How do I automatically destroy child processes in Windows?

Asked 16 years, 3 months ago Modified 3 years, 3 months ago

Viewed 32k times

72

In C++ Windows app, I launch several long running child processes (currently I use CreateProcess(...) to do this.

I want the child processes to be automatically closed **if my main processes crashes** or is closed.

Because of the requirement that this needs to work for a crash of the "parent", I believe this would need to be done using some API/feature of the operating system. So that all the "child" processes are cleaned up.

How do I do this?

windows process

Share

Improve this question

Follow

edited Feb 18, 2014 at 6:38

tshepang
12.4k ● 25 ● 95 ● 139

asked Sep 10, 2008 at 0:13

jm.
23.7k ● 23 ● 81 ● 93

7 Answers

Sorted by: Highest score (default)

88

The Windows API supports objects called "Job Objects". The following code will create a "job" that is configured to shut down all processes when the main application ends (when its handles are cleaned up). This code should only be run once.:

```
HANDLE ghJob = CreateJobObject( NULL, NULL); // GLOBAL
if( ghJob == NULL)
{
    ::MessageBox( 0, "Could not create job object", "T
}
else
{
    JOBOBJECT_EXTENDED_LIMIT_INFORMATION jeli = { 0 };

    // Configure all child processes associated with t
the
    jeli.BasicLimitInformation.LimitFlags = JOB_OBJECT
    if( 0 == SetInformationJobObject( ghJob, JobObject
&jeli, sizeof(jeli)))
    {
        ::MessageBox( 0, "Could not SetInformationJobO
    }
}
```

Then when each child process is created, execute the following code to launch each child each process and add it to the job object:

```
STARTUPINFO info={sizeof(info)};
PROCESS_INFORMATION processInfo;

// Launch child process - example is notepad.exe
```

```
if (::CreateProcess( NULL, "notepad.exe", NULL, NULL,
&info, &processInfo))
{
    ::MessageBox( 0, "CreateProcess succeeded.", "TEST
    if(ghJob)
    {
        if(0 == AssignProcessToJobObject( ghJob, proce
        {
            ::MessageBox( 0, "Could not AssignProcessT
        }
    }

    // Can we free handles now? Not sure about this.
    //CloseHandle(processInfo.hProcess);
    CloseHandle(processInfo.hThread);
}
```

VISTA NOTE: See AssignProcessToJobObject always return "access denied" on Vista if you encounter access-denied issues with AssignProcessToObject() on vista.

Share Improve this answer Follow

edited Nov 27, 2013 at 14:53
KindDragon
6,841 ● 4 ● 49 ● 77

answered Sep 10, 2008 at 0:22
jm.
23.7k ● 23 ● 81 ● 93

To answer your question in the comment: Yes, you should CloseHandle when you don't need the handle anymore.
– Adam Mitz Sep 12, 2008 at 1:30

1 No, I don't think so. JOB_OBJECT_LIMIT_KILL_ON_JOB_CLOSE causes the job to terminate when the last *job* handle is closed, process handles shouldn't make a difference. Have you tried it? – Adam Mitz Sep 16, 2008 at 2:13

1 It doesn't work on Windows 2000 :((But it should under other versions of windows) – rook Jul 8, 2010 at 20:56

4 There's a theoretical race condition here, if the new process exits quickly enough. To eliminate it, use the CREATE_SUSPENDED flag and call ResumeThread only after adding the process to the job. – Harry Johnston Jul 20, 2016 at 1:51

1 I used JobObjects per described above in an executable, which calls MsBuild and other stuff. I can kill the exe at any time, and everything cleans up immediately. Super. :) – Andreas Vergison Jun 14, 2017 at 17:40

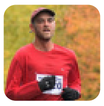
5

One somewhat hackish solution would be for the parent process to attach to each child as a debugger (use DebugActiveProcess). When a debugger terminates all its debuggee processes are terminated as well.

A better solution (assuming you wrote the child processes as well) would be to have the child processes monitor the parent and exit if it goes away.

Share Improve this answer Follow

answered Sep 10, 2008 at 0:28



Rob Walker
47.4k ● 15 ● 100 ● 137

2 Instead of having to call DebugActiveProcess, you could just pass DEBUG_PROCESS as one of your creation flags to CreateProcess. Less code that way. – mrduclaw Jul 17, 2009 at 22:12

3

Windows Job Objects sounds like a good place to start. The name of the Job Object would have to be well-known, or passed to the children (or inherit the handle). The children would need to be notice when the parent dies, either through a failed IPC "heartbeat" or just WFMO/WFSO on the parent's process handle. At that point any child process could TermianteJobObject to bring down the whole group.

Share Improve this answer Follow

answered Sep 10, 2008 at 6:24

Adam Mitz
6,033 ● 1 ● 30 ● 28

Instead a child process just can WFMO/WFSO on parent handle and close itself. Provided each child process does this there is no need for job object. – dkrikun Jan 22, 2018 at 18:29

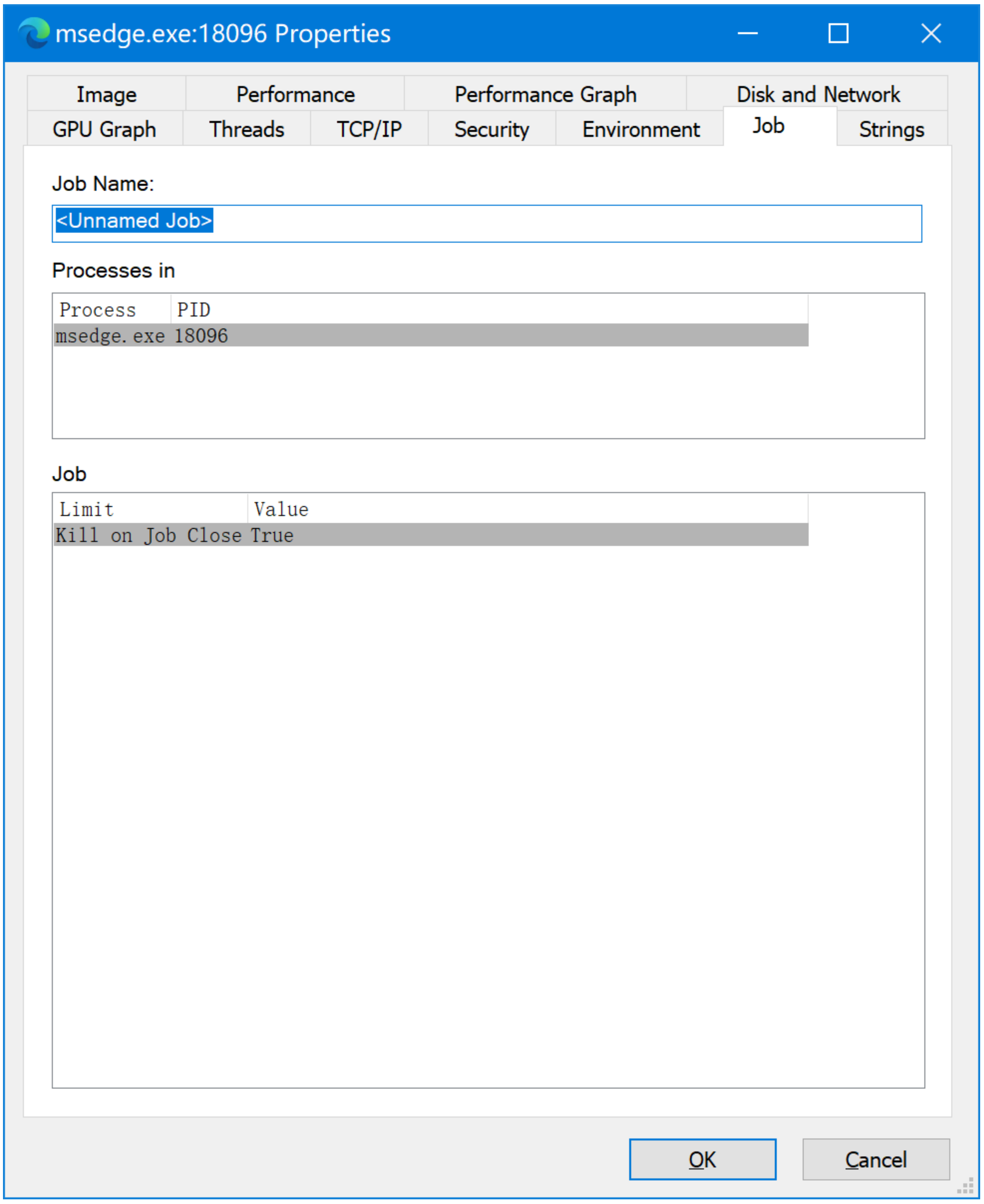
2

You can assign a job to the parent process before creating processes:

```
static HANDLE hjob_kill_on_job_close=INVALID_HANDLE_VA
void init(){
    hjob_kill_on_job_close = CreateJobObject(NULL, NUL
    if (hjob_kill_on_job_close){
        JOBOBJECT_EXTENDED_LIMIT_INFORMATION jobli = {
        jobli.BasicLimitInformation.LimitFlags =
JOB_OBJECT_LIMIT_KILL_ON_JOB_CLOSE;
        SetInformationJobObject(hjob_kill_on_job_close
            JobObjectExtendedLimitInformation,
            &jobli, sizeof(jobli));
        AssignProcessToJobObject(hjob_kill_on_job_clos
    }
}
void deinit(){
    if (hjob_kill_on_job_close) {
        CloseHandle(hjob_kill_on_job_close);
    }
}
```

`JOB_OBJECT_LIMIT_KILL_ON_JOB_CLOSE` causes all processes associated with the job to terminate when the last handle to the job is closed. By default, all child processes will be assigned to the job automatically, unless you passed `CREATE_BREAKAWAY_FROM_JOB` when calling `CreateProcess`. See https://learn.microsoft.com/en-us/windows/win32/procthread/process-creation-flags for more information about `CREATE_BREAKAWAY_FROM_JOB`.

You can use process explorer from Sysinternals to make sure all processes are assigned to the job. Just like this:



Job Name: <Unnamed Job>

Processes in

Process	PID
msedge.exe	18096

Job

Limit	Value
Kill on Job Close	True

Share Improve this answer Follow

answered Sep 19, 2021 at 12:54

Kohill Yang
141 ● 1 ● 4

Please note that this method can only be used in Windows 8/10， since a process can be associated with only one job. Jobs cannot be nested. The ability to nest jobs was added in Windows 8 and Windows Server 2012. – Kohill Yang Sep 19, 2021 at 13:18

1

You can keep a separate watchdog process running. Its only task is watching the current process space to spot situations like you describe. It could even re-launch the original application after a crash or provide different options to the user, collect debug information, etc. Just try to keep it simple enough so that you don't need a second watchdog to watch the first one.

Share Improve this answer Follow

answered Sep 10, 2008 at 0:25

Pedro
623 ● 1 ● 7 ● 12

-2

You'd probably have to keep a list of the processes you start, and kill them off one by one when you exit your program. I'm not sure of the specifics of doing this in C++ but it shouldn't be hard. The difficult part would probably be ensuring that child processes are shutdown in the case of an application crash. .Net has the ability to add a function that get's called when an unhandled exception occurs. I'm not sure if C++ offers the same capabilities.

Share Improve this answer Follow

answered Sep 10, 2008 at 0:19

Kibbee
66.1k ● 28 ● 144 ● 184

3 >> if my main processes crashes – jm. Jul 21, 2010 at 20:59

-2

You could encapsulate each process in a C++ object and keep a list of them in global scope. The destructors can shut down each process. That will work fine if the program exits normally but it it crashes, all bets are off.

Here is a rough example:

```
class myprocess
{
public:
    myprocess(HANDLE hProcess)
        : _hProcess(hProcess)
    { }

    ~myprocess()
    {
        TerminateProcess(_hProcess, 0);
    }

private:
    HANDLE _hProcess;
};

std::list<myprocess> allprocesses;
```

Then whenever you launch one, call allprocessess.push_back(hProcess);

Share Improve this answer Follow

answered Sep 10, 2008 at 0:34

Adam Pierce
34.3k ● 23 ● 71 ● 89

3 >> if my main processes crashes – jm. Jul 21, 2010 at 21:00