U++ - Bug #1129
Detached LocalProcess becomes zombie

06/18/2015 08:47 PM - Jan Dolinár

Status:	Approved	**Start date:**	06/18/2015
Priority:	Normal	**Due date:**	
Assignee:	Miroslav Fidler	**% Done:**	0%
Category:	Core	**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour

Description

Consider this example:

```
#include <Core/Core.h>

using namespace Upp;

CONSOLE_APP_MAIN {
   LocalProcess p;
   p.Start("sleep 5");
   p.Detach();
   Sleep(15000);
}
```

After the detached process exits, it is never collected and stays as a zombie process. I actually encountered this in a long running Skylark process (which executed some scripts in the background), where the number of zombies would gradually grow, threatening the stability of the system.

So I started working on a patch, which you can see in the attachment. It adds an option to do a double fork in the LocalProcess, which is a common solution to this problem. This works very well for simple cases, such as the example above, but when I used it for the Skylark app, I got some errors in log (and I *think* the thread executing the code died as well):

```
Error in my_thread_global_end(): 2 threads didn't exit
Heap leaks detected!
```

It seems that calling Exit(0) from Skylark thread is not enough to shut it down cleanly. Calling SkylarkApp::Quit() would be better in this case, but that can't be done in LocalProcess, because it is not aware about anything Skylark related. The only workaround I found is to not actually call Exit(), but call exec instead and replace the intermediary process with some short-lived program, namely "true".

The only downside to this patch is, that after using DoubleFork(true) the functions GetPid() and IsRunning() can't be used. This might be possible to fix - we would have to send the pid of the grandchild's process to the parent via pipe. Not sure if it is really necessary. If you think someone might need it, I can try to implement it as well.

Also, if you agree to apply this patch, I will update the documentation as well, just let me know in advance.

History

#1 - 06/19/2015 06:34 PM - Zbigniew Rebacz

I think you do not need two methods that doing exactly the same thing:

```
LocalProcess& DoubleFork(bool b = true)                 { doublefork = b; return *this; }
LocalProcess& NoDoubleFork(bool b = true)                 { return DoubleFork(false); }
```

What is wrong with "DoubleFork(false)" to have separate method "NoDoubleFork()".

P.S.
This is only coding standard suggestion.

#2 - 06/20/2015 12:54 PM - Jan Dolinár

Zbigniew Rebacz wrote:

> *What is wrong with "DoubleFork(false)" to have separate method "NoDoubleFork()".*

Nothing is wrong with that. This is just the way boolean flags are set in most U++ classes. See for example ConvertCharset/NoConvertCharset in the very same class. So, to keep the standard style, I wrote it that way too.

#3 - 06/20/2015 01:06 PM - Miroslav Fidler

Anyway, the parameter in NoDoubleFork is not used (do not worry, I will fix that as soon as I will understand the need for the change...:)

As for zombie issue, alternative solution is to reap them... Check e.g. sCleanZombies in ide.

Mirek

#4 - 06/20/2015 01:58 PM - Jan Dolinár

Oh, I see it now :-) It was just a stupid copy&paste error, I forgot to delete the parameter...

sCleanZombies is an interesting approach, but if I understand it correctly, you have to call it periodically - with some reasonable period, that depends on how much process do you launch. I still think it might be in some cases better to prevent process to become zombies in the first place.

For example in my current case, I can't really tell if there will be one zombie per day or multiple per minute. Also, since Skylark doesn't support periodic background jobs, it would mean that I have to collect the zombies either in some of the incoming requests or from outside (e.g. using http request from cron). That just doesn't feel very "effective". Although my server could easily handle those few more cycles, I'm just not happy with such solution when I know it can be done better :)

#5 - 06/20/2015 07:23 PM - Miroslav Fidler

I think that you missed the part when sCleanZombies is installed as signal handler (you do receive SIGCHLD a signal when child process goes zombie.. :)

Note that I do not oppose double-fork here - I think it is a good idea. Just suggesting alternative approach for Skylark...

#6 - 06/20/2015 07:28 PM - Miroslav Fidler

- Status changed from Patch ready to In Progress

Accepted; I am leaving it active for me to complete documentation; Topic++ is now in transition as new C++ parser is now aware of namespaces and so keys linking the code and topics are broken (Foo(String) is now Foo(Upp::String))

#7 - 06/21/2015 01:21 PM - Miroslav Fidler

- Status changed from In Progress to Approved

#8 - 06/22/2015 06:19 AM - Jan Dolinár

Thank you.

Should I try to add the missing GetPid/IsRunning functionality for double fork mode? It should be quite simple...

Files

LocalProcess.patch	2.33 KB	06/18/2015	Jan Dolinár