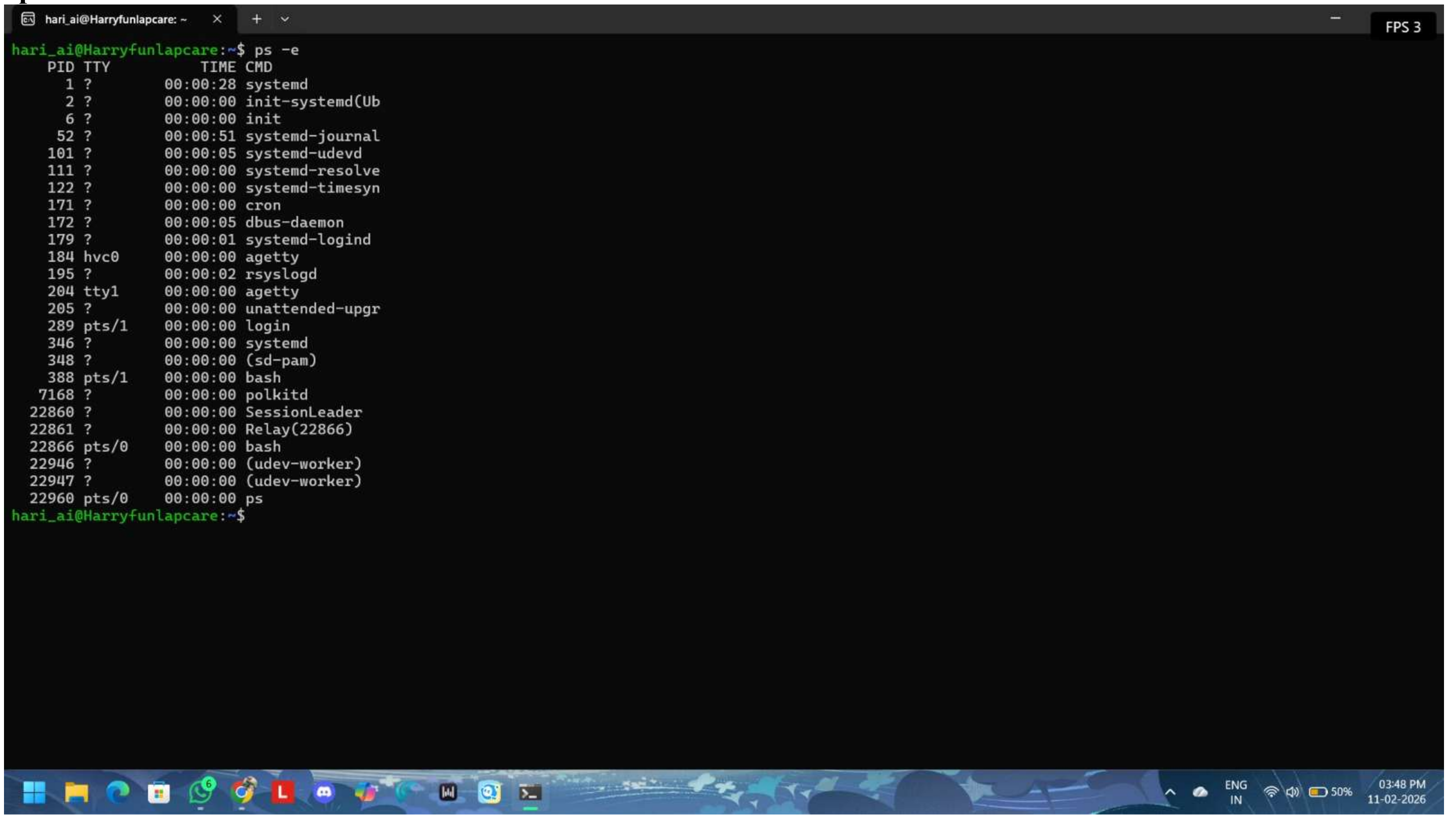
- PID & PPID

Q1. Adam is working in an IT company. He has been given a task to reduce the load of a system by killing some of the processes running in the LINUX operating system. Which commands will he use to complete the given task with the help of the following operation?

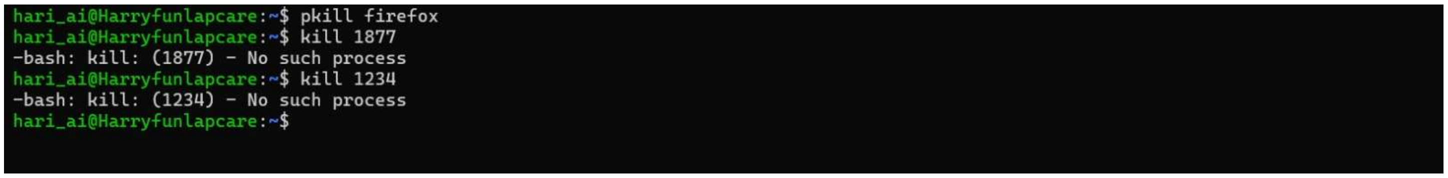
ps -e



```
hari_ai@Harryfunlapcare:~$ ps -e
    PID TTY          TIME CMD
      1 ?        00:00:28 systemd
      2 ?        00:00:00 init-systemd(Ub
      6 ?        00:00:00 init
     52 ?        00:00:51 systemd-journal
    101 ?        00:00:05 systemd-udevd
    111 ?        00:00:00 systemd-resolve
    122 ?        00:00:00 systemd-timesyn
    171 ?        00:00:00 cron
    172 ?        00:00:05 dbus-daemon
    179 ?        00:00:01 systemd-logind
    184 hvc0     00:00:00 agetty
    195 ?        00:00:02 rsyslogd
    204 tty1     00:00:00 agetty
    205 ?        00:00:00 unattended-upgr
    289 pts/1    00:00:00 login
    346 ?        00:00:00 systemd
    348 ?        00:00:00 (sd-pam)
    388 pts/1    00:00:00 bash
   7168 ?        00:00:00 polkitd
  22860 ?        00:00:00 SessionLeader
  22861 ?        00:00:00 Relay(22866)
  22866 pts/0    00:00:00 bash
  22946 ?        00:00:00 (udev-worker)
  22947 ?        00:00:00 (udev-worker)
  22960 pts/0    00:00:00 ps
hari_ai@Harryfunlapcare:~$
```

- **Kill processes by name**

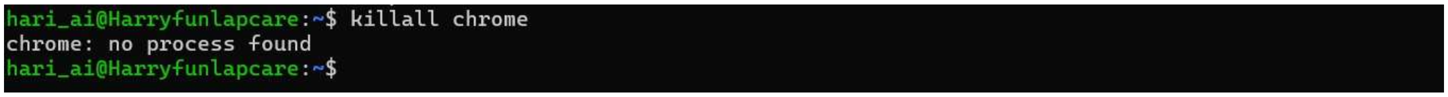
Command : killall bash



```
hari_ai@Harryfunlapcare:~$ pkill firefox
hari_ai@Harryfunlapcare:~$ kill 1877
-bash: kill: (1877) - No such process
hari_ai@Harryfunlapcare:~$ kill 1234
-bash: kill: (1234) - No such process
hari_ai@Harryfunlapcare:~$
```

- **Kill a process based on the process name**

Command : pkill bash

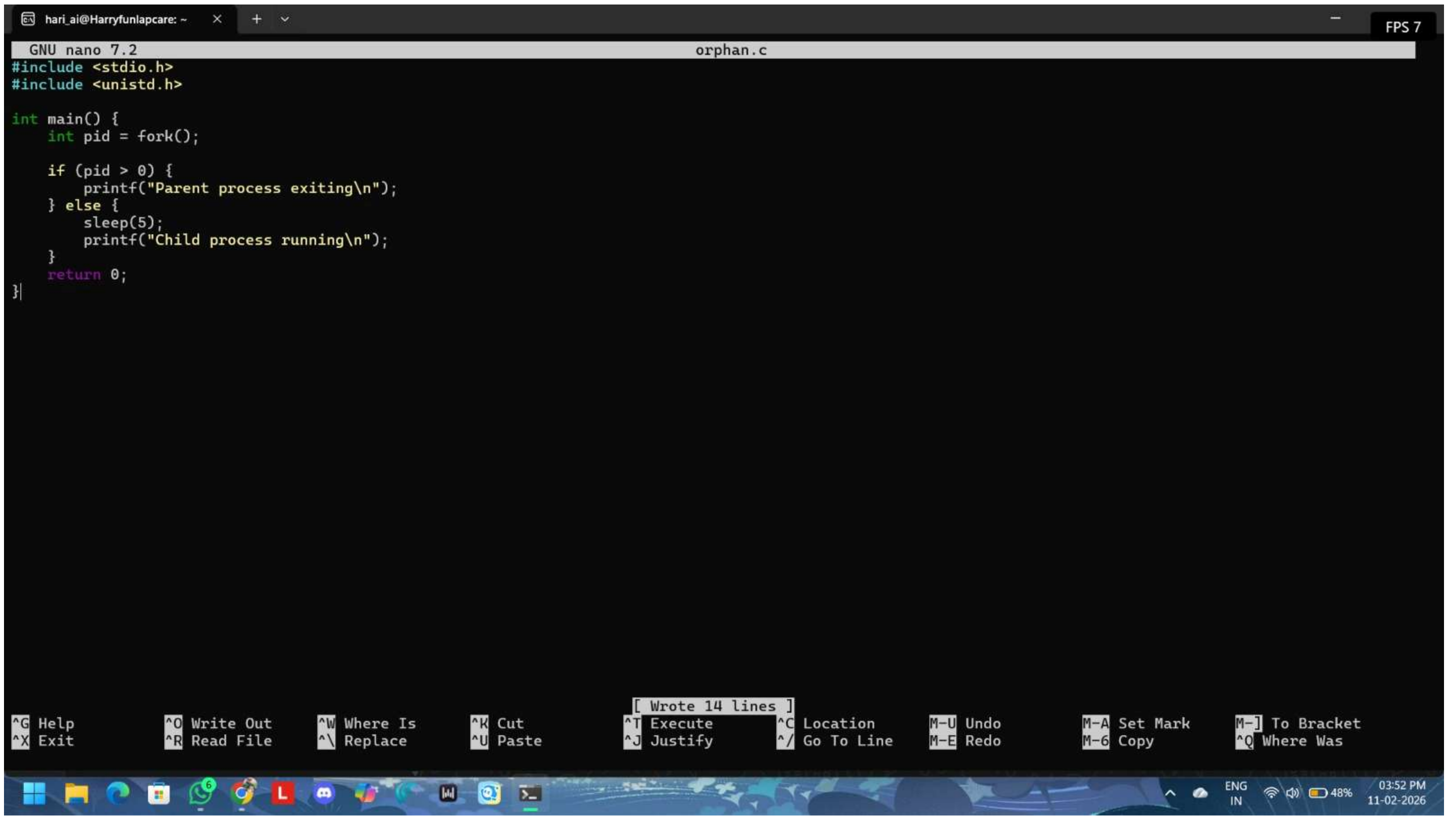


```
hari_ai@Harryfunlapcare:~$ killall chrome
chrome: no process found
hari_ai@Harryfunlapcare:~$
```

- PID & PPID

- **Kill a single process at a time with the given process ID**

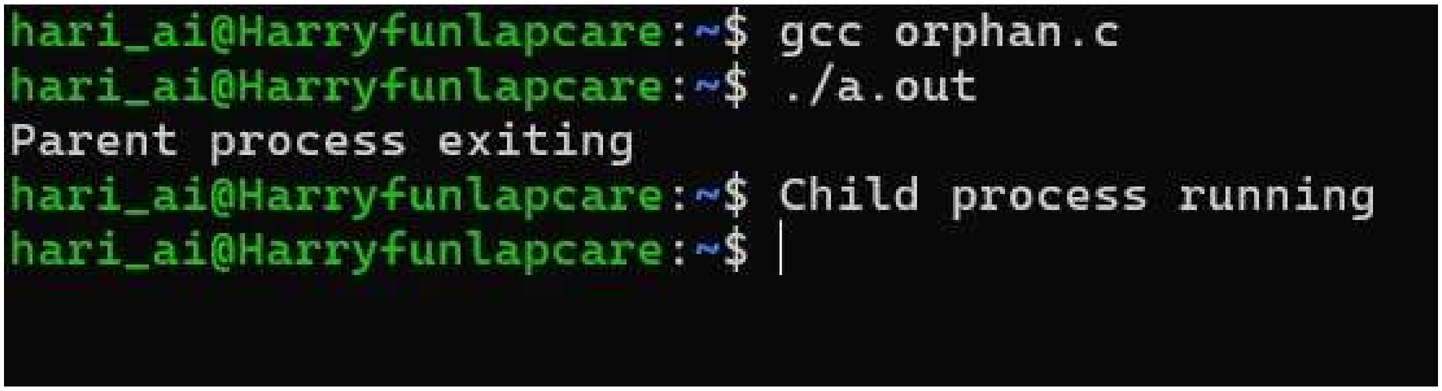
Q2. Write a program for process creation using C



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

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

    if (pid > 0) {
        printf("Parent process exiting\n");
    } else {
        sleep(5);
        printf("Child process running\n");
    }
    return 0;
}
```

```
                                  [ Wrote 14 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark  M-] To Bracket
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy      ^Q Where Was
```

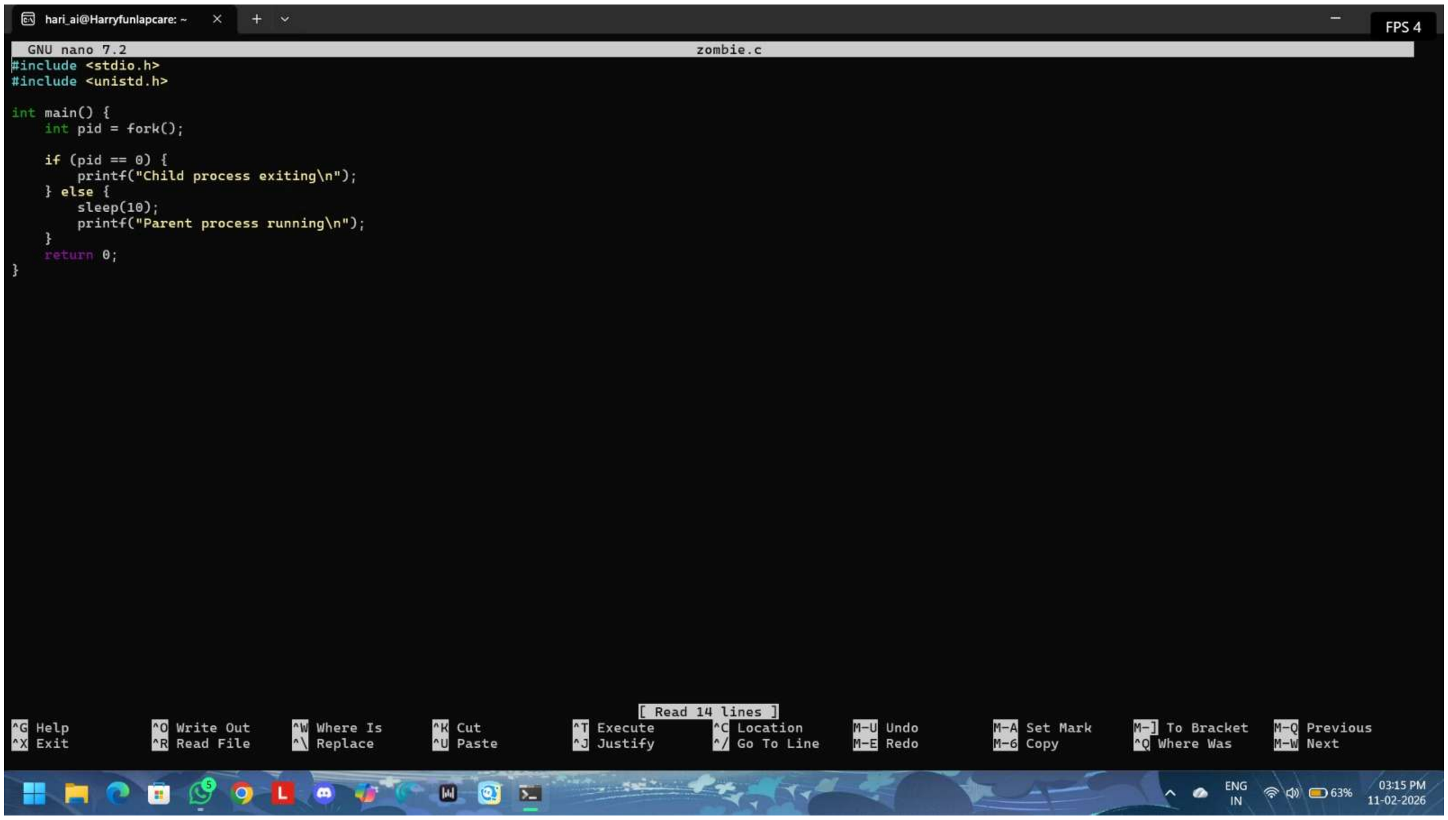
Orphan Process



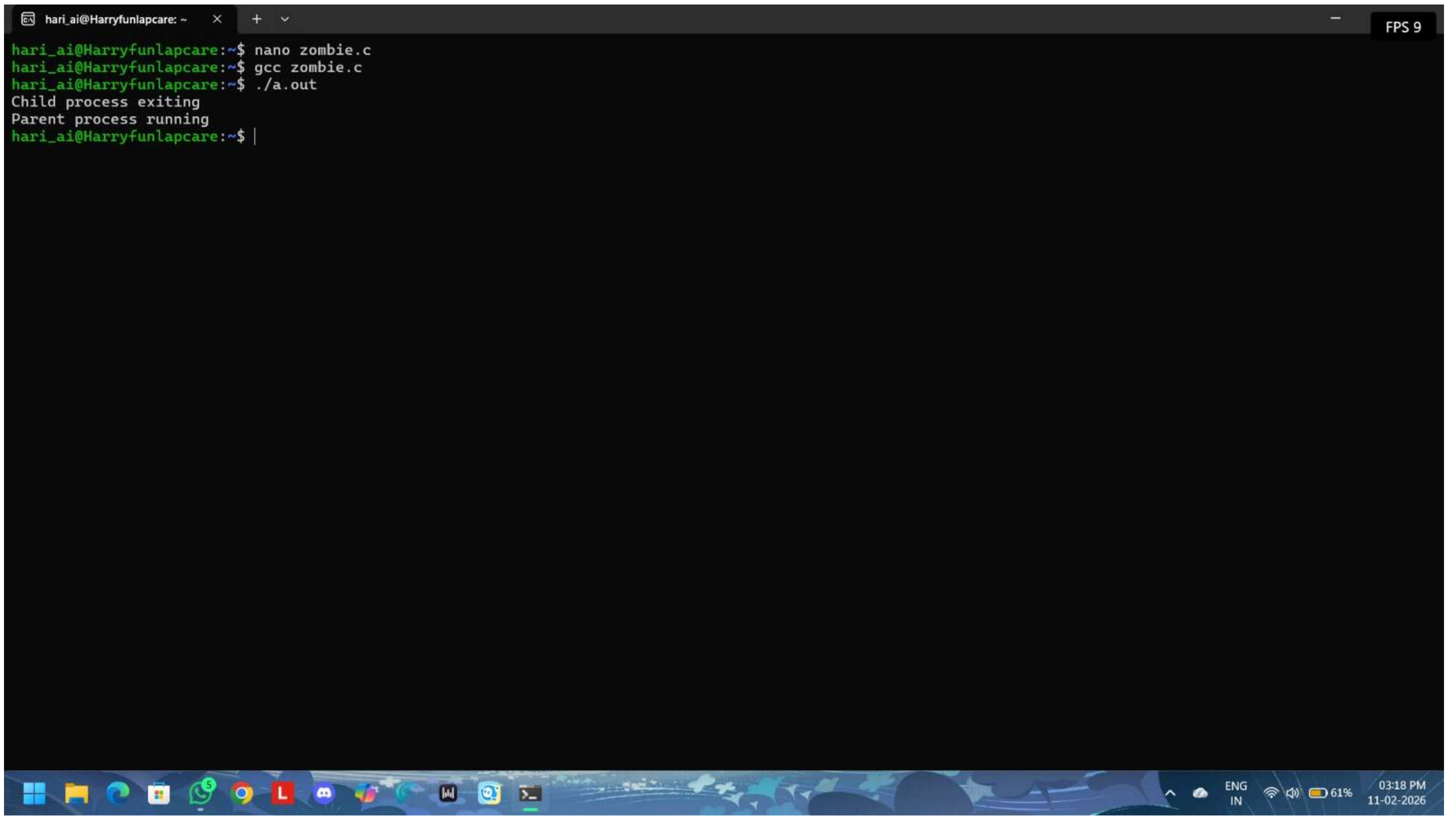
```
hari_ai@Harryfunlapcare:~$ gcc orphan.c
hari_ai@Harryfunlapcare:~$ ./a.out
Parent process exiting
hari_ai@Harryfunlapcare:~$ Child process running
hari_ai@Harryfunlapcare:~$ 
```

- PID & PPID

- **Zombie Process**



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

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

    if (pid == 0) {
        printf("Child process exiting\n");
    } else {
        sleep(10);
        printf("Parent process running\n");
    }
    return 0;
}

                                    [ Read 14 lines ]
^G Help      ^O Write Out  ^W Where Is  ^K Cut    ^T Execute  ^C Location    M-U Undo  M-A Set Mark  M-] To Bracket  M-Q Previous
^X Exit      ^R Read File  ^\ Replace   ^U Paste  ^J Justify  ^/ Go To Line  M-E Redo  M-6 Copy      ^Q Where Was    M-W Next
```

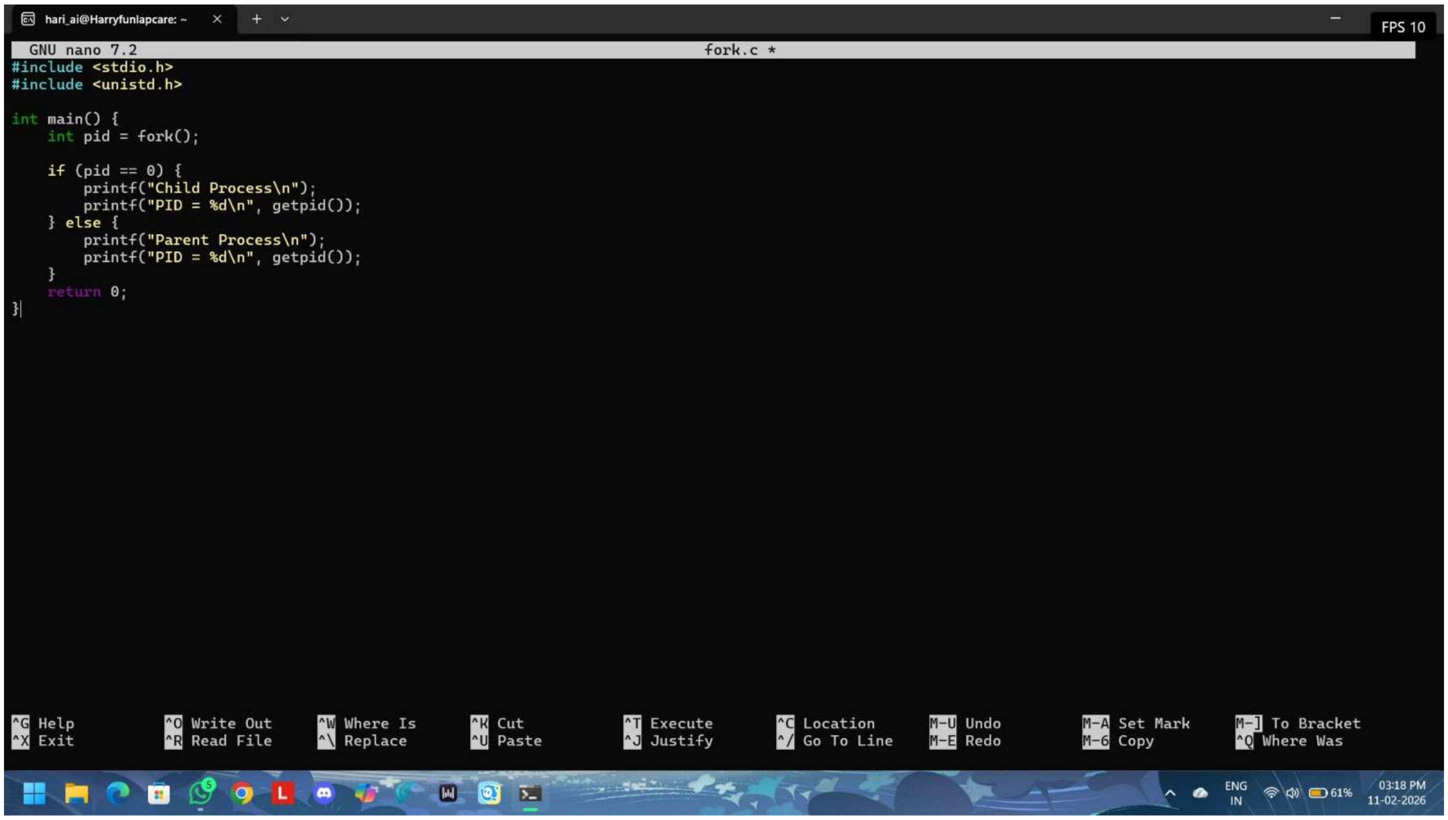


```
hari_ai@Harryfunlapcare:~$ nano zombie.c
hari_ai@Harryfunlapcare:~$ gcc zombie.c
hari_ai@Harryfunlapcare:~$ ./a.out
Child process exiting
Parent process running
hari_ai@Harryfunlapcare:~$ |
```

- PID & PPID

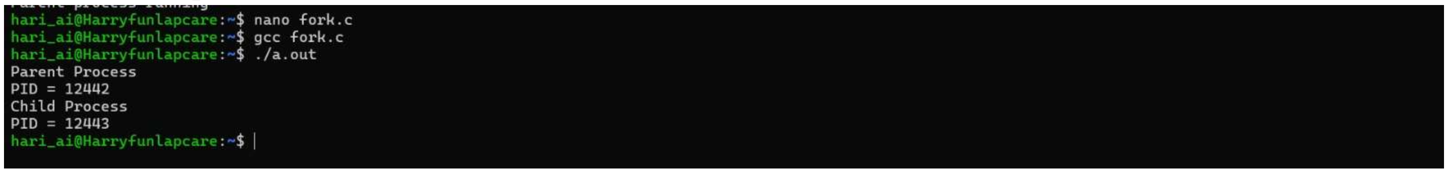
Q3. Create the process using fork () system call.

- **Child Process creation**



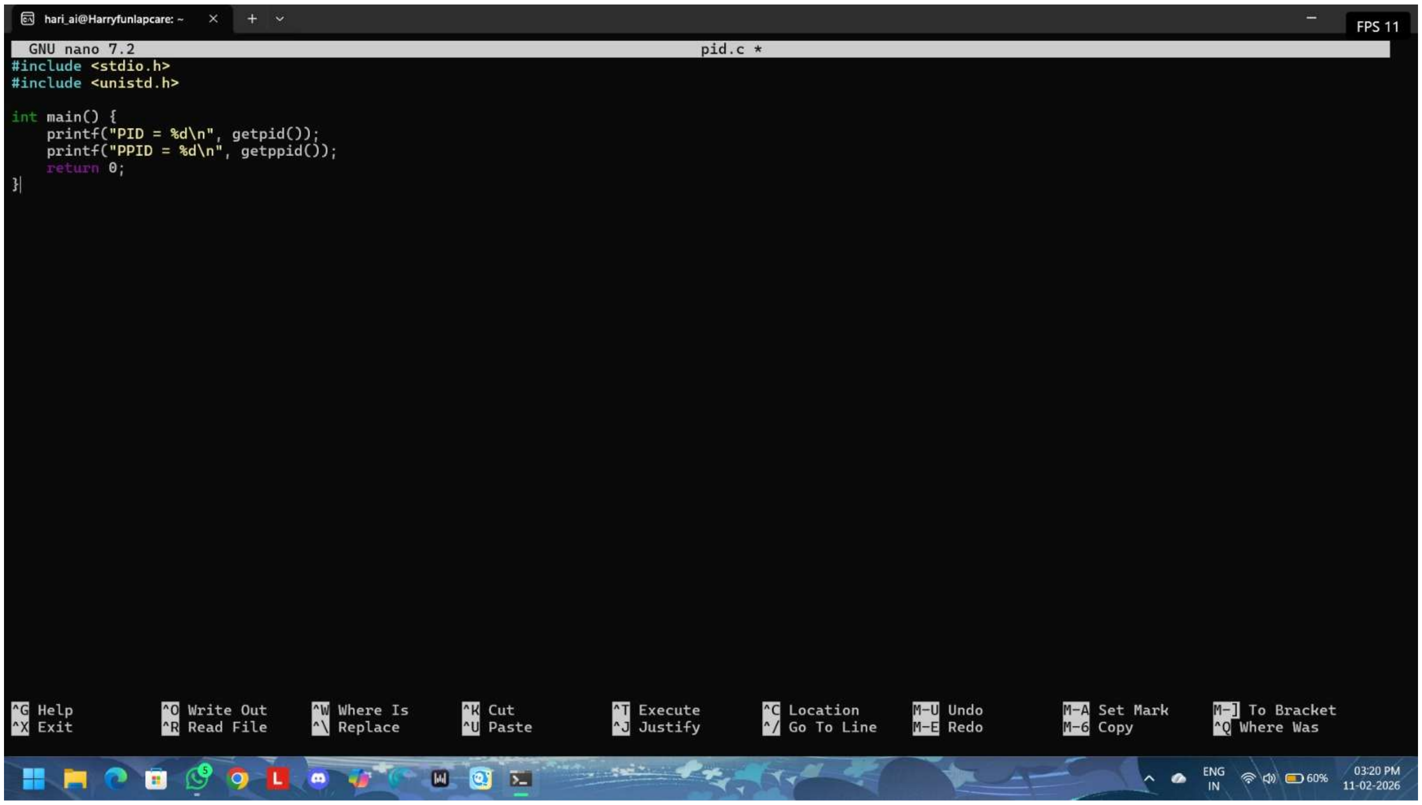
```
GNU nano 7.2                                fork.c *
#include <stdio.h>
#include <unistd.h>

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

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

- **Parent process creation**



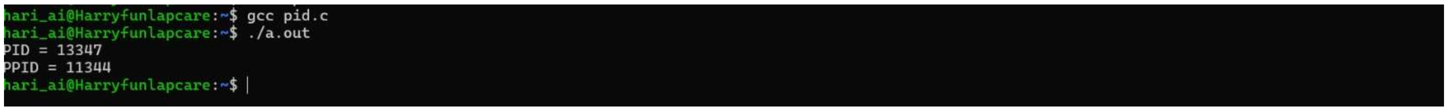
```
hari_ai@Harryfunlapcare:~$ nano fork.c
hari_ai@Harryfunlapcare:~$ gcc fork.c
hari_ai@Harryfunlapcare:~$ ./a.out
Parent Process
PID = 12442
Child Process
PID = 12443
hari_ai@Harryfunlapcare:~$ |
```

- PID & PPID



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

int main() {
    printf("PID = %d\n", getpid());
    printf("PPID = %d\n", getppid());
    return 0;
}
```

hari_ai@Harryfunlapcare: ~
FPS 11
^G Help ^X Exit ^O Write Out ^R Read File ^W Where Is ^\ Replace ^K Cut ^U Paste ^T Execute ^J Justify ^C Location ^/ Go To Line M-U Undo M-E Redo M-A Set Mark M-6 Copy M-] To Bracket ^Q Where Was
ENG IN 60% 03:20 PM 11-02-2026



```
hari_ai@Harryfunlapcare:~$ gcc pid.c
hari_ai@Harryfunlapcare:~$ ./a.out
PID = 13347
PPID = 11344
hari_ai@Harryfunlapcare:~$ |
```