Practical NO 4

Write a program for process creation using C

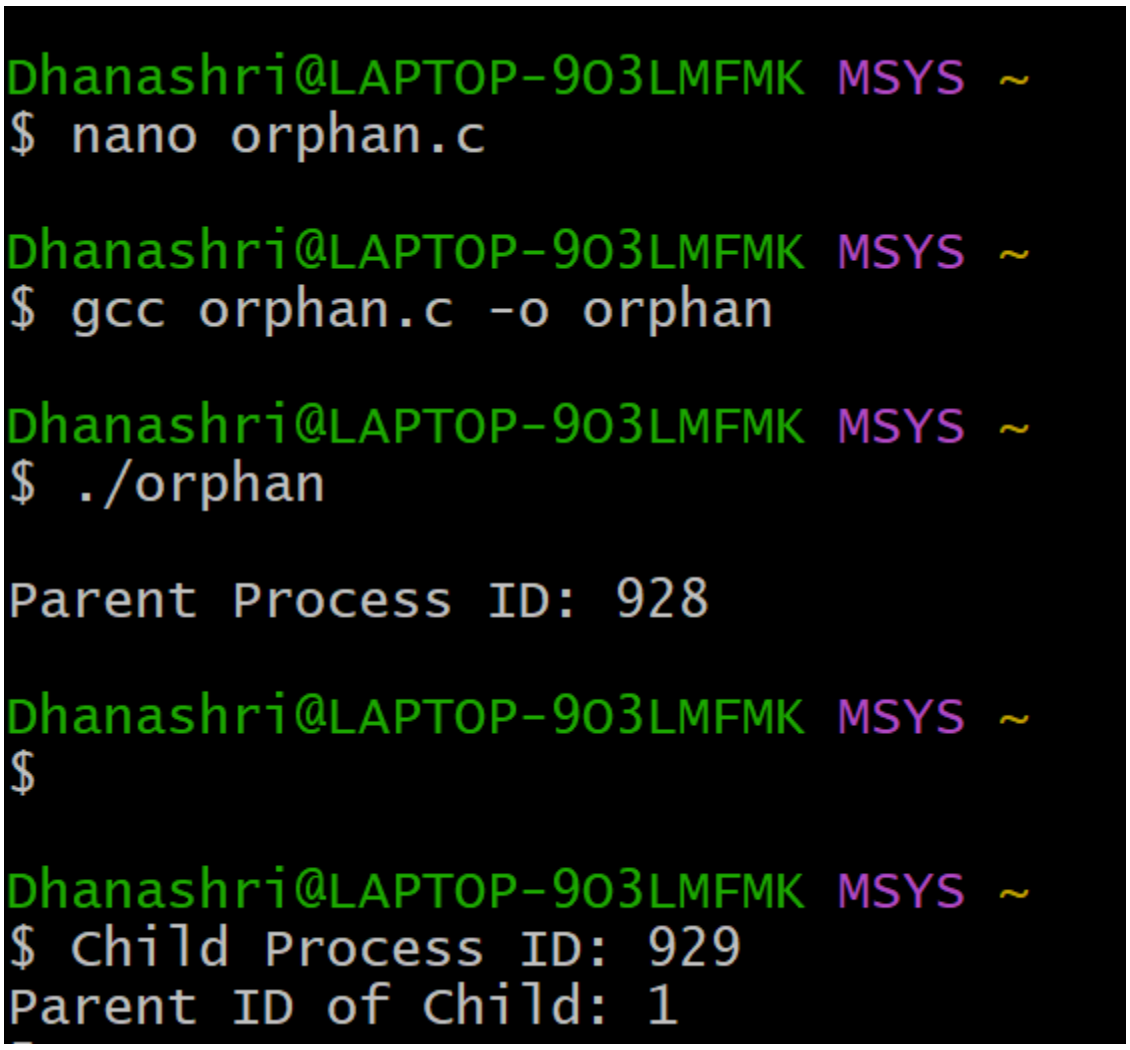
●Orphan Process

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

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

    if (pid > 0) {
        printf("Parent Process ID: %d\n", getpid());
        sleep(2);
    } else {
        sleep(5);
        printf("Child Process ID: %d\n", getpid());
        printf("Parent ID of Child: %d\n", getppid());
    }
    return 0;
}
```

Output


Dhanashri@LAPTOP-9O3LMFMK MSYS ~
$ nano orphan.c
Dhanashri@LAPTOP-9O3LMFMK MSYS ~
$ gcc orphan.c -o orphan
Dhanashri@LAPTOP-9O3LMFMK MSYS ~
$ ./orphan
Parent Process ID: 928
Dhanashri@LAPTOP-9O3LMFMK MSYS ~
$
Dhanashri@LAPTOP-9O3LMFMK MSYS ~
$ Child Process ID: 929
Parent ID of Child: 1

●**Zombie Process**

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

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

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

Output

```
Dhanashri@LAPTOP-9O3LMFMK MSYS ~
$ nano zombie.c

Dhanashri@LAPTOP-9O3LMFMK MSYS ~
$ gcc zombie.c -o zombie

Dhanashri@LAPTOP-9O3LMFMK MSYS ~
$ ./zombie
Child process exiting
Parent process running

Dhanashri@LAPTOP-9O3LMFMK MSYS ~
$ ps -el
      PID    PPID    PGID     WINPID   TTY         UID    STIME COMMAND
      765     764     765      15116  pty0      197609 22:20:11 /usr/bin/bash
      764       1     764      17932  ?         197609 22:20:11 /usr/bin/mintty
      791     765     791      21116  pty0      197609 22:23:34 /usr/bin/ps

Dhanashri@LAPTOP-9O3LMFMK MSYS ~
$
```

Create the process using fork () system call.

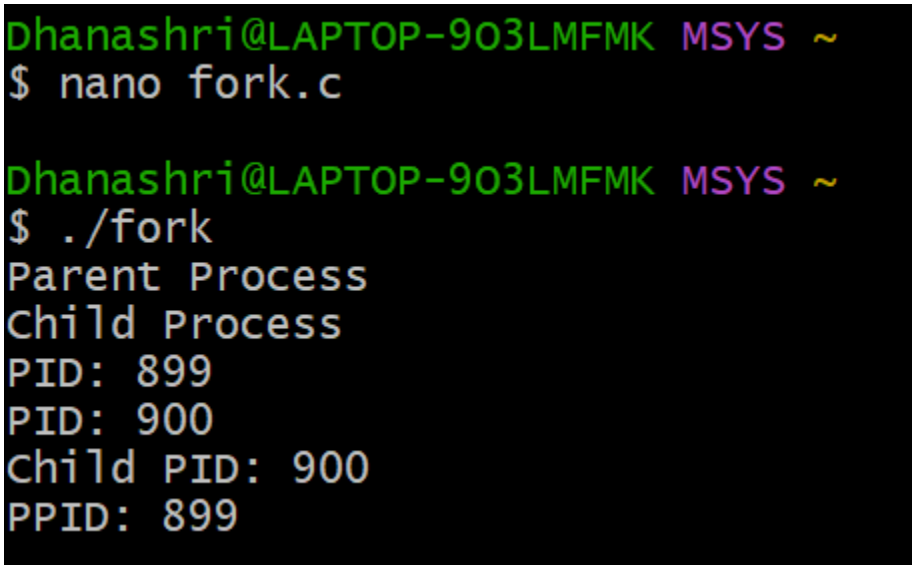
- **Child Process creation**
- **Parent process creation**
- **PPID and PID**

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

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

    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);
    }
    return 0;
}
```

Output


Dhanashri@LAPTOP-9O3LMFMK MSYS ~
$ nano fork.c
Dhanashri@LAPTOP-9O3LMFMK MSYS ~
$ ./fork
Parent Process
Child Process
PID: 899
PID: 900
Child PID: 900
PPID: 899