MIT World Peace University

Operating Systems
Second Year B. Tech, Semester 3

MultiProcess Programming Using Fork - Zombie and Orphan Processes

Assignment 2
Practical Report

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 20

November 3, 2022

1 Code

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

void zombie()
{
    pid_t pid = fork();
    if (pid < 0)
    {
        printf("\t Fork Error \n");
    }
    else if (pid == 0)
    {
        printf("\n");
        printf("Child id is %d : \n", getpid());
    }
    else
    {
        sleep(2);
        // wait(NULL);
        printf("Parent here just woke up after 2 seconds post my child finished. \
    n");
        system("ps -axj | tail");
    }
}

void DeadButNotzombie()
{
    pid_t pid = fork();
    if (pid < 0)
    {
        printf("\t Fork Error \n");
    }
    else if (pid == 0)
    {
        printf("\n");
        printf("Child id is %d : \n", getpid());
    }
    else
    {
        sleep(2);
        wait(NULL);
        printf("Parent here just woke up after 2 seconds post my child finished. \
    n");
        system("ps -axj | tail");
    }
}

void orphan()
{
    int pid = fork();
    if (pid > 0) // parent who finishes soon
    {
        printf("Parent Process here, My Id is This is what my child should get
```

```
    when it calls getppid(): %d", getpid());
55      }
56      else if (pid == 0) // child that doesnt finish and is adopted
57      {
58          sleep(3);
59          printf("\nOrphan child here, My Process id: %d \n", getpid());
60          printf("Orphan child here, Parent Process id upon calling getppid(): %d \n
    ", getppid());
61          system("ps -axj | tail");
62      }
63  }
64
65  int main()
66  {
67      zombie();
68      // DeadButNotzombie();
69      // orphan();
70      return 0;
71  }
```

Listing 1: Assignment 2.Cpp

2 Input and Output

```
1  // Zombie Process (Defunct)
2
3  Child id is 7424
4
5  Parent here just woke up after 2 seconds post my child finished.
6        2    3983       0       0 ?             -1 I        0   0:00 [kworker/3:0-
       events]
7        2    6653       0       0 ?             -1 I        0   0:00 [kworker/3:2-
       mm_percpu_wq]
8        2    6757       0       0 ?             -1 I        0   0:00 [kworker/u16:2-
       events_unbound]
9      937    7118    2640    2640 ?             -1 Sl    1000   0:00 /home/
       krishnaraj/.vscode-insiders/extensions/ms-vscode.cpptools-1.13.3-linux-x64/bin/
       cpptools-srv 3005 {329B06C5-76B5-4C8F-B516-80B5A714933A}
10       2    7342       0       0 ?             -1 I        0   0:00 [kworker/6:0-
       events]
11    3215    7423    7423    3215 pts/2       7423 S+    1000   0:00 /run/media/
       krishnaraj/Classes/University/Second Year/First Semister/Operating Systems/
       Programs/Assignment_2
12    7423    7424    7423    3215 pts/2       7423 Z+    1000   0:00 [Assignment_2]
       <defunct>
13    7423    7440    7423    3215 pts/2       7423 S+    1000   0:00 sh -c ps -axj |
        tail
14    7440    7441    7423    3215 pts/2       7423 R+    1000   0:00 ps -axj
15    7440    7442    7423    3215 pts/2       7423 S+    1000   0:00 tail
16
17 // Zombie Process But if Parent waits for child, in which case it doesnt become a
       Zombie
18
19 Child id is 8768
20
21 Parent here just woke up after 2 seconds post my child finished.
22       2    7342       0       0 ?             -1 I        0   0:00 [kworker/6:0-
       events]
```

```
      2    7494       0       0 ?             -1 I        0   0:00 [kworker/u16:3-
    ext4-rsv-conversion]
      2    7601       0       0 ?             -1 I        0   0:00 [kworker/3:1-
    events]
    937    7628     957     957 ?             -1 Sl    1000   0:07 /usr/bin/gjs /
    usr/lib/org.gnome.NautilusPreviewer
      2    8243       0       0 ?             -1 I        0   0:00 [kworker/u16:1-
    events_power_efficient]
      2    8603       0       0 ?             -1 I        0   0:00 [kworker/3:0]
   3215    8767    8767    3215 pts/2       8767 S+    1000   0:00 /run/media/
    krishnaraj/Classes/University/Second Year/First Semister/Operating Systems/
    Programs/Assignment_2
   8767    8784    8767    3215 pts/2       8767 S+    1000   0:00 sh -c ps -axj |
     tail
   8784    8785    8767    3215 pts/2       8767 R+    1000   0:00 ps -axj
   8784    8786    8767    3215 pts/2       8767 S+    1000   0:00 tail
 krishnaraj@Krishnaraj-Arch     /run/media/krishnaraj/Classes/University/Seco


// Orphan Process

Parent Process here, My Id is This (what my child should get when it calls getppid
    ()): 8307

krishnaraj@Krishnaraj-Arch     /run/media/krishnaraj/Classes/University/Second
    Year/First Semister/Operating Systems/Programs         main

Orphan child here, My Process id: 8308
Orphan child here, Parent Process id upon calling getppid(): 937

    937    7118    2640    2640 ?             -1 Sl    1000   0:00 /home/
    krishnaraj/.vscode-insiders/extensions/ms-vscode.cpptools-1.13.3-linux-x64/bin/
    cpptools-srv 3005 {329B06C5-76B5-4C8F-B516-80B5A714933A}
      2    7342       0       0 ?             -1 I        0   0:00 [kworker/6:0-
    events]
      2    7494       0       0 ?             -1 I        0   0:00 [kworker/u16:3-
    ext4-rsv-conversion]
      2    7601       0       0 ?             -1 I        0   0:00 [kworker/3:1-
    mm_percpu_wq]
    937    7628     957     957 ?             -1 Sl    1000   0:06 /usr/bin/gjs /
    usr/lib/org.gnome.NautilusPreviewer
      2    8243       0       0 ?             -1 I        0   0:00 [kworker/u16:1-
    flush-8:16]
    937    8308    8307    3215 pts/2       3215 S     1000   0:00 /run/media/
    krishnaraj/Classes/University/Second Year/First Semister/Operating Systems/
    Programs/Assignment_2
   8308    8357    8307    3215 pts/2       3215 S     1000   0:00 sh -c ps -axj |
     tail
   8357    8358    8307    3215 pts/2       3215 R     1000   0:00 ps -axj
   8357    8359    8307    3215 pts/2       3215 S     1000   0:00 tail
```

Listing 2: Input and Output.Cpp

11/10/20

OS- ASSIGNMENT - 2

Krishnaraj PL
1032210888
PA20

Q.1. Explain Orphan Process concept. Who takes care of orphan processes and how?

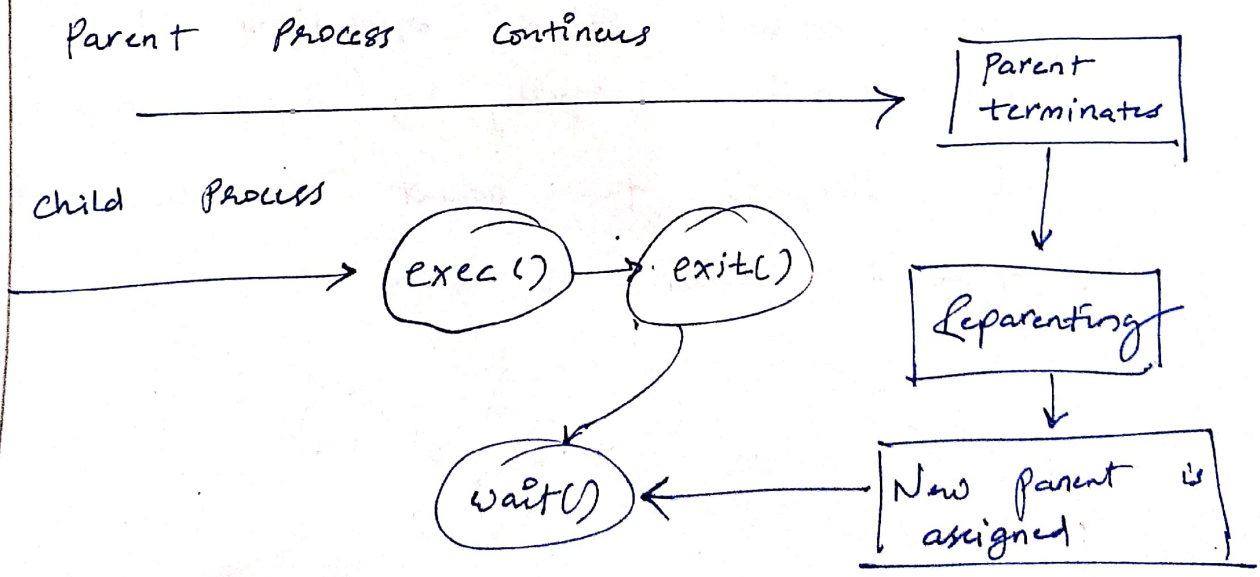
→ Orphan Process: A process whose parent process no more exists; that is, it has terminated without waiting for its child process to terminate is called an orphan process.

→ It gets a new parent (Reparenting)
The Kernel detects that a process has become orphaned, and assigns a new parent to it.

→ In most cases, it is the INIT process (PID = 1)

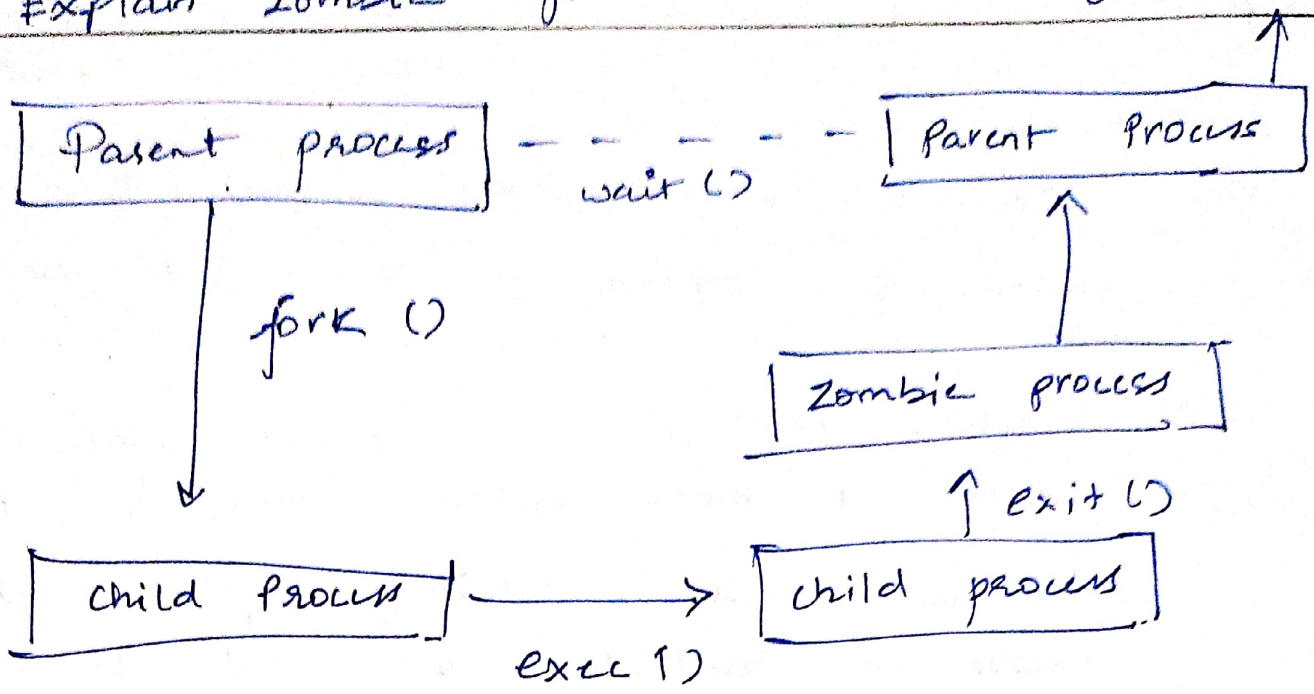
→ The new parent waits for child to be terminated, and then asks OS to clear the PCB of the orphaned process.

Q.2.



q.2. Explain Zombie process with a diagram.

→



(1.) A zombie process is a process whose execution is completed but it has an entry in the process table.

(2.) Zombie processes usually occur for child processes as the parent process still reads its child's exit status.

q.3. Explain sleep () function.

→ Sleep function is c allows the users to wait for a current thread for a specific time interval. Other operations of the CPU will function properly ; but sleep() function will sleep () the parent executable for the specified amount of time.