Zombie and Orphan Processes in C

- Difficulty Level : Medium
- Last Updated : 29 May, 2017

Prerequisite: fork() in C

Zombie Process:

A process which has finished the execution but still has entry in the process table to report to its parent process is known as a zombie process. A child process always first becomes a zombie before being removed from the process table. The parent process reads the exit status of the child process which reaps off the child process entry from the process table.
In the following code, the child finishes its execution using exit() system call while the parent sleeps for 50 seconds, hence doesn't call wait() and the child process's entry still exists in the process table.

```
// A C program to demonstrate Zombie Process.

// Child becomes Zombie as parent is sleeping

// when child process exits.

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

    // Fork returns process id

    // in parent process

    pid_t child_pid = fork();



    // Parent process

    if (child_pid > 0)
```

```
        sleep(50);


    // Child process
    else
        exit(0);


    return 0;
}
```

Note that the above code may not work with online compiler as fork() is disabled.

Orphan Process:

A process whose parent process no more exists i.e. either finished or terminated without waiting for its child process to terminate is called an orphan process.

In the following code, parent finishes execution and exits while the child process is still executing and is called an orphan process now.
However, the orphan process is soon adopted by init process, once its parent process dies.

```
// A C program to demonstrate Orphan Process.
// Parent process finishes execution while the
// child process is running. The child process
// becomes orphan.
#include<stdio.h>
```

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

int main()
{
    // Create a child process
    int pid = fork();

    if (pid > 0)
        printf("in parent process");

// Note that pid is 0 in child process
// and negative if fork() fails
else if (pid == 0)
{
    sleep(30);
    printf("in child process");
}

return 0;
```

```
}
```

Note that the above code may not work with online compilers as fork() is disabled.

Related :
Any idea What are Zombies in Operating System?
Zombie Processes and their Prevention
This article is contributed by **Pranjal Mathur**.If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.
Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above