

Experiment 10

Aim: To create an Orphan process and a Zombie Process.

- **An orphan process** is a process whose parent has terminated before it finishes its execution.
- **A zombie process** is a process that has completed execution but still has an entry in the process table.

Program to create an orphan process

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

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

    if (p == 0) {
        // Child process
        sleep(5);
        printf("I am child having PID: %d\n", getpid());
        printf("My parent PID is: %d\n", getppid());
    } else {
        // Parent process
        printf("My child PID is: %d\n", p);
    }

    return 0;
}
```

```
localhost:~# vi orphan.c
// orphan.c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

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

    if (p == 0) {
        // Child process
        sleep(5);
        printf("I am child having PID: %d\n", getpid());
        printf("My parent PID is: %d\n", getppid());
    } else {
        // Parent process
        printf("I am parent having PID: %d\n", getpid());
        printf("My child PID is: %d\n", p);
    }

    return 0;
}

localhost:~# gcc -o orphan orphan.c
localhost:~# ./orphan
I am parent having PID: 81
My child PID is: 82
localhost:~# I am child having PID: 82
My parent PID is: 1
```

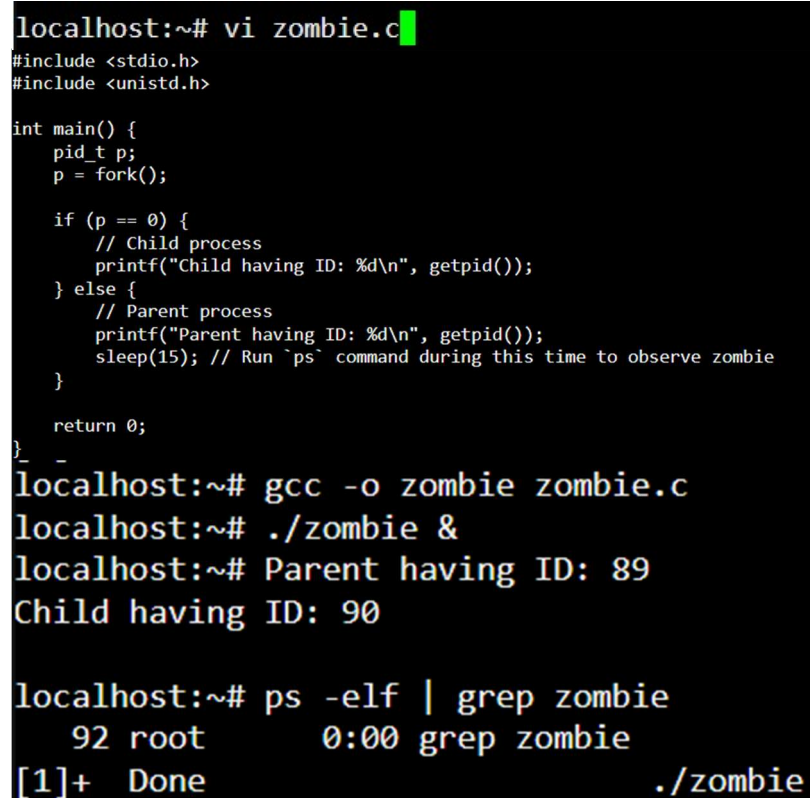
Program to create a zombie process

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

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

    if (p == 0) {
        // Child process
        printf("Child having ID: %d\n", getpid());
    } else {
        // Parent process
        printf("Parent having ID: %d\n", getpid());
        sleep(15); // Run `ps` command during this time to observe zombie
    }

    return 0;
}
```



The terminal screenshot shows the following sequence of commands and outputs:

```
localhost:~# vi zombie.c
#include <stdio.h>
#include <unistd.h>

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

    if (p == 0) {
        // Child process
        printf("Child having ID: %d\n", getpid());
    } else {
        // Parent process
        printf("Parent having ID: %d\n", getpid());
        sleep(15); // Run `ps` command during this time to observe zombie
    }

    return 0;
}
localhost:~# gcc -o zombie zombie.c
localhost:~# ./zombie &
localhost:~# Parent having ID: 89
Child having ID: 90

localhost:~# ps -elf | grep zombie
  92 root      0:00 grep zombie
[1]+  Done                  ./zombie
```