



ASSIGNMENT # 3

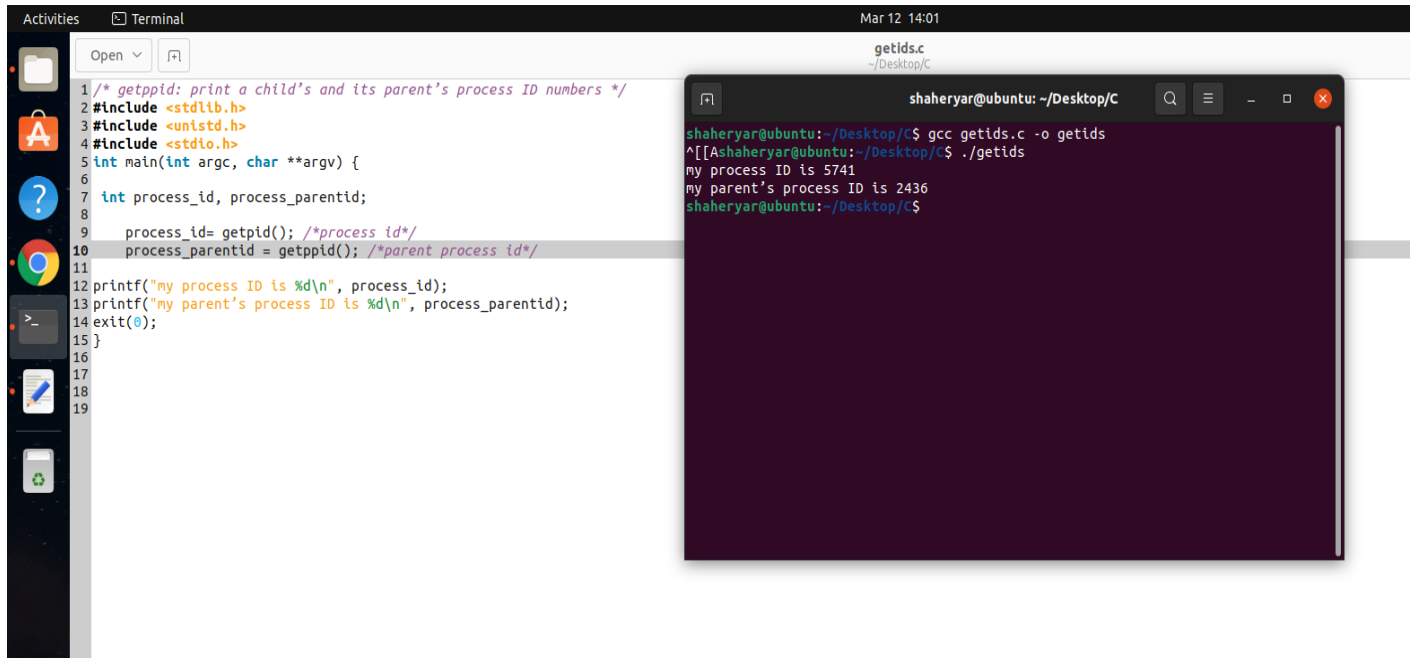
NAME : SHAHERYAR ASHFAQ

ROLL NO : **20P-0128**

SECTION : BS-CS **4-B**

SUBJECT : OPERATING SYSTEM

QUESTION -1



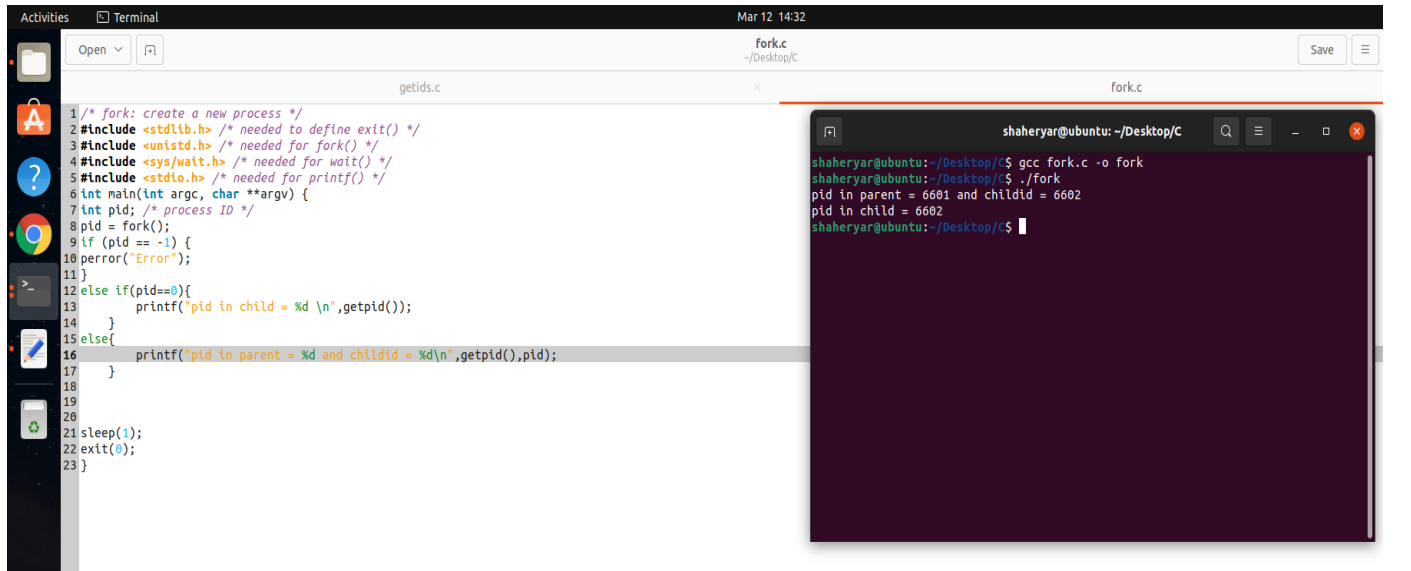
The screenshot shows a terminal window with the following content:

```
1 /* getpid: 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     int process_id, process_parentid;
8
9     process_id= getpid(); /*process id*/
10    process_parentid = getppid(); /*parent process id*/
11
12    printf("my process ID is %d\n", process_id);
13    printf("my parent's process ID is %d\n", process_parentid);
14    exit(0);
15 }
16
17
18
19
```

The terminal output shows the compilation and execution of the program:

```
shaheryar@ubuntu: ~/Desktop/C
shaheryar@ubuntu:~/Desktop/C$ gcc getids.c -o getids
shaheryar@ubuntu:~/Desktop/C$ ./getids
my process ID is 5741
my parent's process ID is 2436
shaheryar@ubuntu:~/Desktop/C$
```

QUESTION -2



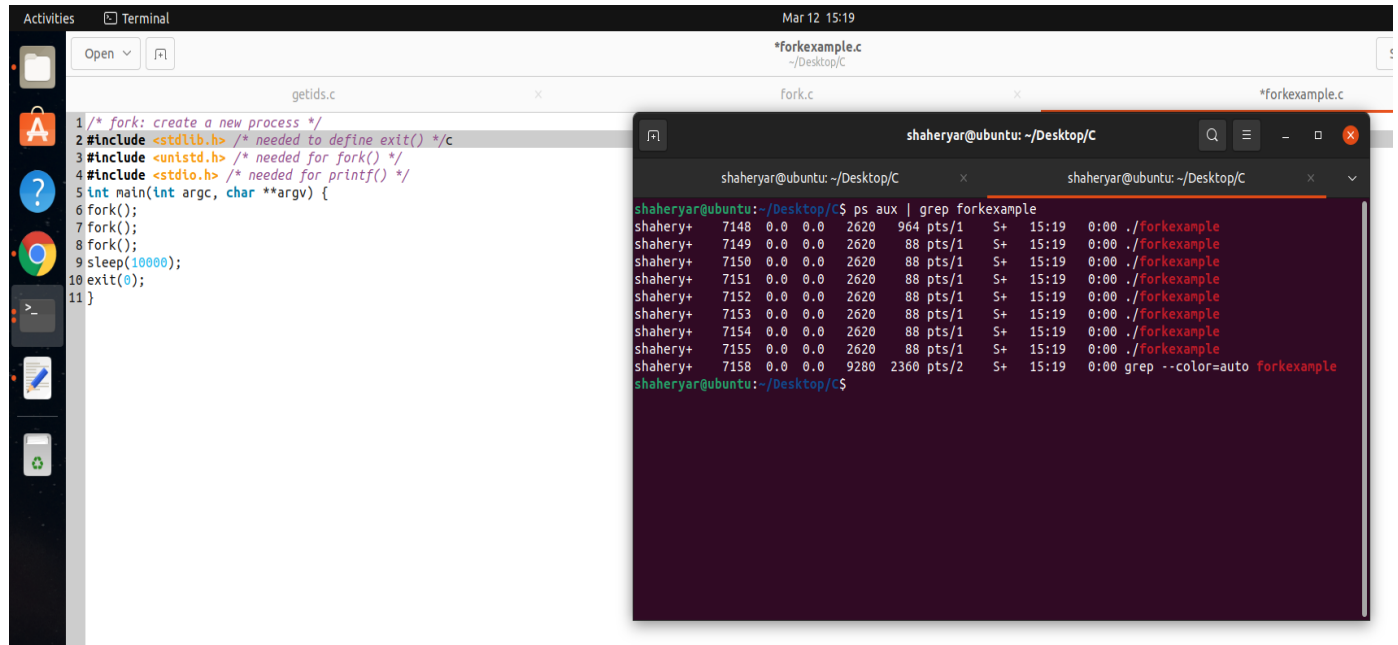
The screenshot shows a terminal window with the following content:

```
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     int pid; /* process ID */
8     pid = fork();
9     if (pid == -1) {
10        perror("Error");
11    }
12    else if (pid==0){
13        printf("pid in child = %d \n",getpid());
14    }
15    else{
16        printf("pid in parent = %d and childid = %d\n",getpid(),pid);
17    }
18
19
20
21    sleep(1);
22    exit(0);
23 }
```

The terminal output shows the compilation and execution of the program:

```
shaheryar@ubuntu:~/Desktop/C$ gcc fork.c -o fork
shaheryar@ubuntu:~/Desktop/C$ ./fork
pid in parent = 6601 and childid = 6602
pid in child = 6602
shaheryar@ubuntu:~/Desktop/C$
```

QUESTION -3



The screenshot displays a Linux desktop with a terminal window and a code editor. The code editor shows a C program named `forkexample.c` located at `~/Desktop/C`. The program uses `fork()` to create child processes. The terminal window shows the command `ps aux | grep forkexample` being executed, resulting in a list of processes.

```
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     fork();
7     fork();
8     fork();
9     sleep(10000);
10    exit(0);
11 }
```

The terminal output shows the following processes:

USER	PID	PPID	CPU	MEM	VSZ	RSS	TTY	STAT	TIME	COMMAND
shaheryar	7148	0.0	0.0	2620	964	pts/1	S+	15:19	0:00	./forkexample
shaheryar	7149	0.0	0.0	2620	88	pts/1	S+	15:19	0:00	./forkexample
shaheryar	7150	0.0	0.0	2620	88	pts/1	S+	15:19	0:00	./forkexample
shaheryar	7151	0.0	0.0	2620	88	pts/1	S+	15:19	0:00	./forkexample
shaheryar	7152	0.0	0.0	2620	88	pts/1	S+	15:19	0:00	./forkexample
shaheryar	7153	0.0	0.0	2620	88	pts/1	S+	15:19	0:00	./forkexample
shaheryar	7154	0.0	0.0	2620	88	pts/1	S+	15:19	0:00	./forkexample
shaheryar	7155	0.0	0.0	2620	88	pts/1	S+	15:19	0:00	./forkexample
shaheryar	7158	0.0	0.0	9280	2360	pts/2	S+	15:19	0:00	grep --color=auto forkexample