Subject: Segfaults with One container?
Posted by phirox on Tue, 22 Sep 2009 11:21:34 GMT
View Forum Message <> Reply to Message

I've had a really weird bug in my program, and have been able to downsize it to the following testcase. But I still cannot track what is going on(using v1517):

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

using namespace Upp;

CONSOLE_APP_MAIN
{
 Cout() << "main1\n";
 Socket s;
 Cout() << "main2\n";
 s.NoBlock();
 Cout() << "main3\n";
}
```

Results in:

Quote:
main1
main2
Assertion failed in /home/phirox/upp/uppsrc/Core/Other.h, line 17
ptr

My idea is that accessing any variable in the One<Data> object part of the new socket is making it crash. I've tried using several compilers and even other machines, but they all reproduce this result.

Subject: Re: Segfaults with One container?
Posted by mirek on Tue, 22 Sep 2009 15:52:12 GMT
View Forum Message <> Reply to Message

phirox wrote on Tue, 22 September 2009 07:21I've had a really weird bug in my program, and have been able to downsize it to the following testcase. But I still cannot track what is going on(using v1517):

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

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

using namespace Upp;

CONSOLE_APP_MAIN
{
 Cout() << "main1\n";
 Socket s;
 Cout() << "main2\n";
 s.NoBlock();
 Cout() << "main3\n";
}
```

Results in:

Quote:
main1
main2
Assertion failed in /home/phirox/upp/uppsrc/Core/Other.h, line 17
ptr

My idea is that accessing any variable in the One<Data> object part of the new socket is making it crash. I've tried using several compilers and even other machines, but they all reproduce this result.

Socket must be open prior to calling NoBlock.

Mirek

Subject: Re: Segfaults with One container?
Posted by phirox on Wed, 23 Sep 2009 07:22:39 GMT
View Forum Message <> Reply to Message

Thank you for your fast response, and as always it was right on. I think I have stared to long at the problem to notice it myself

For future references the solution for me was to add the following code:

```
 s.Init();
 One<Socket::Data> data = new Socket::Data;
 s.Attach(data);
```

The reason for all of this; under *nix when you fork() or reload your own or other program with exec() all sockets are inherited. So for example by just adding data->socket = 3; in the above example you can continue working on it without interruption. Maybe an idea to add this feature as a function OldSocket/OpenedSocket besides the existing ClientSocket and ServerSocket?

Subject: Re: Segfaults with One container?
Posted by mirek on Thu, 24 Sep 2009 07:43:51 GMT
View Forum Message <> Reply to Message

phirox wrote on Wed, 23 September 2009 03:22Thank you for your fast response, and as always it was right on. I think I have stared to long at the problem to notice it myself

For future references the solution for me was to add the following code:

```
s.Init();
One<Socket::Data> data = new Socket::Data;
s.Attach(data);
```

The reason for all of this; under *nix when you fork() or reload your own or other program with exec() all sockets are inherited. So for example by just adding data->socket = 3; in the above example you can continue working on it without interruption. Maybe an idea to add this feature as a function OldSocket/OpenedSocket besides the existing ClientSocket and ServerSocket?

I am not quite happy about this; I think Socket::Data should in fact be private. I guess Socket::Attach(SOCKET) would solve the problem, right?

Subject: Re: Segfaults with One container?
Posted by phirox on Thu, 24 Sep 2009 08:22:03 GMT
View Forum Message <> Reply to Message

luzr wrote on Thu, 24 September 2009 09:43I am not quite happy about this; I think Socket::Data should in fact be private. I guess Socket::Attach(SOCKET) would solve the problem, right?

Such a function would be great, but would need one addition. Socket settings such as linger/blocking/delay will not be saved in the class specific variables, even though they will remain effective on the socket itself. I can find only one that is really used; to determine if to peek or directly read. So the function should at least be Socket::Attach(SOCKET, is_blocking)

Subject: Re: Segfaults with One container?
Posted by mirek on Thu, 24 Sep 2009 09:06:42 GMT

phirox wrote on Thu, 24 September 2009 04:22luzr wrote on Thu, 24 September 2009 09:43I am not quite happy about this; I think Socket::Data should in fact be private. I guess Socket::Attach(SOCKET) would solve the problem, right?

Such a function would be great, but would need one addition. Socket settings such as linger/blocking/delay will not be saved in the class specific variables, even though they will remain effective on the socket itself. I can find only one that is really used; to determine if to peek or directly read. So the function should at least be Socket::Attach(SOCKET, is_blocking)

BTW, I must have missed something, but why you dont just keep using the same Socket after fork?

Mirek

Subject: Re: Segfaults with One container?
Posted by phirox on Thu, 24 Sep 2009 10:25:29 GMT

luzr wrote on Thu, 24 September 2009 11:06
BTW, I must have missed something, but why you dont just keep using the same Socket after fork?

Mirek

Because fork was just an example of one of the methods. I use exec(), which starts a whole new process. I catch the HUP signal which reloads the binary, basically giving the ability to update my program without losing any connections. It does this by saving every class variable that is necessary in a sessions file(xml) and then loading it back in the new binary. Which then resumes the socket connections.

Subject: Re: Segfaults with One container?
Posted by mirek on Fri, 02 Oct 2009 08:01:20 GMT

phirox wrote on Thu, 24 September 2009 06:25luzr wrote on Thu, 24 September 2009 11:06
BTW, I must have missed something, but why you dont just keep using the same Socket after fork?

Mirek

Because fork was just an example of one of the methods. I use exec(), which starts a whole new process. I catch the HUP signal which reloads the binary, basically giving the ability to update my program without losing any connections. It does this by saving every class variable that is necessary in a sessions file(xml) and then loading it back in the new binary. Which then resumes

the socket connections.

Well, Linger and NoDelay are not stored in member variables, so that leaves us with blocking. I was thinking about reading this value from system (on Attach), but it seems like this is not possible in Win32.

Therefore, for now:

void AttachSocket(Socket& socket, SOCKET s, bool blocking)

(Note this is a global function, because Socket is supposed to support SSL sockets too...).

Mirek
