

- 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(Up
   6 ?
  52 ?
 101 ?
 111 ?
 122 ?
 171 ?
 172 ?
 179 ?
 184 hvc0
 195 ?
 204 ttyl
 205 ?
 289 pts/1
 346 ?
 348 ?
 388 pts/1
 7168 ?
 22860 ?
 22861 ?
 22866 pts/0
 22946 ?
 22947 ?
 22960 pts/0
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

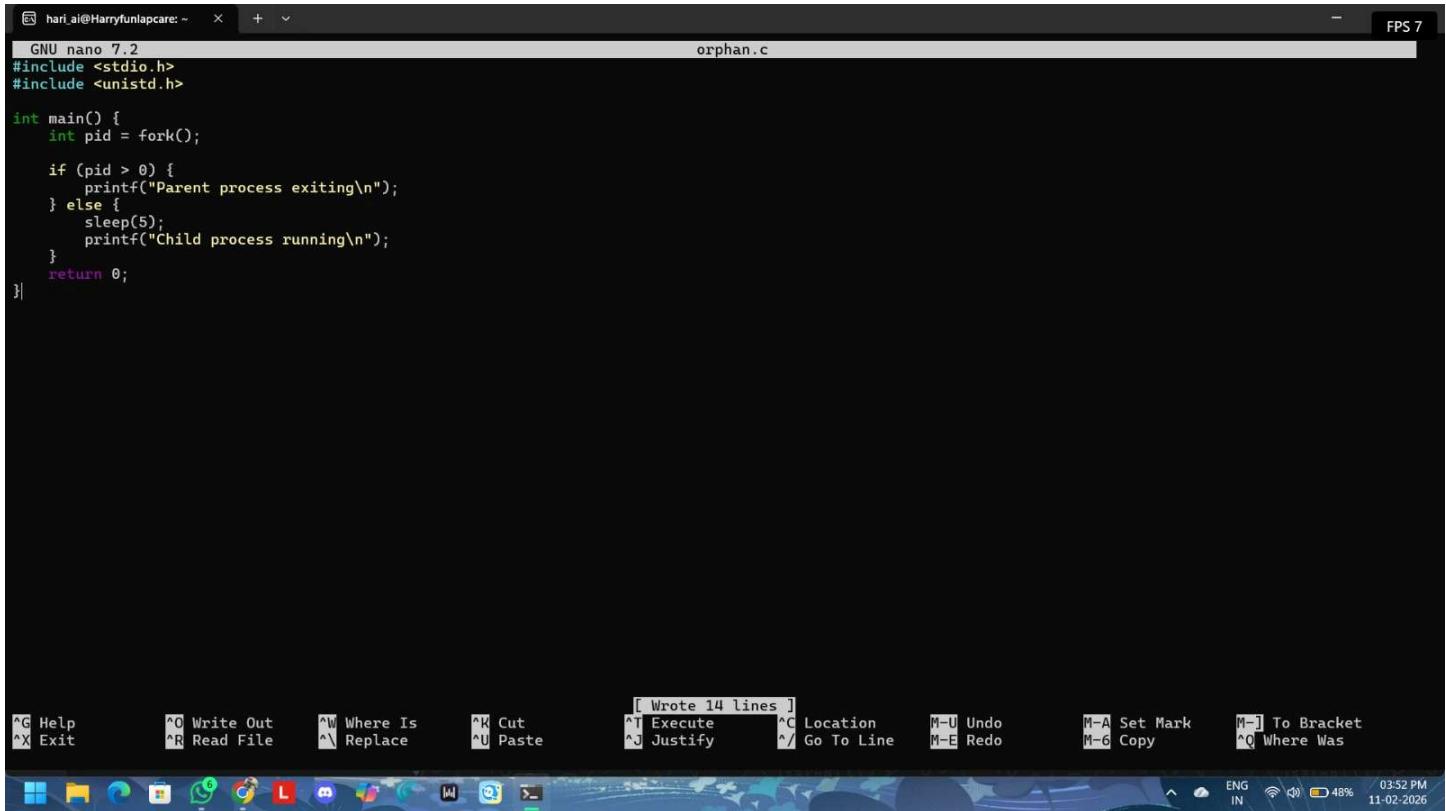
Command : pkill bash

```

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



The screenshot shows a terminal window titled "hari_ai@Harryfunlapcare: ~". The file being edited is "orphan.c". The code is as follows:

```

GNU nano 7.2
orphan.c

#include <stdio.h>
#include <unistd.h>

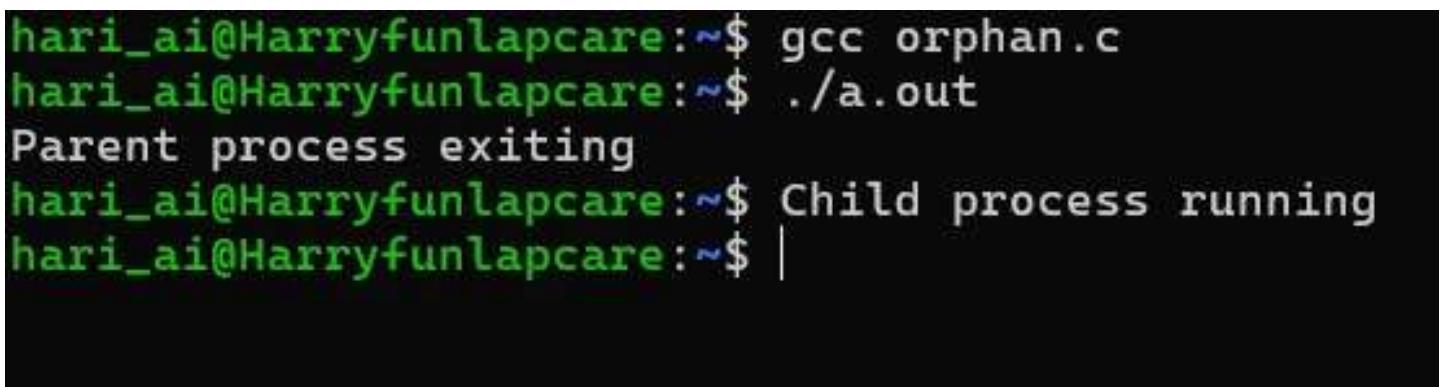
int main() {
    int pid = fork();

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

```

The terminal window includes a status bar at the bottom with various keyboard shortcuts and system information like battery level (48%) and date/time (11-02-2026).

Orphan Process



The screenshot shows a terminal window with the following session:

```

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

The screenshot shows a terminal window titled "hari_ai@Harryfunlapcare: ~". The code in the terminal is:

```
GNU nano 7.2
#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;
}
```

The status bar at the bottom of the terminal window displays various system information, including the date and time (03:15 PM 11-02-2026), battery level (63%), and network connectivity.

The screenshot shows a terminal window titled "hari_ai@Harryfunlapcare: ~". The command history and output are:

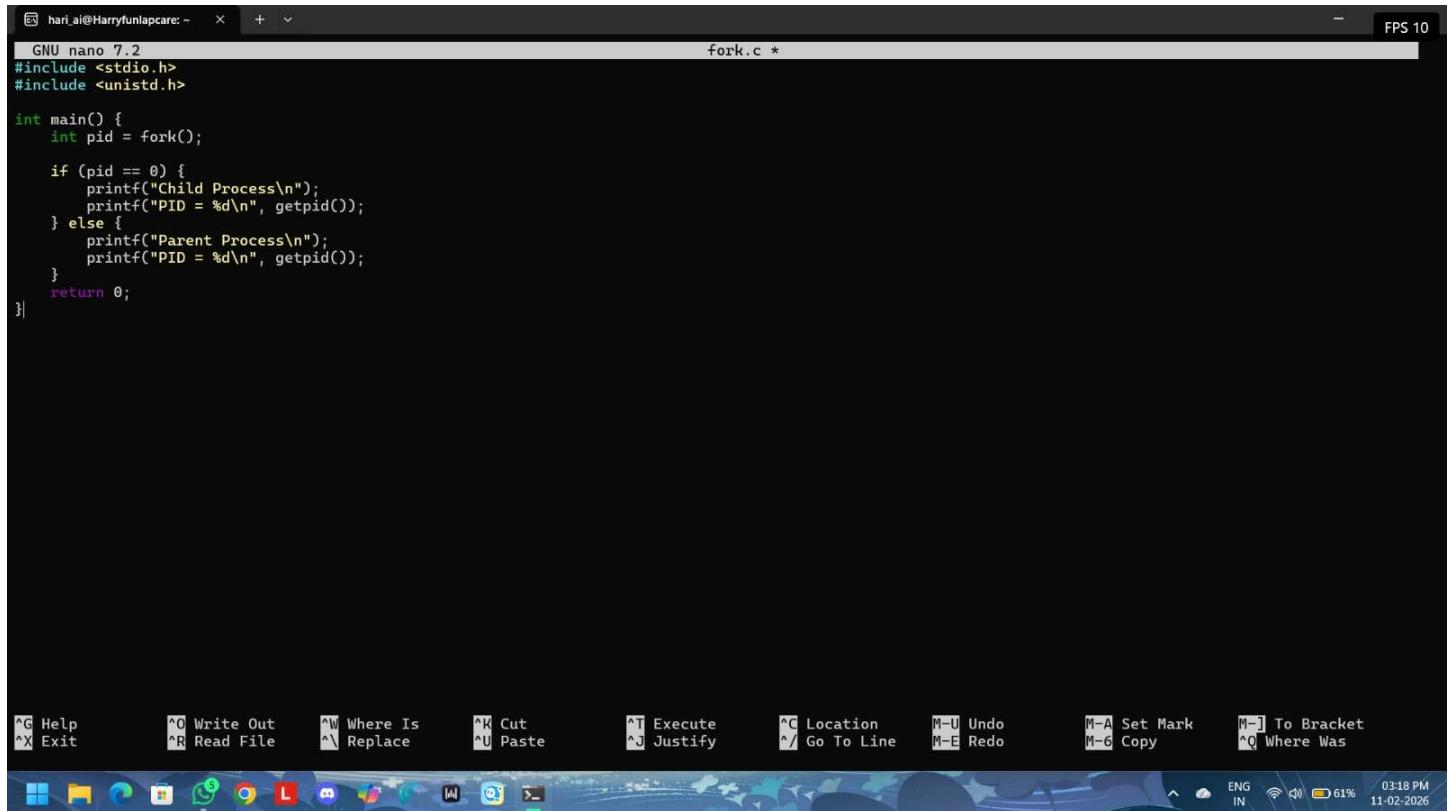
```
hari_ai@Harryfunlapcare:~$ nano zombie.c
hari_ai@Harryfunlapcare:~$ gcc zombie.c
hari_ai@Harryfunlapcare:~$ ./a.out
Child process exiting
Parent process running
hari_ai@Harryfunlapcare:~$ |
```

The status bar at the bottom of the terminal window displays various system information, including the date and time (03:18 PM 11-02-2026), battery level (61%), and network connectivity.

- PID & PPID

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

- Child Process creation



The screenshot shows a terminal window titled "fork.c *". The code inside the window is:

```

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;
}

```

The terminal window includes a menu bar at the top with options like "File", "Edit", "View", "Insert", "Format", "Search", "Tools", and "Help". Below the menu is a toolbar with icons for Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, Copy, To Bracket, and Where Was. At the bottom of the window is a status bar showing system information: ENG IN, 61%, 03:18 PM, and 11-02-2026.

- Parent process creation



The screenshot shows a terminal window with the following command and output:

```

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

A screenshot of a Linux desktop environment. In the foreground, there is a terminal window titled "pid.c *" showing C code to print the current process ID (PID) and its parent process ID (PPID). Below the terminal is a menu bar with various options like Help, Write Out, Read File, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, Copy, To Bracket, Where Is, Replace, and Where Was. At the bottom, there is a dock with icons for various applications and a system tray showing battery status (60%), signal strength, and the date and time (03:20 PM, 11-02-2026).

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

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

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