# **What are zombie processes?**

**1) With commands like top I can see that at the moment I have several zombie processes.**

**2) What are zombie processes?**

**Do they affect the performance of the system or the application they are zombie to. Do they use too much memory or any memory at all?**

Zombies are DEAD processes. They can not be 'kill' (You cannot kill the DEAD). All processes eventually die, and when they do they become zombies. They consume almost no resources, which is to be expected because they are dead!

The reason for zombies is so the zombie's parent (process) can retrieve the zombie's exit status and resource usage statistics. The parent signals the operating system that it no longer needs the zombie by using one of the **wait()** system calls.

When a process dies, its child processes all become children of process number 1, which is the **init** process. **Init is ``always'' waiting for children to die, so that they don't remain as zombies.**

If you have zombie processes it means those zombies have not been waited for by their parent **(look at PPID displayed by ps -l).** You have three choices: Fix the parent process (make it wait); kill the parent; or live with it. Remember that living with it is not so hard because zombies take up little more than one extra line in the output of ps.

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

Zombies that exist for more than a short period of time typically indicate a bug in the parent program. As with other leaks, the presence of a few zombies isn't worrisome in itself, but may indicate a problem that would grow serious under heavier loads.

To remove zombies from a system, the SIGCHLD signal can be sent to the parent manually, using the kill command. If the parent process still refuses to reap the zombie, the next step would 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.

Zombie processes (also show as <defunct>), aren't real processes at all. They are just entries in the kernel process table. This is the only resource they consume. They do not consume any CPU or RAM. The only danger of having zombies, is running out of space in process table (you can use cat /proc/sys/kernel/threads-max to see how many entries are allowed on your system).

They appear only when their parent process (i.e. process which fork()'ed them) is alive, but did not yet call wait() system function. Once parent dies, the zombies are wait()'ed for by init and disappear.

A zombie process is not the same as an orphan process. Orphan processes don't become zombie processes; instead, they are adopted by init (process ID 1), which waits on its children.   
  
The term zombie process derives from the common definition of zombie an undead person.

To kill a zombie (process) you have to kill its parent process (just like real zombies!), but the question was how to find it.

**Find the zombie** (The question answered this part):

a@SERVER:~$ ps aux | grep 'Z'

What you get is Zombies and anything else with a Z in it, so you will also get the grep:

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

usera 13572 0.0 0.0 7628 992 pts/2 S+ 19:40 0:00 grep --color=auto Z

usera 93572 0.0 0.0 0 0 ?? Z 19:40 0:00 something

**Find the zombie's parent:**

a@SERVER:~$ pstree -p -s 93572

Will give you:

init(1)---cnid\_metad(1311)---cnid\_dbd(5145)

In this case you do not want to kill that parent process and you should be quite happy with one zombie, but killing the immediate parent process 5145 should get rid of it.

## Example

*Synchronously* waiting for the specific child processes in a (specific) order may leave zombies present longer than the above-mentioned "short period of time". It is not necessarily a program bug.

#include<sys/wait.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();

if(pids[i]==0)

{sleep(i+1);

exit(0);

}}

for(i=9;i>=0;--i)

waitpid(pids[i],NULL,0);

return 0;}