

NAME: ASJID TAHIR

ROLL NO: 19P-0085

SECTION: BSE-A

COURSE: OPERATING SYSTEM

INSTRUCTOR: DR.NOUMAN

ASSIGNMENT # 02:

FORK SYSTEM CALL:

QUESTION NO 01:

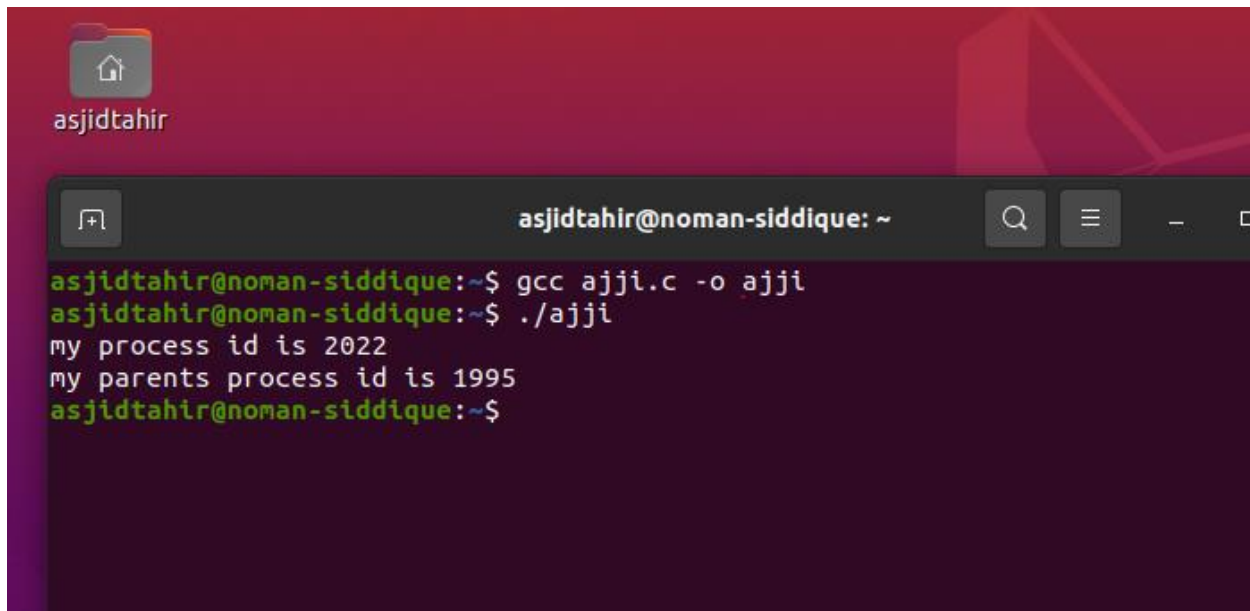


The image shows a code editor window titled 'ajji.c' with the following C code:

```
1 #include<unistd.h>
2 #include<stdlib.h>
3 #include<stdio.h>
4 int main(int argc, char **argv)
5 {
6     printf("my process id is %d\n",getpid());
7     printf("my parents process id is %d\n",getppid());
8     exit(0);
9 }
10
```

Below the code editor, a terminal window is open, showing the command `gedit ajji.c` being executed.

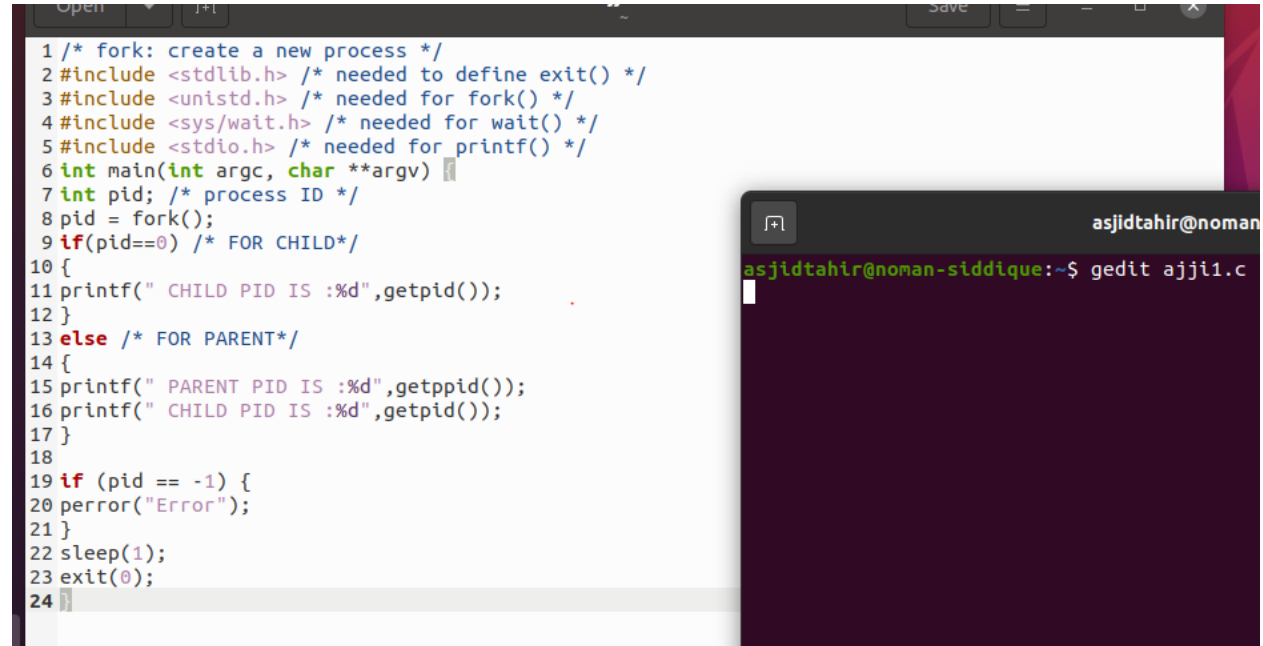
OUTPUT:



The image shows a terminal window with the following output:

```
asjdtahir@noman-siddique:~$ gcc ajji.c -o ajji
asjdtahir@noman-siddique:~$ ./ajji
my process id is 2022
my parents process id is 1995
asjdtahir@noman-siddique:~$
```

QUESTION NO 02:

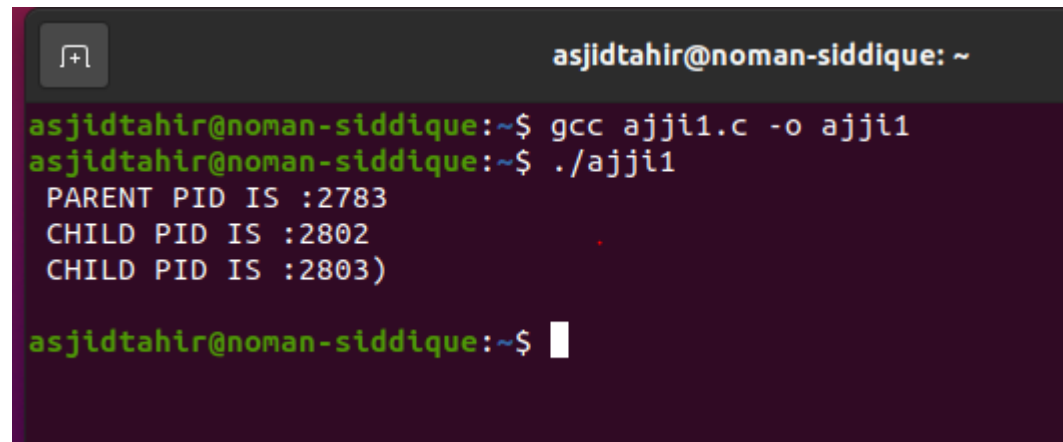


The image shows a code editor window on the left and a terminal window on the right. The code editor contains a C program that uses the `fork()` system call to create a new process. The program includes headers for `stdlib.h`, `unistd.h`, `sys/wait.h`, and `stdio.h`. It defines a `main` function that takes `argc` and `argv` as arguments. It declares a `pid` variable and calls `fork()` to create a child process. If `fork()` returns 0, it indicates the child process, and it prints the child's PID. Otherwise, it indicates the parent process, and it prints the parent's PID and the child's PID. The program also includes a `sleep(1)` call and an `exit(0)` call. The terminal window on the right shows the command `gedit ajji1.c` being executed.

```
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==0) /* FOR CHILD*/
10 {
11     printf(" CHILD PID IS :%d",getpid());
12 }
13 else /* FOR PARENT*/
14 {
15     printf(" PARENT PID IS :%d",getppid());
16     printf(" CHILD PID IS :%d",getpid());
17 }
18
19 if (pid == -1) {
20     perror("Error");
21 }
22 sleep(1);
23 exit(0);
24 }
```

asjdtahir@noman-siddique:~\$ gedit ajji1.c

OUTPUT:

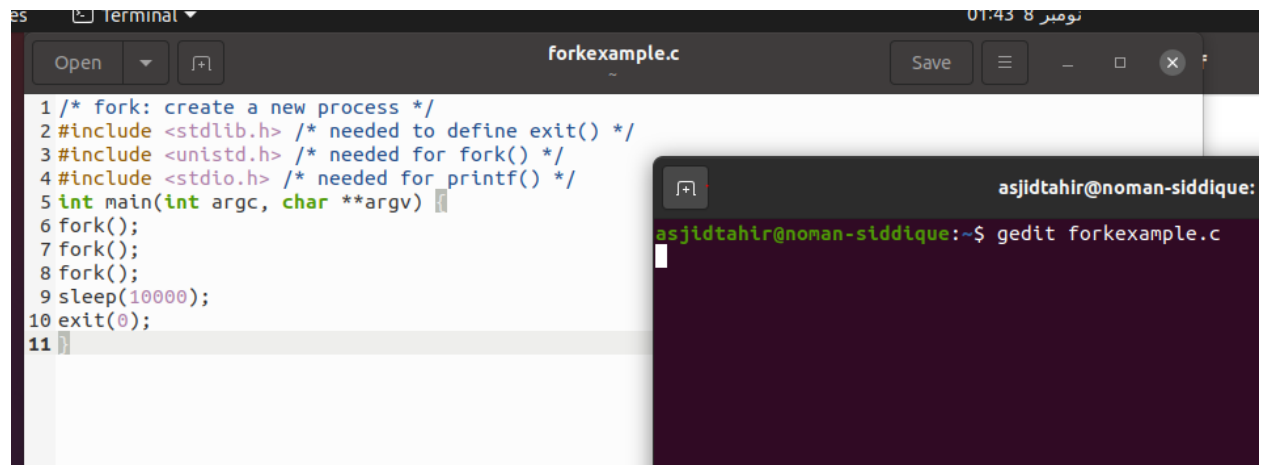


The image shows a terminal window with the output of the program. The user has compiled the program using `gcc ajji1.c -o ajji1` and then executed it using `./ajji1`. The output shows the parent PID as 2783, the child PID as 2802, and the child PID as 2803. The terminal window title is `asjdtahir@noman-siddique: ~`.

```
asjdtahir@noman-siddique:~$ gcc ajji1.c -o ajji1
asjdtahir@noman-siddique:~$ ./ajji1
PARENT PID IS :2783
CHILD PID IS :2802
CHILD PID IS :2803

asjdtahir@noman-siddique:~$
```

QUESTION NO 03:

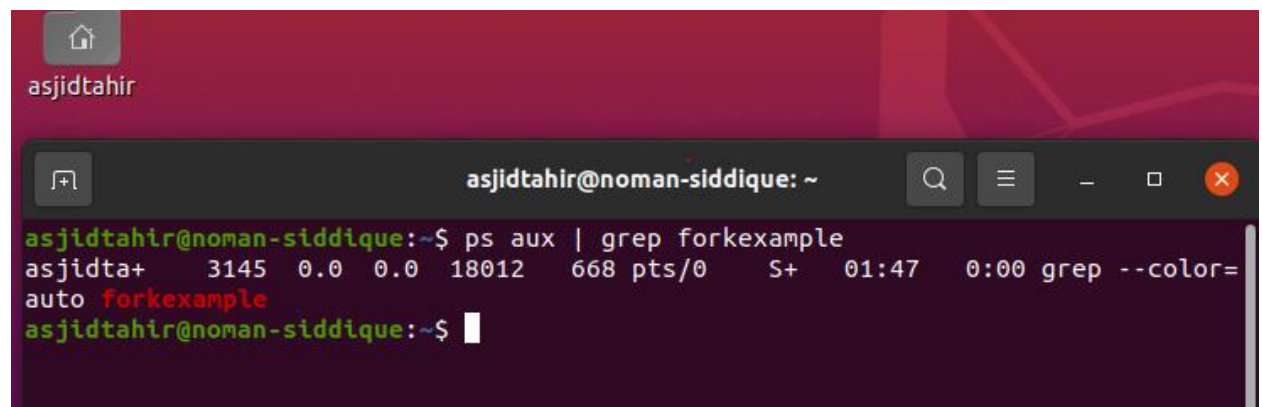


The image shows a code editor window titled 'forkexample.c' with the following C code:

```
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 }
```

Below the code editor is a terminal window titled 'asjdtahir@noman-siddique:'. The terminal shows the command 'gedit forkexample.c' being executed.

OUTPUT:



The image shows a terminal window titled 'asjdtahir@noman-siddique: ~'. The terminal shows the command 'ps aux | grep forkexample' being executed, resulting in the following output:

```
asjdtahir@noman-siddique:~$ ps aux | grep forkexample
asjdtah+  3145  0.0  0.0 18012  668 pts/0    S+   01:47   0:00 grep --color=
auto forkexample
asjdtahir@noman-siddique:~$
```