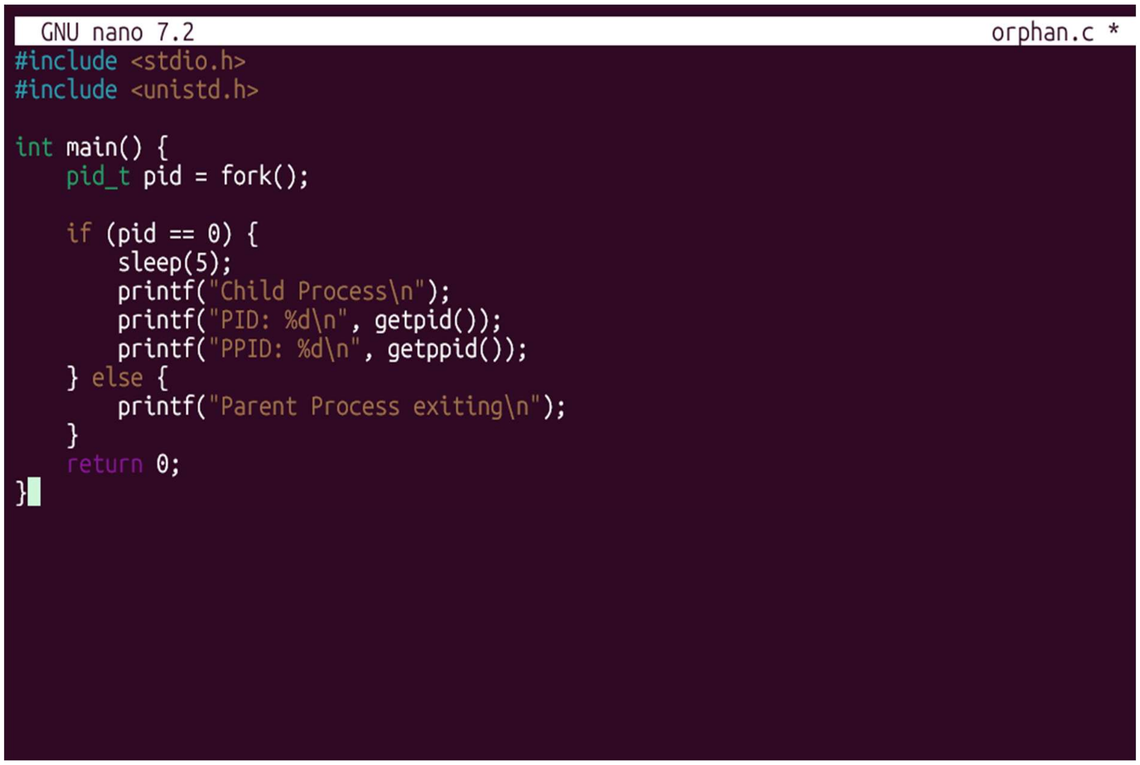
<u>Code/Output:-</u>

2. Write a program for process creation using C

- **Orphan Process**

Code:-

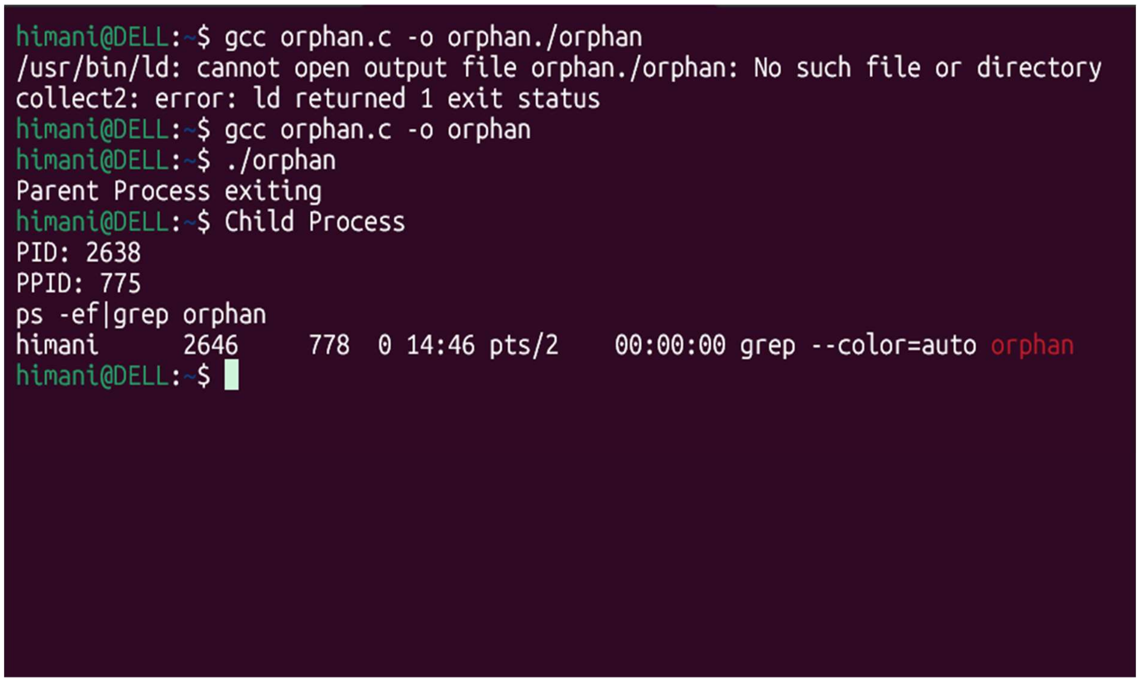


```
GNU nano 7.2                                                    orphan.c *
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid = fork();

    if (pid == 0) {
        sleep(5);
        printf("Child Process\n");
        printf("PID: %d\n", getpid());
        printf("PPID: %d\n", getppid());
    } else {
        printf("Parent Process exiting\n");
    }
    return 0;
}
```

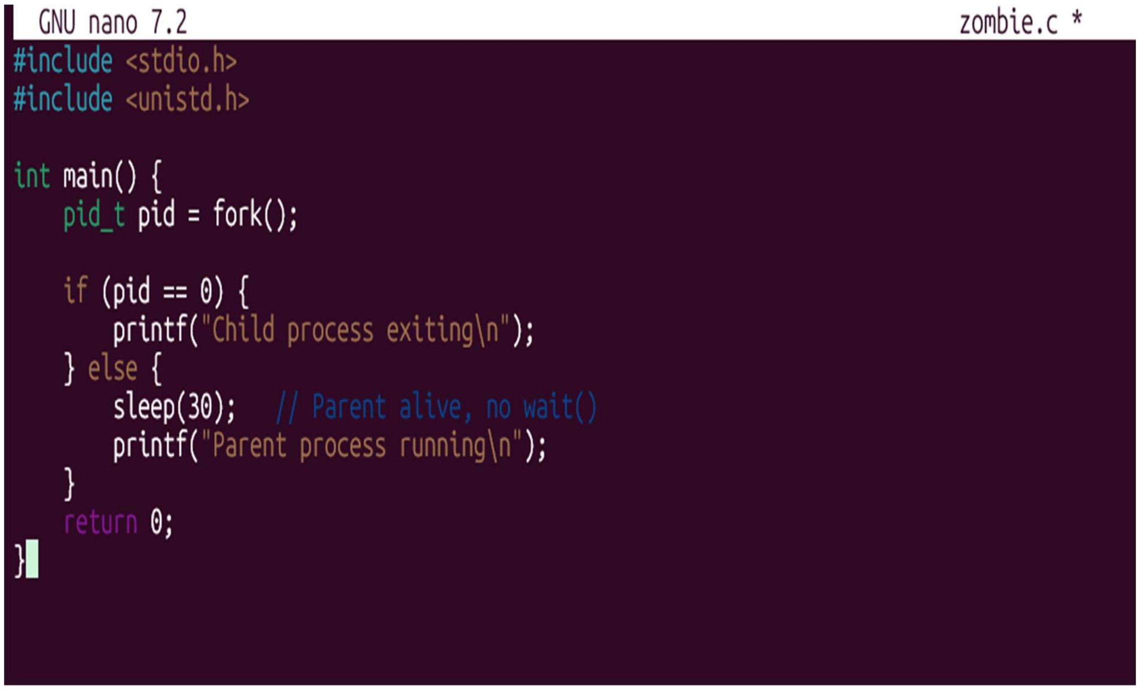
Output:-



```
himani@DELL:~$ gcc orphan.c -o orphan./orphan
/usr/bin/ld: cannot open output file orphan./orphan: No such file or directory
collect2: error: ld returned 1 exit status
himani@DELL:~$ gcc orphan.c -o orphan
himani@DELL:~$ ./orphan
Parent Process exiting
himani@DELL:~$ Child Process
PID: 2638
PPID: 775
ps -ef|grep orphan
himani      2646     778  0 14:46 pts/2    00:00:00 grep --color=auto orphan
himani@DELL:~$ 
```

- **Zombie Process**

Code:-

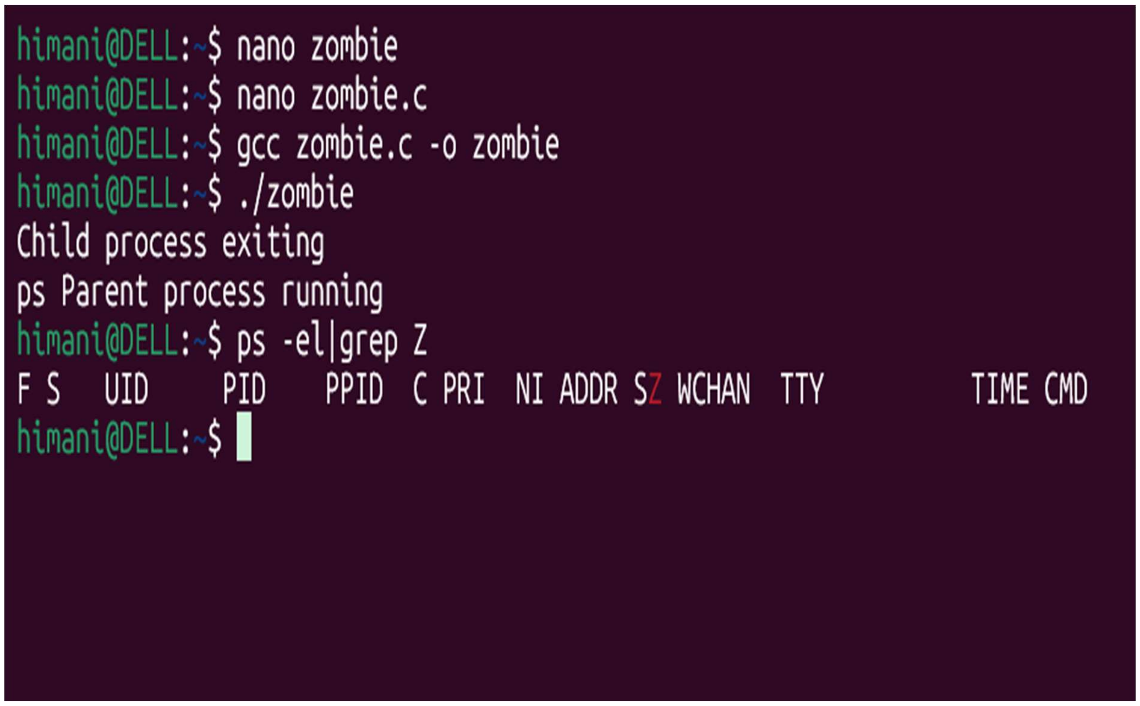


```
GNU nano 7.2                                              zombie.c *
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid = fork();

    if (pid == 0) {
        printf("Child process exiting\n");
    } else {
        sleep(30);    // Parent alive, no wait()
        printf("Parent process running\n");
    }
    return 0;
}
```

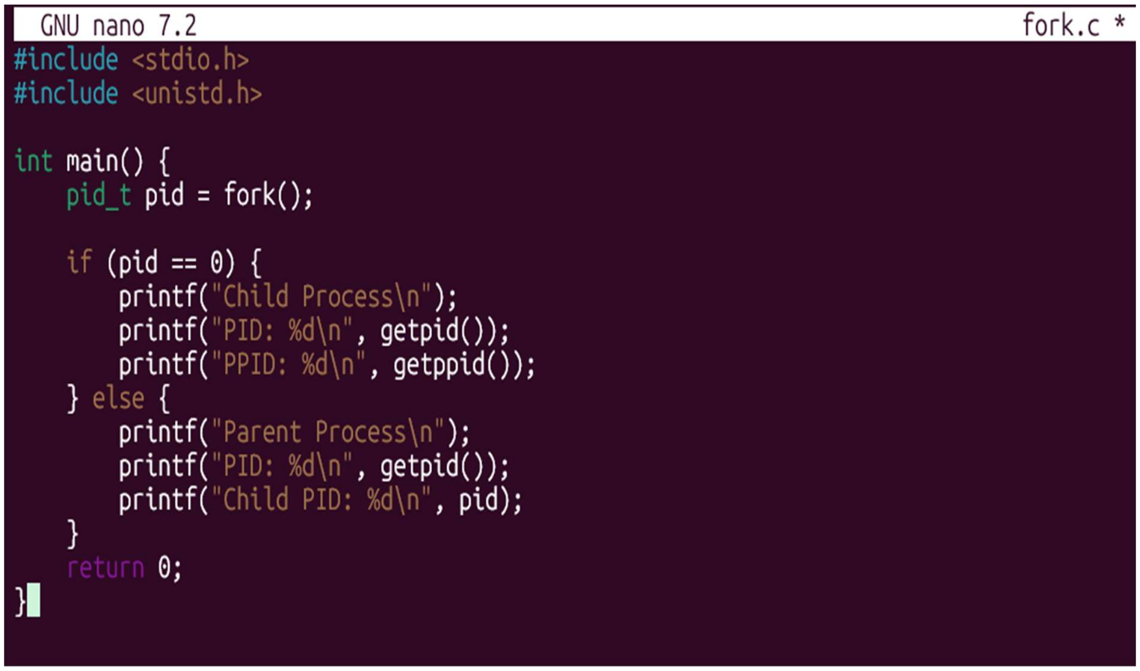
Output:-



```
himani@DELL:~$ nano zombie
himani@DELL:~$ nano zombie.c
himani@DELL:~$ gcc zombie.c -o zombie
himani@DELL:~$ ./zombie
Child process exiting
ps Parent process running
himani@DELL:~$ ps -el|grep Z
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
himani@DELL:~$
```

4. **Create the process using fork 0 system call**
    - **Child Process creation**
    - **Parent Process creation**

Code:-

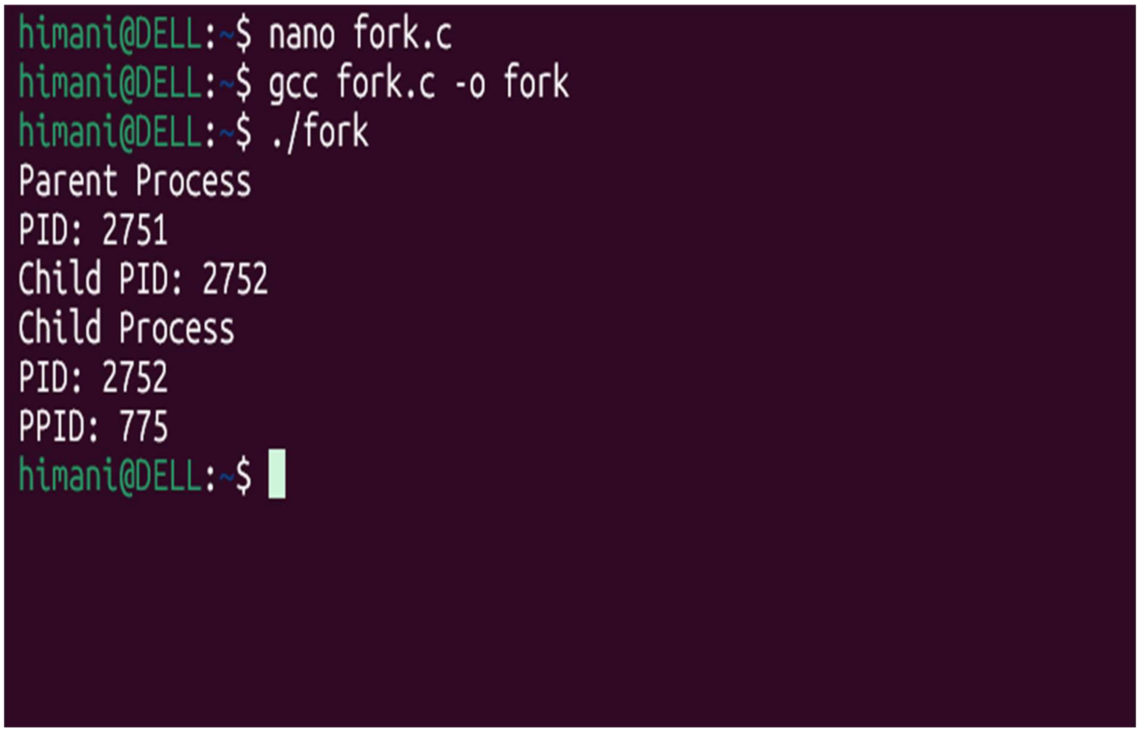


```
GNU nano 7.2                                              fork.c *
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid = fork();

    if (pid == 0) {
        printf("Child Process\n");
        printf("PID: %d\n", getpid());
        printf("PPID: %d\n", getppid());
    } else {
        printf("Parent Process\n");
        printf("PID: %d\n", getpid());
        printf("Child PID: %d\n", pid);
    }
    return 0;
}
```

Output:-



```
himani@DELL:~$ nano fork.c
himani@DELL:~$ gcc fork.c -o fork
himani@DELL:~$ ./fork
Parent Process
PID: 2751
Child PID: 2752
Child Process
PID: 2752
PPID: 775
himani@DELL:~$
```