

Forum:[Hack Forums](#)

Thread: [Kill child process when parent process terminates](#)

chrizator - July 21, 2018, 5:53 am

I'm currently working on a little tool that redirects cmd.exe I/O to a server by using pipes. Everything worked like a charm, but I noticed a bug.

When I create the process using `CreateProcess()`, I set `blnInheritHandles` to `TRUE`, in order to inherit the pipe handles.

But when the program gets terminated forcefully, the child process (cmd.exe) stays alive. At first, I thought it's not that big of a deal. But when the child process stays alive, somehow, my `SOCKET` handle doesn't get closed and the server will not notice the disconnect, though the parent process exited.

Is my assumption right?

And if it is, how would I close the child process when the parent process terminates?

Thank y'all in advance! ?

Auriana - July 21, 2018, 2:16 pm

When the parent creates the child process, it has the information about the child such as the handle and the process ID. From there you have a couple of options, the first is if you're running your parent process on a window with event callbacks registered, you can catch `wm_close` operation and from there, terminate the child process and close the handle. Or if you don't have a window registered, set a callback using `SetConsoleCtrlHandler` docs.microsoft.com/en-us/windows/console/setconsolectrlhandler which will capture the close event if you check the operation passed accordingly and from there you can terminate the child and close the handle.

chrizator - July 21, 2018, 4:31 pm

(Jul 21, 2018 02:16 PM)Auriana Wrote: [\[CITE\]](#) When the parent creates the child process, it has the information about the child such as the handle and the process ID. From there you have a couple of options, the first is if you're running your parent process on a window with event callbacks registered, you can catch `wm_close` operation and from there, terminate the child process and close the handle. Or if you don't have a window registered, set a callback using `SetConsoleCtrlHandler` docs.microsoft.com/en-us/windows/console/setconsolectrlhandler which will capture the close event if you check the operation passed accordingly and from there you can terminate the child and close the handle.

Thanks for your reply. Unfortunately, the function you described does not work in my scenario, since I don't have a console running.

I guess, the best thing I could do is to create a "Watcher" process that waits for the handle to be closed. Kind of nasty, though :/

Auriana - July 21, 2018, 8:42 pm

(Jul 21, 2018 04:31 PM)chrizator Wrote: [\[CITE\]](#) Thanks for your reply. Unfortunately, the function you described does not work in my scenario, since I don't have a console running.

I guess, the best thing I could do is to create a "Watcher" process that waits for the handle to be closed. Kind of nasty, though :/

If you don't have a console running then it's easier because your window has a registered callback

event handler for the main message loop. Which receives a `wm_close/wm_quit` event.

Anzee - July 22, 2018, 5:09 am

You should initialize a new thread instead of forking an entirely new process. If the parent process of a child is killed that child is orphaned and can be considered it's own process entirely.

The following code snippet daemonizes a program by forking a child and orphaning it:

```
Code if (fork() != 0) {  
    exit(0);  
}
```

This starts a new thread within the parent process and joins it using `pthread`:

```
Code void foo()  
{  
    while(1) {  
    }  
}  
  
int main()  
{  
    pthread_t thread;  
  
    pthread_create(&thread, NULL, foo, (void*) 1);  
  
    pthread_join(thread, NULL);  
  
    return 0;  
}
```