

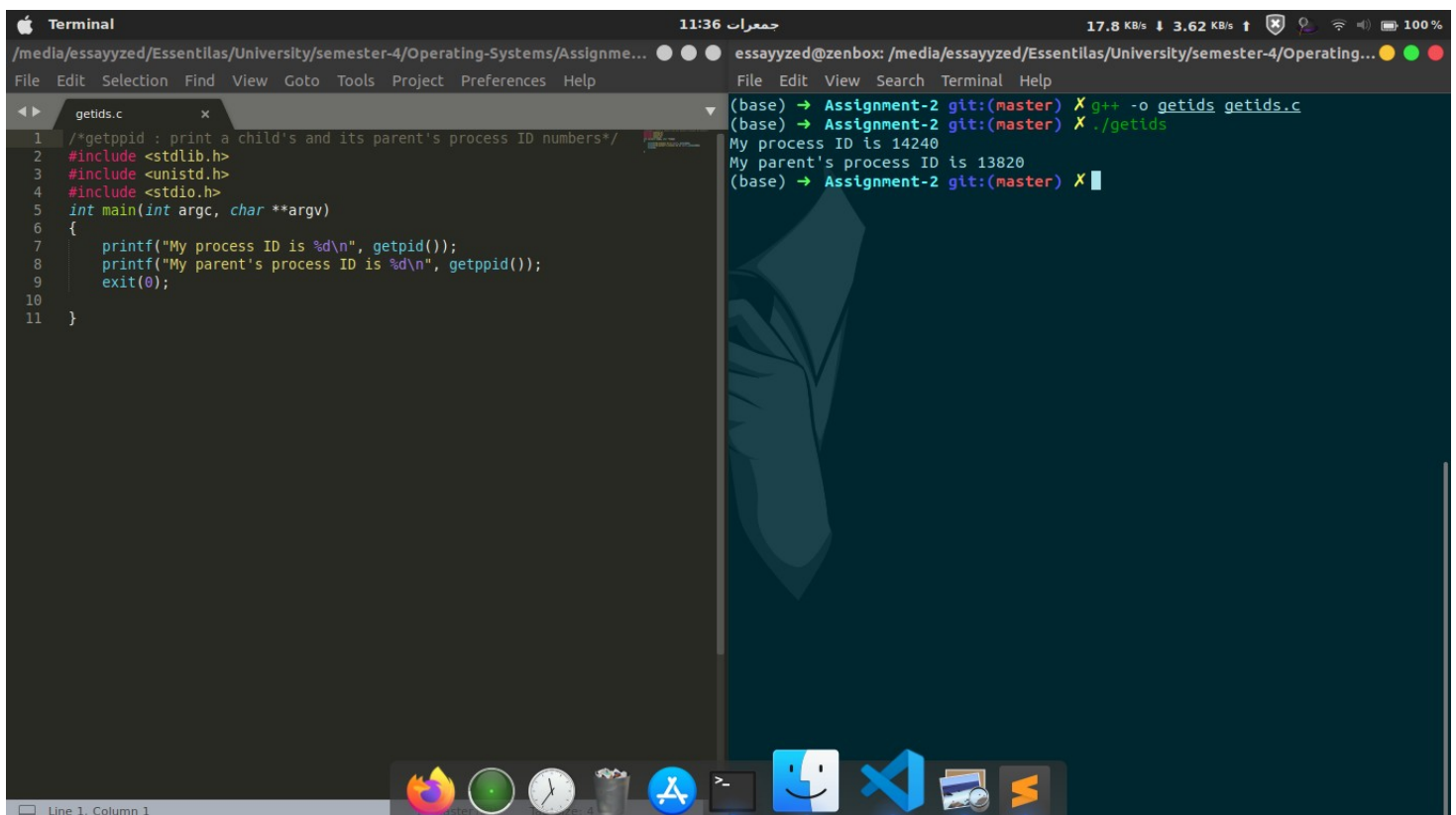
Assignment-2

CS-220 Operating System
Fork System Call

Syed Asad Zaman
p18-0034
Section B

7.

Getting PIDs(Process & Parent)

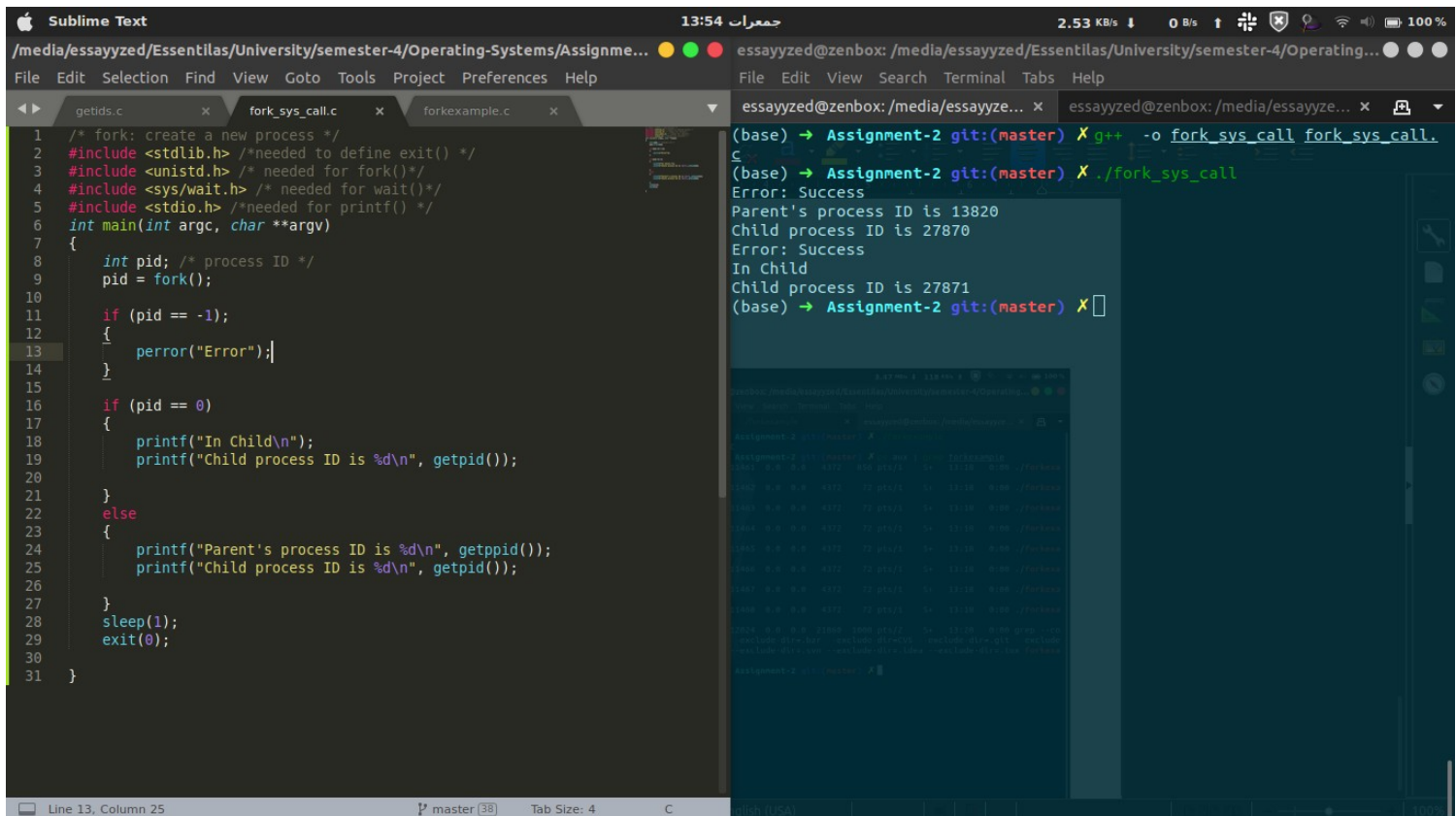


```
Terminal
/media/essayzed/Essentilas/University/semester-4/Operating-Systems/Assignme... essayzed@zenbox: /media/essayzed/Essentilas/University/semester-4/Operating...
File Edit Selection Find View Goto Tools Project Preferences Help

getids.c x
1 /*getppid : print a child's and its parent's process ID numbers*/
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <stdio.h>
5 int main(int argc, char **argv)
6 {
7     printf("My process ID is %d\n", getpid());
8     printf("My parent's process ID is %d\n", getppid());
9     exit(0);
10 }
11 }

(base) → Assignment-2 git:(master) X g++ -o getids getids.c
(base) → Assignment-2 git:(master) X ./getids
My process ID is 14240
My parent's process ID is 13820
(base) → Assignment-2 git:(master) X
```

2. Fork System Call



The screenshot shows a Sublime Text editor with a C program that demonstrates the use of the `fork()` system call. The program is named `fork_sys_call.c` and is located in the directory `/media/essayzyed/Essentilas/University/semester-4/Operating-Systems/Assignme...`. The code is as follows:

```
1 /* fork: create a new process */
2 #include <stdlib.h> /*needed to define exit() */
3 #include <unistd.h> /* needed for fork()*/
4 #include <sys/wait.h> /* needed for wait()*/
5 #include <stdio.h> /*needed for printf() */
6 int main(int argc, char **argv)
7 {
8     int pid; /* process ID */
9     pid = fork();
10
11     if (pid == -1);
12     {
13         perror("Error");
14     }
15
16     if (pid == 0)
17     {
18         printf("In Child\n");
19         printf("Child process ID is %d\n", getpid());
20     }
21     else
22     {
23         printf("Parent's process ID is %d\n", getppid());
24         printf("Child process ID is %d\n", getpid());
25     }
26     sleep(1);
27     exit(0);
28 }
```

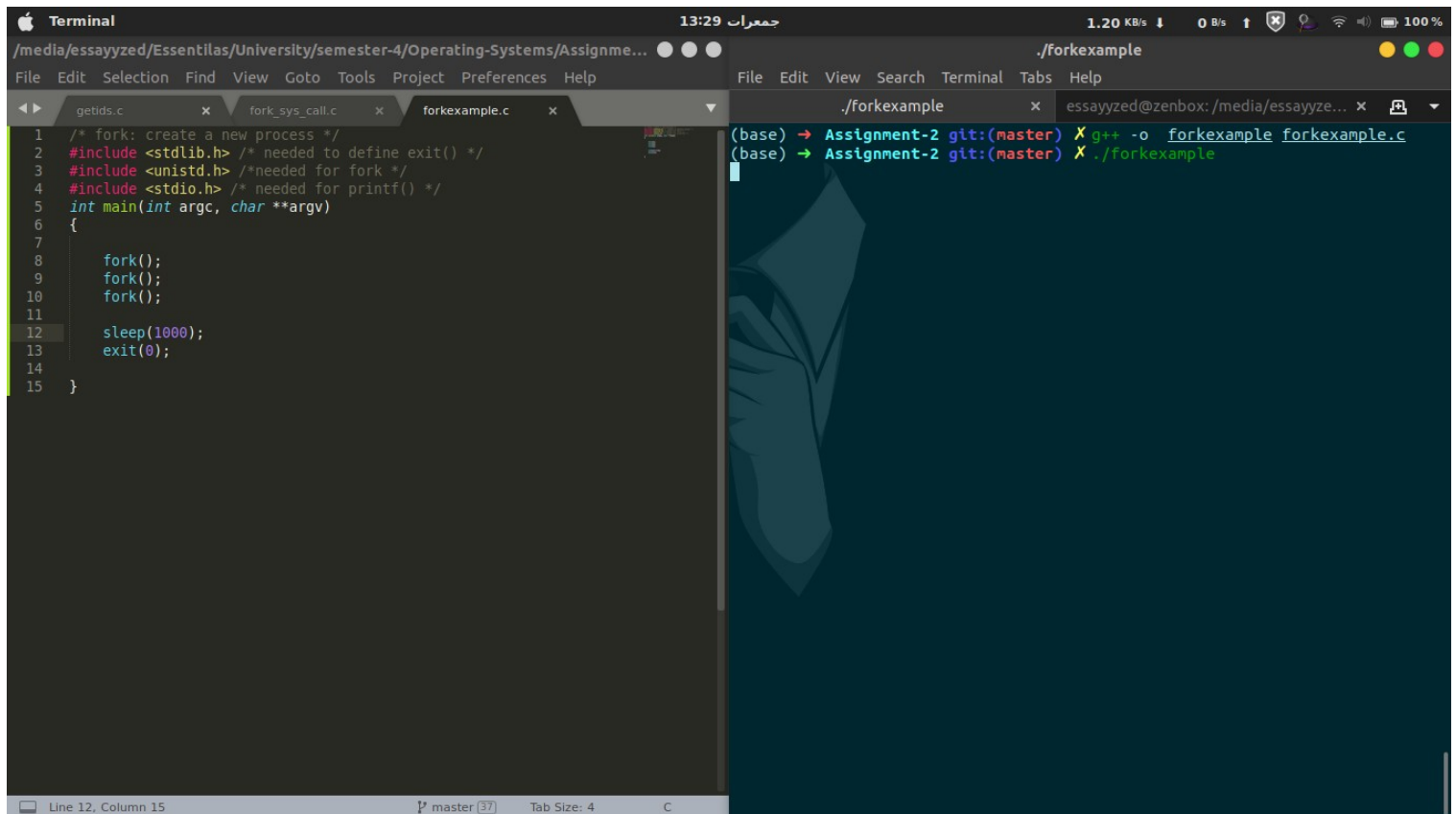
The terminal window shows the execution of the program. The user runs `g++ -o fork_sys_call fork_sys_call.c` and `./fork_sys_call`. The output is as follows:

```
(base) → Assignment-2 git:(master) X g++ -o fork_sys_call fork_sys_call.c
(base) → Assignment-2 git:(master) X ./fork_sys_call
Error: Success
Parent's process ID is 13820
Child process ID is 27870
Error: Success
In Child
Child process ID is 27871
(base) → Assignment-2 git:(master) X
```

The terminal window also shows a list of files in the directory, including `Assignment-2`, `fork_sys_call.c`, `fork_sys_call.o`, and `fork_sys_call`.

3.

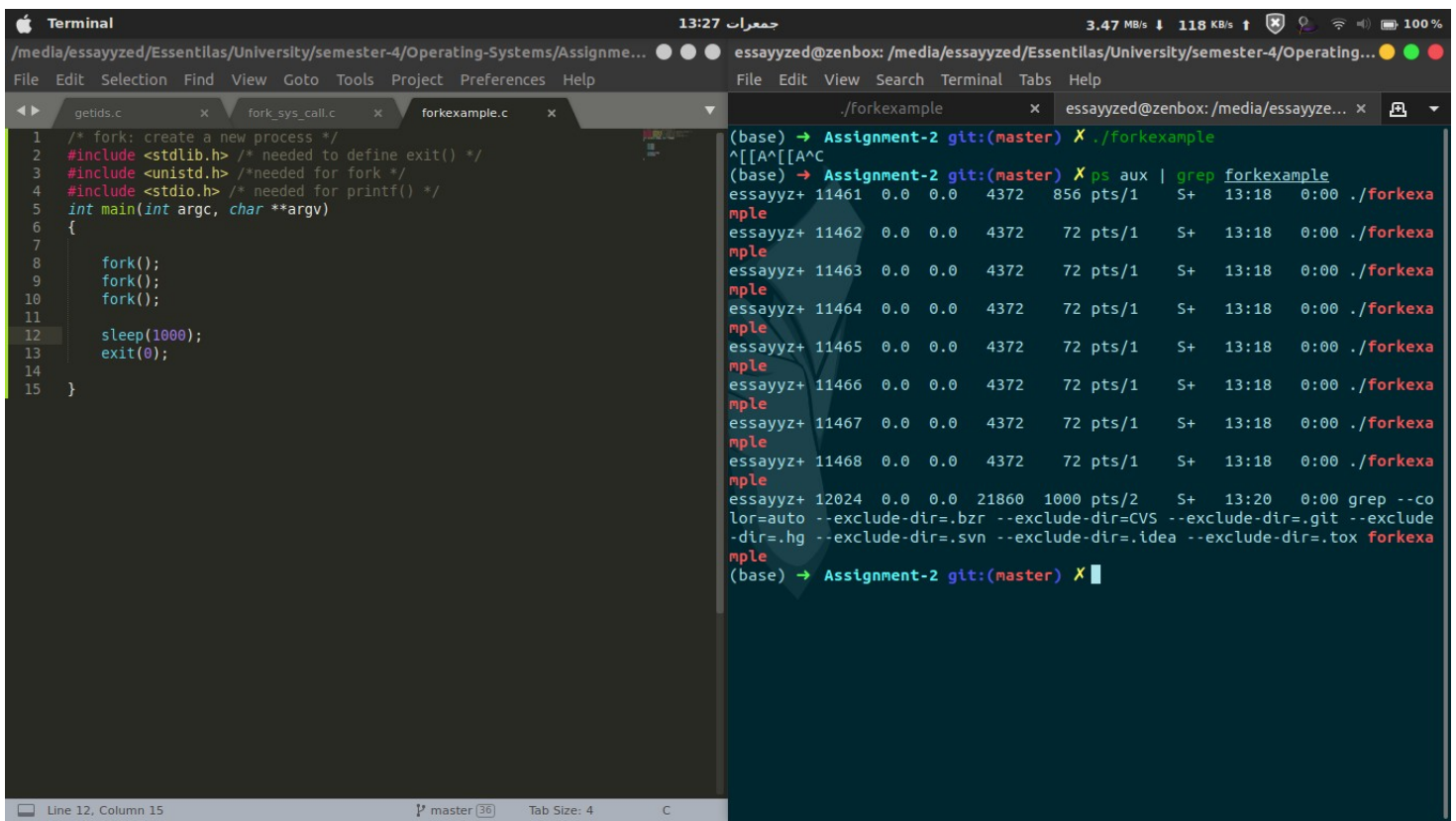
Fork Processes Count



```
1 /* fork: create a new process */
2 #include <stdlib.h> /* needed to define exit() */
3 #include <unistd.h> /* needed for fork */
4 #include <stdio.h> /* needed for printf() */
5 int main(int argc, char **argv)
6 {
7
8     fork();
9     fork();
10    fork();
11
12    sleep(1000);
13    exit(0);
14
15 }
```

The terminal window shows the execution of the program. The prompt is `(base) → Assignment-2 git:(master) X`. The user enters `g++ -o forkexample forkexample.c` and then `./forkexample`. The program runs for 1000 seconds and then exits.

Processes : 8



```
1 /* fork: create a new process */
2 #include <stdlib.h> /* needed to define exit() */
3 #include <unistd.h> /* needed for fork */
4 #include <stdio.h> /* needed for printf() */
5 int main(int argc, char **argv)
6 {
7
8     fork();
9     fork();
10    fork();
11
12    sleep(1000);
13    exit(0);
14
15 }
```

The terminal window shows the execution of the program. The prompt is `(base) → Assignment-2 git:(master) X`. The user enters `./forkexample`. The program runs for 1000 seconds and then exits. The user then enters `ps aux | grep forkexample` to count the number of processes. The output shows 8 processes: the parent process and 7 child processes.

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	TIME	COMMAND	
essayyz+	11461	0.0	0.0	4372	856	pts/1	S+	13:18 0:00 ./forkexample	
mple	essayyz+	11462	0.0	0.0	4372	72	pts/1	S+	13:18 0:00 ./forkexample
mple	essayyz+	11463	0.0	0.0	4372	72	pts/1	S+	13:18 0:00 ./forkexample
mple	essayyz+	11464	0.0	0.0	4372	72	pts/1	S+	13:18 0:00 ./forkexample
mple	essayyz+	11465	0.0	0.0	4372	72	pts/1	S+	13:18 0:00 ./forkexample
mple	essayyz+	11466	0.0	0.0	4372	72	pts/1	S+	13:18 0:00 ./forkexample
mple	essayyz+	11467	0.0	0.0	4372	72	pts/1	S+	13:18 0:00 ./forkexample
mple	essayyz+	11468	0.0	0.0	4372	72	pts/1	S+	13:18 0:00 ./forkexample