Zombie processes in Linux

Processes in UNIX like systems

A child process inherits most of its attributes, such as file descriptors, from its parent. In Unix, a child process is typically created as a copy of the parent, using the fork system call.

Each process may create many child processes but will have at most one parent process; if a process does not have a parent this usually indicates that it was created directly by the kernel (The kernel is a program which acts as an interface between applications and hardware for resource allocation, memory management, space management, process management and task management). In some systems, including Linux-based systems, the very first process (called init) is started by the kernel at booting time and never terminates.

Child Process exit/interrupt

When a process ends via exit, following things occur:

1] Parent receives SIGCHLD signal

2] all the memory and resources associated with it are deallocated so they can be used by other processes

Orphan process:

When a parent of child process dies, the child process becomes orphan but shortly gets adopted by init.

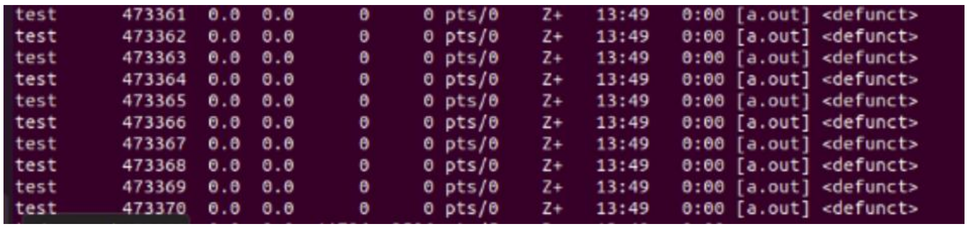
Zombie processes:

When a child process terminates, it becomes a zombie process, and continues to exist as an entry in the system process table even though it is no longer an actively executing program. Under normal operation it will typically be immediately waited on by its parent, and then reaped by the system, reclaiming the resource (the process table entry). The parent can read the child's exit status by executing the waitpid() or wait() system call, whereupon the zombie is removed. The waitpid() or wait() call may be executed in sequential code, but it is commonly executed in a handler for the SIGCHLD signal. If a child is not waited on by its parent, it continues to consume this resource indefinitely, and thus is a resource leak.

After the zombie is removed, its process identifier (PID) and entry in the process table can then be reused.

How to identify zombie processes in Linux?

Zombies can be identified in the output from the Unix **ps aux** command by the presence of a "Z" or "Z+" or "Z<" in the "STAT" column.



```
test      473361  0.0  0.0      0     0 pts/0    Z+   13:49   0:00 [a.out] <defunct>
test      473362  0.0  0.0      0     0 pts/0    Z+   13:49   0:00 [a.out] <defunct>
test      473363  0.0  0.0      0     0 pts/0    Z+   13:49   0:00 [a.out] <defunct>
test      473364  0.0  0.0      0     0 pts/0    Z+   13:49   0:00 [a.out] <defunct>
test      473365  0.0  0.0      0     0 pts/0    Z+   13:49   0:00 [a.out] <defunct>
test      473366  0.0  0.0      0     0 pts/0    Z+   13:49   0:00 [a.out] <defunct>
test      473367  0.0  0.0      0     0 pts/0    Z+   13:49   0:00 [a.out] <defunct>
test      473368  0.0  0.0      0     0 pts/0    Z+   13:49   0:00 [a.out] <defunct>
test      473369  0.0  0.0      0     0 pts/0    Z+   13:49   0:00 [a.out] <defunct>
test      473370  0.0  0.0      0     0 pts/0    Z+   13:49   0:00 [a.out] <defunct>
```

***How to explicitly remove zombies from a system if parent doesn't wait on process after receiving* SIGCHLD?**

The SIGCHLD signal can be sent to the parent manually, using the kill command. If the parent process still refuses to reap the zombie, and if it would be fine to terminate the parent process, the next step can be to remove the parent process. When a process loses its parent, init becomes its new parent. init periodically executes the wait system call to reap any zombies with init as parent.

How to create zombie processes for testing purpose?

It is not easy to make existing process to be a zombie. Following piece of code creates a parent process which forks child processes, calls exit on each of them and sleeps for some time (in this time the processes are in zombie state) and lastly calls wait for each.

```
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
 int main(void)
{
    pid_t pids[10];
    int i;
    for (i = 9; i >= 0; --i) {
        pids[i] = fork(); // Creates child processes
        if (pids[i] == 0) {
            printf("Child%d\n", i);
            sleep(i+1); // last process will sleep for 10 sec before exit
            _exit(0);
        }
    }
    sleep(50); // Processes are in zombie state
    for (i = 9; i >= 0; --i) {
        printf("parent%d\n", i);
        waitpid(pids[i], NULL, 0); // Parent procesess waited for every child to remove them from
//zombie state
    }
    return 0;
}
```

-Udayati More