // opendir returns NULL if couldn't open directory
                    if (dr == NULL){
                            printf("Could not open current directory" );
                            return 0;
                    }
                    // printing out content in current directory
                    while ((de = readdir(dr)) != NULL){
                            printf("%s\n", de->d_name);

                            closedir(dr);
                    }
        }

    return 0;
}
```

Here is the program running:

```
kinsey@kinsey-VirtualBox:~$ gcc lab3_8.c -o lab3_8
kinsey@kinsey-VirtualBox:~$ ./lab3_8
Parent Process is waiting for Child Process to end...
Child Process...
Child has ended.
.thunderbird
lab3
```

9. Here is my code:

```
1 // Lab3: Question 9
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <unistd.h>
5 int main(){
6         //Declare the variables pid, pid1, pid2
7         pid_t pid, pid1, pid2;
8         printf("Start of the program, process id: %d\n", getpid());
9
10        //Call fork() system call to create process
11        pid = fork();
12
13        //If pid == -1, exit
14        if(pid == -1){
15                fprintf(stderr, "Error: Unable to create child process.\n");
16                exit(1);
17        }
18
19        //If pid != -1, get the process id using getpid()
20        if(pid != 0){
21                pid1 = getpid();
22                printf("Process is of parent process: %d\n", pid1);
23                //parent process printing PID of child
24                printf("In Parent Process the Child's pid is %d\n", pid);
25        }else{
```

```
25        }else{
26                pid2 = getpid();
27                printf("\nProcess is of child process: %d\n", pid2);
28                //Child printing PID of parent
29                printf("In Child Process the Parents's pid is %d\n",
  getppid());
30        }
31
32        //Stop the program
33        printf("End of program, process id: %d\n", getpid());
34        return 0;
35 }
```

Here is the Question 9 program running:

```
kinsey@kinsey-VirtualBox:~$ gcc lab3_9.c -o lab3_9
kinsey@kinsey-VirtualBox:~$ ./lab3_9
Start of the program, process id: 7785
Process is of parent process: 7785
In Parent Process the Child's pid is 7786
End of program, process id: 7785

Process is of child process: 7786
In Child Process the Parents's pid is 7785
End of program, process id: 7786
```