Zombie Processes and their Prevention

Difficulty Level : Medium • Last Updated : 16 Jul, 2021

Prerequisites: fork() in C, Zombie Process

Zombie state: When a process is created in UNIX using fork() system call, the address space of the Parent process is replicated. If the parent process calls wait() system call, then the execution of parent is suspended until the child is terminated. At the termination of the child, a 'SIGCHLD' signal is generated which is delivered to the parent by the kernel. Parent, on receipt of 'SIGCHLD' reads the status of the child from the process table. Even though, the child is terminated, there is an entry in the process table corresponding to the child where the status is stored. When parent collects the status, this entry is deleted. Thus, all the traces of the child process are removed from the system. If the parent decides not to wait for the child's termination and it executes its subsequent task, then at the termination of the child, the exit status is not read. Hence, there remains an entry in the process table even after the termination of the child. This state of the child process is known as the Zombie state.

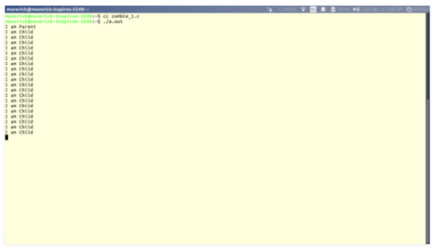
C

```c
// A C program to demonstrate working of
// fork() and process table entries.
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>
#include<sys/types.h>

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

    if (pid == 0)
    {
        for (i=0; i<20; i++)
            printf("I am Child\n");
    }
    else
    {
        printf("I am Parent\n");
        while(1);
    }
}
```

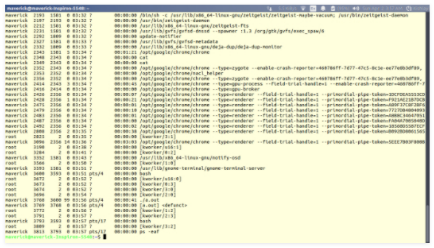
Output :



```
maverick@maverick-Inspiron-5548:~$ cc zombie_1.c
maverick@maverick-Inspiron-5548:~$ ./a.out
I am Parent
I am Child
I am Child
I am Child
I am Child
I am Child
I am Child
I am Child
I am Child
I am Child
I am Child
I am Child
I am Child
I am Child
I am Child
I am Child
I am Child
I am Child
I am Child
I am Child
I am Child
```

Now check the process table using the following command in the terminal

$ ps -eaf



```
maverick  2193  1581  0 03:32 ?        00:00:00 /bin/sh -c /usr/lib/x86_64-linux-gnu/zeitgeist/zeitgeist-maybe-vacuum; /usr/bin/zeitgeist-daemon
maverick  2197  2193  0 03:32 ?        00:00:00 /usr/bin/zeitgeist-daemon
maverick  2212  1581  0 03:32 ?        00:00:00 /usr/lib/x86_64-linux-gnu/zeitgeist-fts
maverick  2231  1581  0 03:32 ?        00:00:00 /usr/lib/gvfs/gvfsd-dnssd --spawner :1.3 /org/gtk/gvfs/exec_spaw/6
maverick  2292  1809  0 03:32 ?        00:00:00 update-notifier
maverick  2327  1581  0 03:33 ?        00:00:00 /usr/lib/gvfs/gvfsd-metadata
maverick  2332  1809  0 03:33 ?        00:00:00 /usr/lib/x86_64-linux-gnu/deja-dup/deja-dup-monitor
maverick  2343  1581  5 03:34 ?        00:01:21 /opt/google/chrome/chrome
maverick  2348  2343  0 03:34 ?        00:00:00 cat
maverick  2349  2343  0 03:34 ?        00:00:00 cat
maverick  2352  2343  0 03:34 ?        00:00:00 /opt/google/chrome/chrome --type=zygote --enable-crash-reporter=460786ff-7d77-47c5-8c1e-ee77e8b3df89,
maverick  2353  2352  0 03:34 ?        00:00:00 /opt/google/chrome/nacl_helper
maverick  2356  2352  0 03:34 ?        00:00:00 /opt/google/chrome/chrome --type=zygote --enable-crash-reporter=460786ff-7d77-47c5-8c1e-ee77e8b3df89,
maverick  2414  2343  3 03:34 ?        00:00:45 /opt/google/chrome/chrome --type=gpu-process --field-trial-handle=1 --enable-crash-reporter=460786ff-7
maverick  2416  2414  0 03:34 ?        00:00:00 /opt/google/chrome/chrome --type=gpu-broker
maverick  2426  2356  0 03:34 ?        00:00:07 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=1DCFDEA5553CD
maverick  2428  2356  1 03:34 ?        00:00:21 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=F921AE21B7DCB
maverick  2475  2356  0 03:34 ?        00:00:01 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=A09F37C8F28F6
maverick  2479  2356  0 03:34 ?        00:00:10 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=7727D84804093
maverick  2483  2356  0 03:34 ?        00:00:01 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=A808C34647051
maverick  2487  2356  0 03:34 ?        00:00:00 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=FA04A7085840D
maverick  2492  2356  0 03:34 ?        00:00:02 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=18560D5587ECF
maverick  2808  2356  2 03:35 ?        00:00:38 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=80928D8061565
root      2825     2  0 03:35 ?        00:00:00 [kworker/3:1]
maverick  3096  2356 14 03:36 ?        00:03:03 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=5EEE7803F8088
root      3190     2  0 03:38 ?        00:00:00 [kworker/u16:1]
root      3284     2  0 03:41 ?        00:00:00 [kworker/0:2]
maverick  3352  1581  0 03:43 ?        00:00:00 /usr/lib/x86_64-linux-gnu/notify-osd
root      3566     2  0 03:50 ?        00:00:00 [kworker/1:0]
maverick  3593  1581  0 03:51 ?        00:00:00 /usr/lib/gnome-terminal/gnome-terminal-server
maverick  3600  3593  0 03:51 pts/4    00:00:00 bash
root      3672     2  0 03:52 ?        00:00:00 [kworker/u16:0]
root      3673     2  0 03:52 ?        00:00:00 [kworker/0:3]
root      3674     2  0 03:52 ?        00:00:00 [kworker/3:0]
root      3696     2  0 03:54 ?        00:00:00 [kworker/2:0]
maverick  3768  3600 99 03:56 pts/4    00:00:41 ./a.out
maverick  3769  3768  0 03:56 pts/4    00:00:00 [a.out] <defunct>
root      3772     2  0 03:56 ?        00:00:00 [kworker/1:2]
root      3791     2  0 03:57 ?        00:00:00 [kworker/2:3]
maverick  3793  3593  0 03:57 pts/17   00:00:00 bash
root      3809     2  0 03:57 ?        00:00:00 [kworker/3:2]
maverick  3813  3793  0 03:57 pts/17   00:00:00 ps -eaf
maverick@maverick-Inspiron-5548:~$
```

Here the entry **[a.out] defunct** shows the zombie process.

Why do we need to prevent the creation of Zombie process?

There is one process table per system. The size of the process table is finite. If too many zombie processes are generated, then the process table will be full. That is, the system will not be able to generate any new process, then the system will come to a standstill. Hence, we need to prevent the creation of zombie processes.

Different ways in which the creation of Zombie can be Prevented

1. Using wait() system call: When the parent process calls wait(), after the creation of a child, it indicates that, it will wait for the child to

WHAT'S NEW

DSA Course Class 9 to 12 School Students
View Details

DSA Self Paced Course
View Details

DSA Live Classes for Working Professionals
View Details

MOST POPULAR IN C LANGUAGE

Different methods to reverse a string in C/C++

Left Shift and Right Shift Operators in C/C++

Multidimensional Arrays in C / C++

Enumeration (or enum) in C

Substring in C++

1. Using wait() system call: When the parent process calls wait(), after the creation of a child, it indicates that, it will wait for the child to complete and it will reap the exit status of the child. The parent process is suspended(waits in a waiting queue) until the child is terminated. It must be understood that during this period, the parent process does nothing just waits.

C

```c
// A C program to demonstrate working of
// fork()/wait() and Zombie processes
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>
#include<sys/types.h>

int main()
{
    int i;
    int pid = fork();
    if (pid==0)
    {
        for (i=0; i<20; i++)
            printf("I am Child\n");
    }
    else
    {
        wait(NULL);
        printf("I am Parent\n");
        while(1);
    }
}
```

2. By ignoring the SIGCHLD signal: When a child is terminated, a corresponding SIGCHLD signal is delivered to the parent, if we call the 'signal(SIGCHLD,SIG_IGN)', then the SIGCHLD signal is ignored by the system, and the child process entry is deleted from the process table. Thus, no zombie is created. However, in this case, the parent cannot know about the exit status of the child.

C

```c
// A C program to demonstrate ignoring
// SIGCHLD signal to prevent Zombie processes
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>
#include<sys/types.h>

int main()
{
    int i;
    int pid = fork();
    if (pid == 0)
        for (i=0; i<20; i++)
            printf("I am Child\n");
    else
    {
        signal(SIGCHLD,SIG_IGN);
        printf("I am Parent\n");
        while(1);
    }
}
```

3. By using a signal handler: The parent process installs a signal handler for the SIGCHLD signal. The signal handler calls wait() system call within it. In this scenario, when the child terminated, the SIGCHLD is delivered to the parent. On receipt of SIGCHLD, the corresponding handler is activated, which in turn calls the wait() system call. Hence, the parent collects the exit status almost immediately and the child entry in the process table is cleared. Thus no zombie is created.

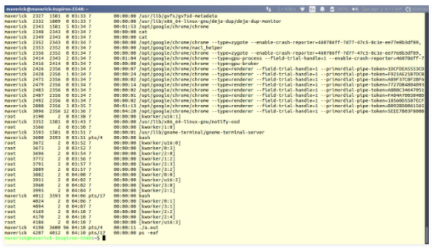
C

```c
// A C program to demonstrate handling of
// SIGCHLD signal to prevent Zombie processes.
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>
#include<sys/types.h>

void func(int signum)
{
    wait(NULL);
}

int main()
{
    int i;
    int pid = fork();
    if (pid == 0)
        for (i=0; i<20; i++)
            printf("I am Child\n");
    else
    {
        signal(SIGCHLD, func);
        printf("I am Parent\n");
        while(1);
    }
}
```

Output:



```
maverick@maverick-Inspiron-5548: ~
maverick  2327  1581  0 03:33 ?        00:00:00 /usr/lib/gvfs/gvfsd-metadata
maverick  2332  1809  0 03:33 ?        00:00:00 /usr/lib/x86_64-linux-gnu/deja-dup/deja-dup-monitor
maverick  2343  1581  5 03:34 ?        00:01:53 /opt/google/chrome/chrome
maverick  2348  2343  0 03:34 ?        00:00:00 cat
maverick  2349  2343  0 03:34 ?        00:00:00 cat
maverick  2352  2343  0 03:34 ?        00:00:00 /opt/google/chrome/chrome --type=zygote --enable-crash-reporter=460786ff-7d77-47c5-8c1e-ee77e8b3df89,
maverick  2353  2352  0 03:34 ?        00:00:00 /opt/google/chrome/nacl_helper
maverick  2356  2352  0 03:34 ?        00:00:00 /opt/google/chrome/chrome --type=zygote --enable-crash-reporter=460786ff-7d77-47c5-8c1e-ee77e8b3df89,
maverick  2414  2343  2 03:34 ?        00:01:04 /opt/google/chrome/chrome --type=gpu-process --field-trial-handle=1 --enable-crash-reporter=460786ff-7
maverick  2416  2414  0 03:34 ?        00:00:00 /opt/google/chrome/chrome --type=gpu-broker
maverick  2426  2356  0 03:34 ?        00:00:07 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=1DCFDEA5553CD
maverick  2428  2356  1 03:34 ?        00:00:24 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=F921A62187DC8
maverick  2475  2356  0 03:34 ?        00:00:02 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=A89F37C8F28F6
maverick  2479  2356  0 03:34 ?        00:00:14 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=7727D84884093
maverick  2483  2356  0 03:34 ?        00:00:02 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=A880C34647051
maverick  2487  2356  0 03:34 ?        00:00:01 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=FAD4A7085040D
maverick  2492  2356  0 03:34 ?        00:00:02 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=1856005587ECF
maverick  2808  2356  3 03:35 ?        00:01:13 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=80928D8061565
maverick  3096  2356 12 03:36 ?        00:04:28 /opt/google/chrome/chrome --type=renderer --field-trial-handle=1 --primordial-pipe-token=5EEE7883F8088
root      3190     2  0 03:38 ?        00:00:00 [kworker/u16:1]
maverick  3352  1581  0 03:43 ?        00:00:00 /usr/lib/x86_64-linux-gnu/notify-osd
root      3566     2  0 03:50 ?        00:00:00 [kworker/1:0]
maverick  3593  1581  0 03:51 ?        00:00:01 /usr/lib/gnome-terminal/gnome-terminal-server
maverick  3600  3593  0 03:51 pts/4    00:00:00 bash
root      3672     2  0 03:52 ?        00:00:00 [kworker/u16:0]
root      3673     2  0 03:52 ?        00:00:00 [kworker/0:3]
root      3696     2  0 03:54 ?        00:00:00 [kworker/2:0]
root      3772     2  0 03:56 ?        00:00:00 [kworker/1:2]
root      3791     2  0 03:57 ?        00:00:00 [kworker/2:3]
root      3809     2  0 03:57 ?        00:00:00 [kworker/3:2]
root      3882     2  0 04:00 ?        00:00:00 [kworker/0:0]
root      3951     2  0 04:02 ?        00:00:00 [kworker/u16:2]
root      3960     2  0 04:02 ?        00:00:00 [kworker/3:0]
root      3995     2  0 04:04 ?        00:00:00 [kworker/2:1]
maverick  4012  3593  0 04:06 pts/17   00:00:00 bash
root      4024     2  0 04:06 ?        00:00:00 [kworker/0:1]
root      4094     2  0 04:07 ?        00:00:00 [kworker/3:1]
root      4169     2  0 04:10 ?        00:00:00 [kworker/2:2]
root      4170     2  0 04:10 ?        00:00:00 [kworker/2:4]
root      4186     2  0 04:10 ?        00:00:00 [kworker/u16:3]
maverick  4198  3600 96 04:10 pts/4    00:00:11 ./a.out
maverick  4207  4012  0 04:10 pts/17   00:00:00 ps -eaf
maverick@maverick-Inspiron-5548:~$
```

Here no any **[a.out] defunct** i.e. no any Zombie process is created.

This article is contributed by **Kishlay Verma**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

MOST VISITED IN LINUX-UNIX

Sed Command in Linux/Unix with examples

grep command in Unix/Linux

AWK command in Unix/Linux with examples

cp command in Linux with examples

cut command in Linux with examples

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Want to learn from the best curated videos and practice problems, check out the C Foundation Course for Basic to Advanced C.

Like 21

Next

Zombie and Orphan Processes in C

RECOMMENDED ARTICLES

Page : 1 2 3

01 Zombie and Orphan Processes in C
16, May 16

02 Chain processes vs Fan of processes using fork() function in C
05, Jul 21

03 How to execute zombie and orphan process in a single program?
04, Sep 18

04 Maximum number of Zombie process a system can handle
27, May 17

05 Double forking to prevent Zombie process
30, May 17

06 Format String Vulnerability and Prevention with Example
23, Apr 17

07 Difference between Deadlock Prevention and Deadlock Avoidance
16, May 20

08 Deadlock Prevention And Avoidance
29, Jun 15

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Medium

Easy Normal Medium Hard Expert

Improved By : sagar0719kumar, shubhammalpani

Article Tags : Processes & Threads, system-programming, C Language, Linux-Unix, Operating Systems

Practice Tags : Operating Systems

Improve Article Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GeeksforGeeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Company
About Us
Careers
Privacy Policy
Contact Us
Copyright Policy

Learn
Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

Practice
Courses
Company-wise
Topic-wise
How to begin?

Contribute
Write an Article
Write Interview Experience
Internships
Videos