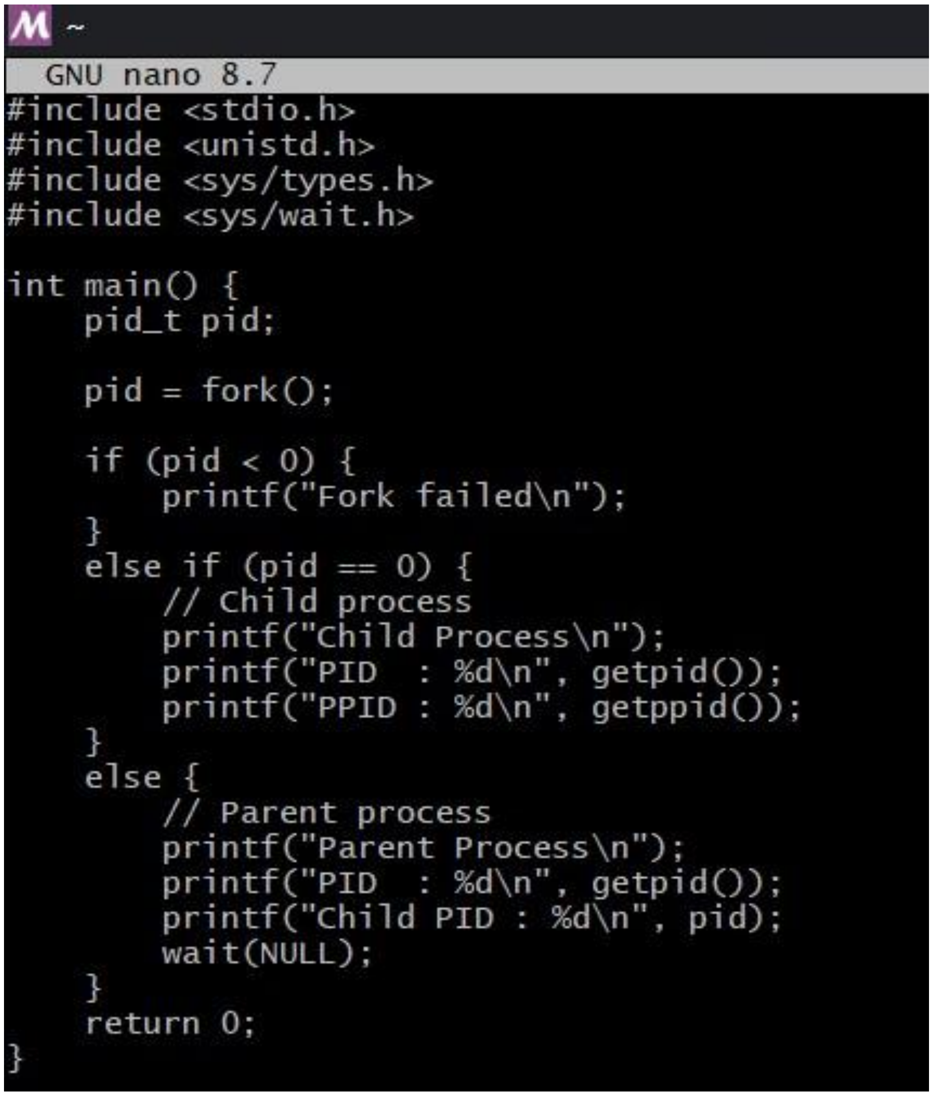
1.Adam is working in an IT company. He has been given a task to reduce the load of a system by killing some of the processes running in the LINUX operating system. Which commands will he use to complete the given task with the help of the following operation?

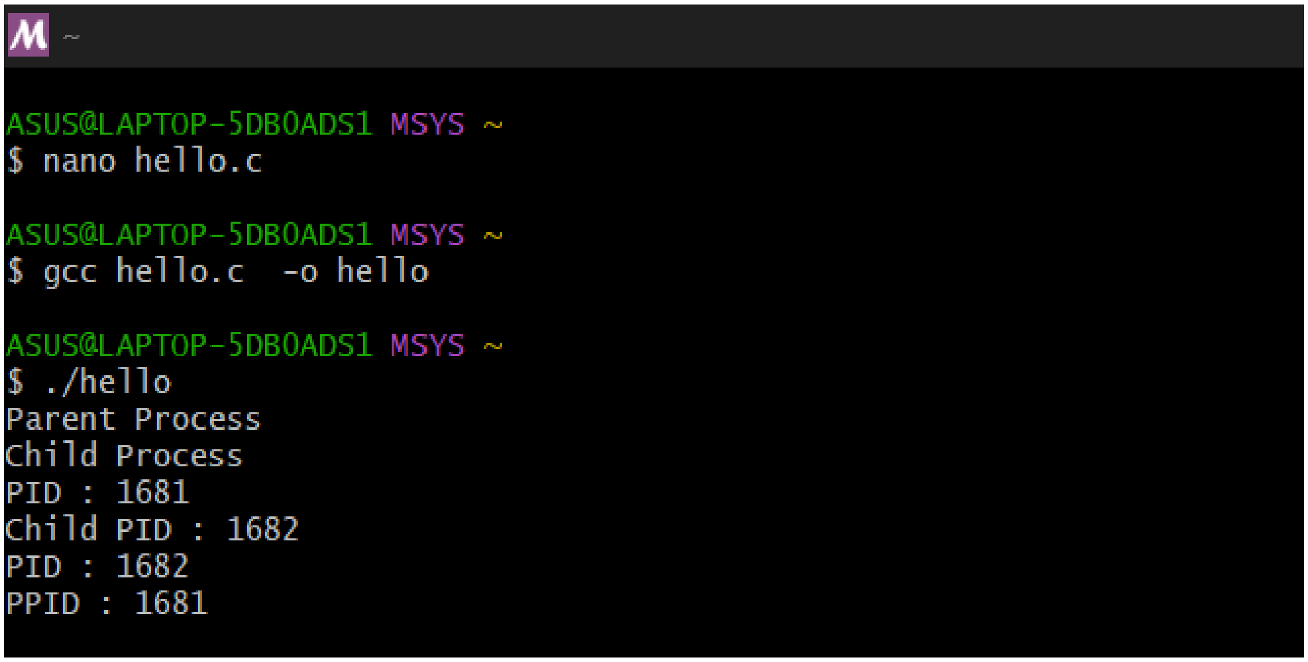
(i) Kill processes by name
(ii) Kill a process based on the process name
(iii) Kill a single process at a time with the given process ID

CODE:



```
~
  GNU nano 8.7
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main() {
    pid_t pid;

    pid = fork();

    if (pid < 0) {
        printf("Fork failed\n");
    }
    else if (pid == 0) {
        // Child process
        printf("Child Process\n");
        printf("PID  : %d\n", getpid());
        printf("PPID : %d\n", getppid());
    }
    else {
        // Parent process
        printf("Parent Process\n");
        printf("PID  : %d\n", getpid());
        printf("Child PID : %d\n", pid);
        wait(NULL);
    }
    return 0;
}
```

OUTPUT:

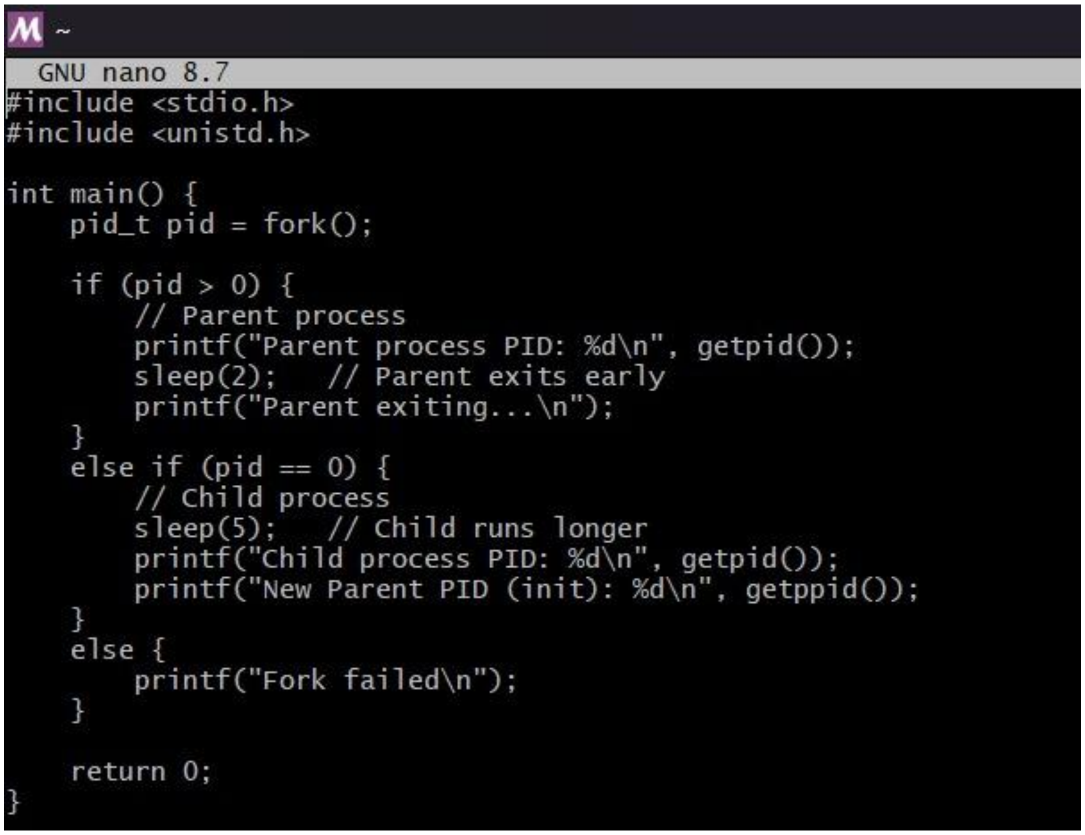


```
ASUS@LAPTOP-5DB0ADS1 MSYS ~
$ nano hello.c

ASUS@LAPTOP-5DB0ADS1 MSYS ~
$ gcc hello.c  -o hello

ASUS@LAPTOP-5DB0ADS1 MSYS ~
$ ./hello
Parent Process
Child Process
PID : 1681
Child PID : 1682
PID : 1682
PPID : 1681
```

2. Write a program for process creation using C

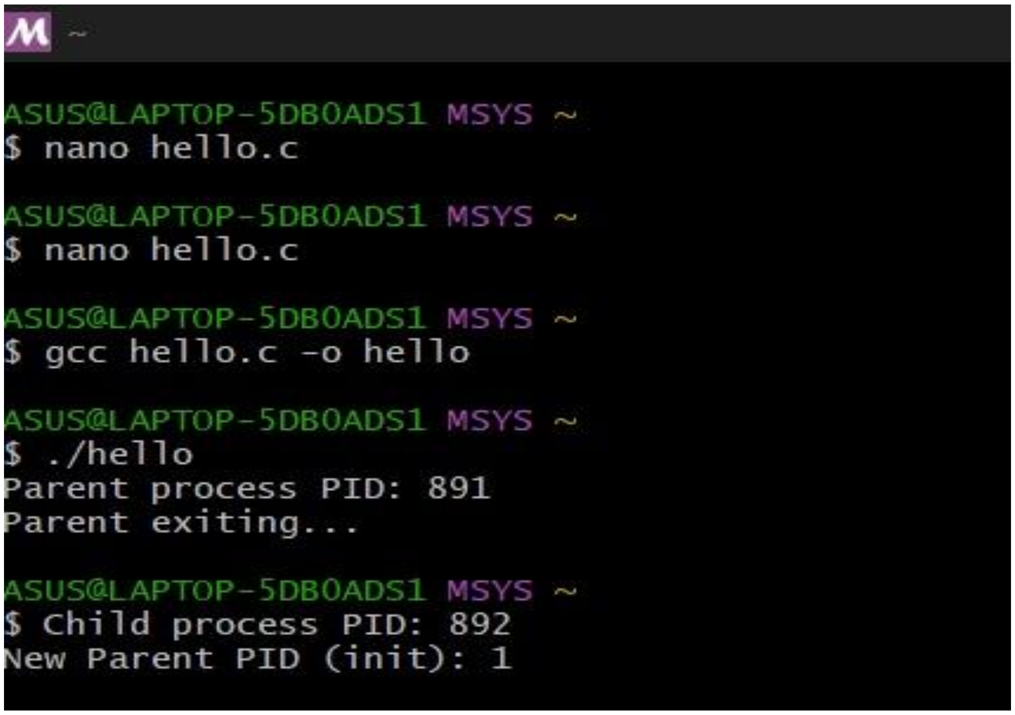
(i) Orphan Process

CODE:



```
  GNU nano 8.7
#include <stdio.h>
#include <unistd.h>

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

    if (pid > 0) {
        // Parent process
        printf("Parent process PID: %d\n", getpid());
        sleep(2);   // Parent exits early
        printf("Parent exiting...\n");
    }
    else if (pid == 0) {
        // Child process
        sleep(5);   // Child runs longer
        printf("Child process PID: %d\n", getpid());
        printf("New Parent PID (init): %d\n", getppid());
    }
    else {
        printf("Fork failed\n");
    }

    return 0;
}
```

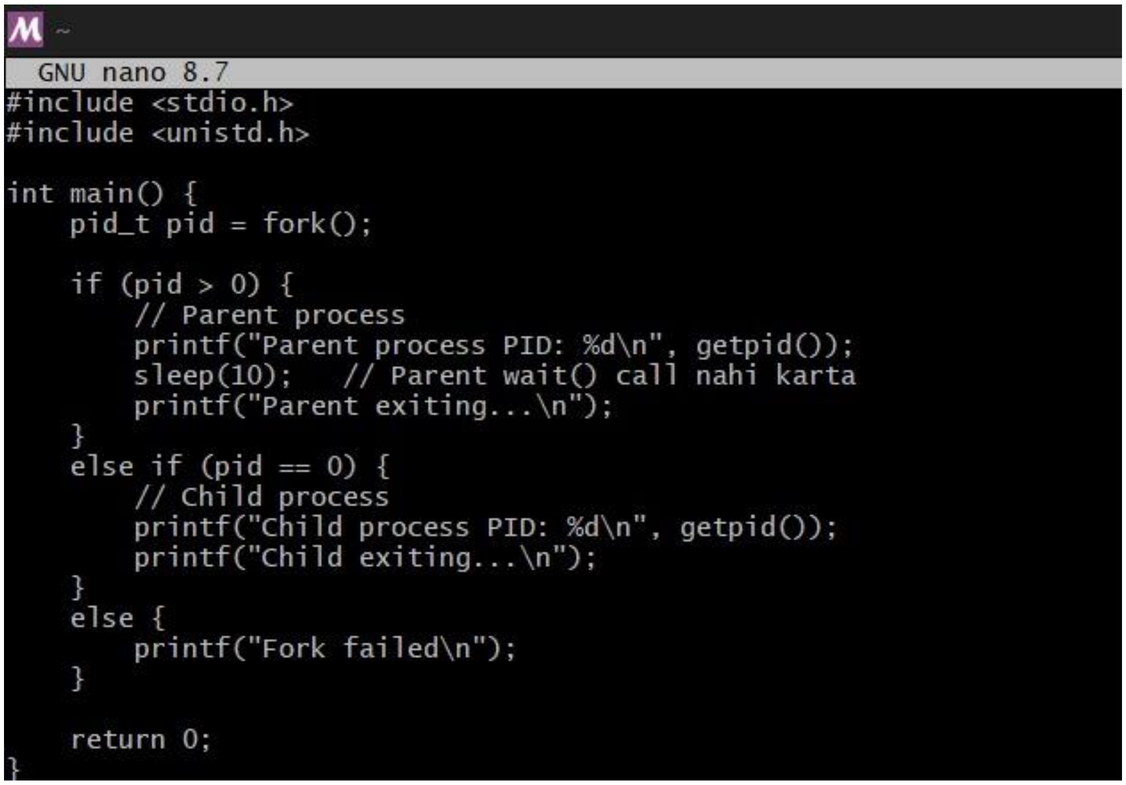
OUTPUT:



```
ASUS@LAPTOP-5DB0ADS1 MSYS ~
$ nano hello.c

ASUS@LAPTOP-5DB0ADS1 MSYS ~
$ nano hello.c

ASUS@LAPTOP-5DB0ADS1 MSYS ~
$ gcc hello.c -o hello

ASUS@LAPTOP-5DB0ADS1 MSYS ~
$ ./hello
Parent process PID: 891
Parent exiting...

ASUS@LAPTOP-5DB0ADS1 MSYS ~
$ Child process PID: 892
New Parent PID (init): 1
```

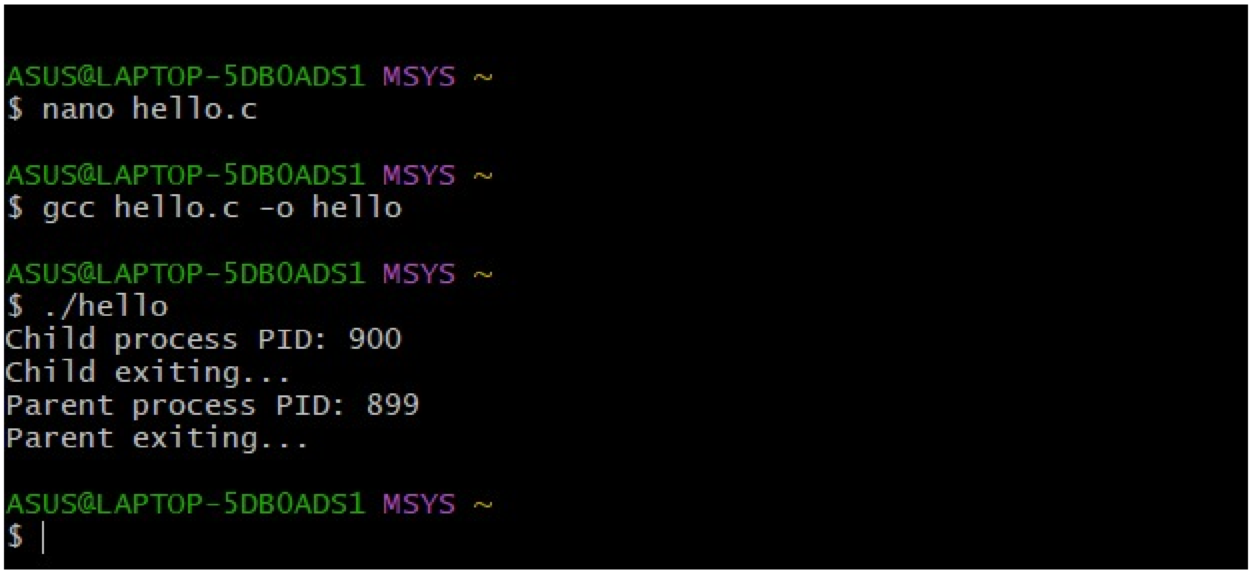
(ii) Zombine Process

CODE:



```
  GNU nano 8.7
#include <stdio.h>
#include <unistd.h>

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

    if (pid > 0) {
        // Parent process
        printf("Parent process PID: %d\n", getpid());
        sleep(10);   // Parent wait() call nahi karta
        printf("Parent exiting...\n");
    }
    else if (pid == 0) {
        // Child process
        printf("Child process PID: %d\n", getpid());
        printf("Child exiting...\n");
    }
    else {
        printf("Fork failed\n");
    }

    return 0;
}
```

OUTPUT:

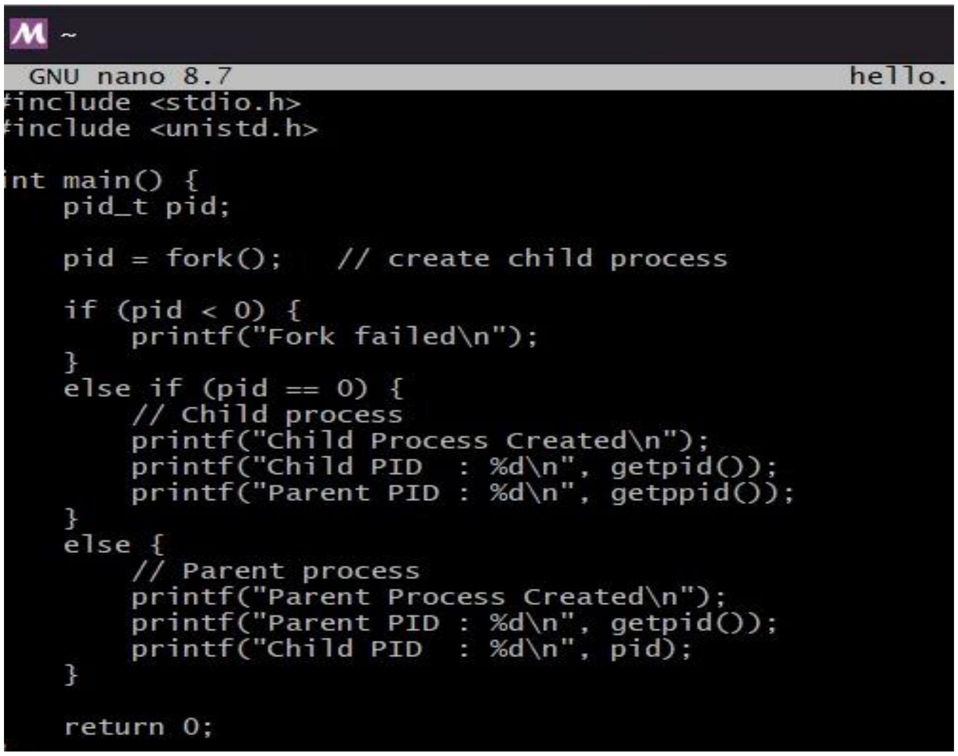


```
ASUS@LAPTOP-5DB0ADS1 MSYS ~
$ nano hello.c

ASUS@LAPTOP-5DB0ADS1 MSYS ~
$ gcc hello.c -o hello

ASUS@LAPTOP-5DB0ADS1 MSYS ~
$ ./hello
Child process PID: 900
Child exiting...
Parent process PID: 899
Parent exiting...

ASUS@LAPTOP-5DB0ADS1 MSYS ~
$ |
```

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

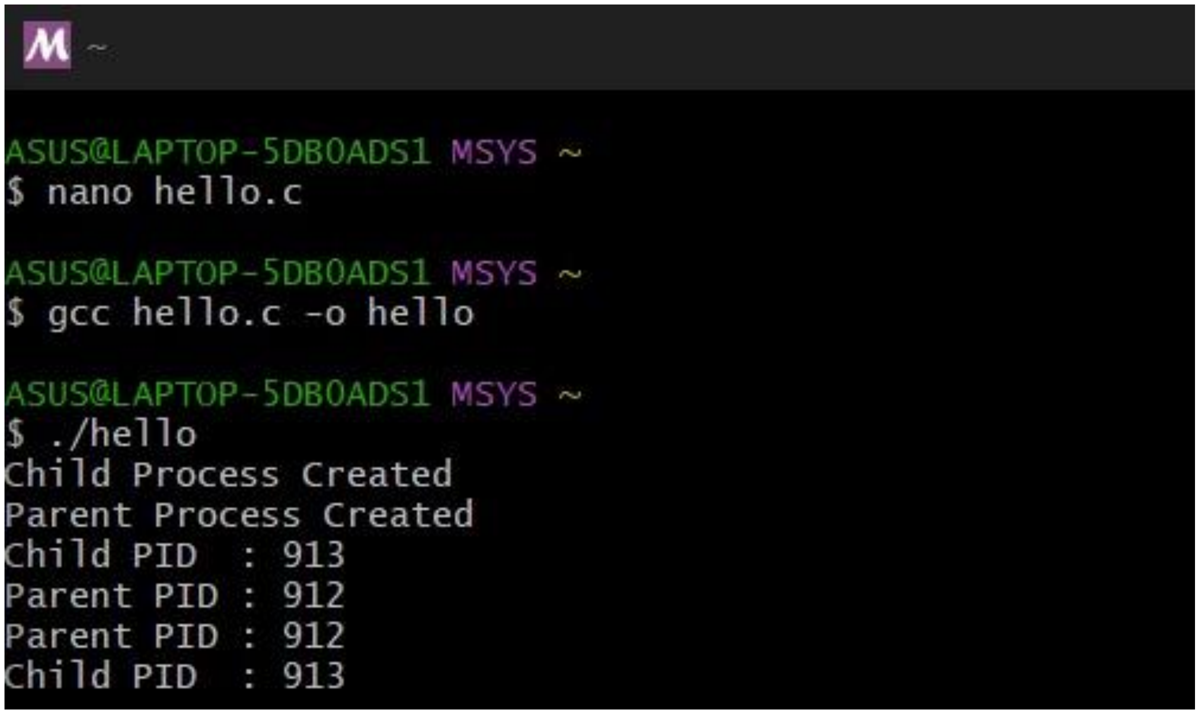
(i) Child Process creation
(ii) Parent process creation
(iii) PPID and PID

CODE:



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

int main() {
    pid_t pid;

    pid = fork();   // create child process

    if (pid < 0) {
        printf("Fork failed\n");
    }
    else if (pid == 0) {
        // Child process
        printf("Child Process Created\n");
        printf("Child PID  : %d\n", getpid());
        printf("Parent PID : %d\n", getppid());
    }
    else {
        // Parent process
        printf("Parent Process Created\n");
        printf("Parent PID : %d\n", getpid());
        printf("Child PID  : %d\n", pid);
    }

    return 0;
```

OUTPUT:



```
ASUS@LAPTOP-5DB0ADS1 MSYS ~
$ nano hello.c

ASUS@LAPTOP-5DB0ADS1 MSYS ~
$ gcc hello.c -o hello

ASUS@LAPTOP-5DB0ADS1 MSYS ~
$ ./hello
Child Process Created
Parent Process Created
Child PID  : 913
Parent PID : 912
Parent PID : 912
Child PID  : 913
```