Subject: Program didn't exit in Task Manager after click [X]
Posted by sinpeople on Mon, 26 Oct 2020 01:36:51 GMT
View Forum Message <> Reply to Message

Hi folks,

I have a dialog program which contains a "Start" button to start 2x thread services. It is copied and modified from an existing program. The original program does not have this program but this program has. I tried to isolate the program as the following:
1) Launch program, click [X] directly, program can exit
2) Launch program, click "Start" button, then click [X], program cannot exit.
I could not spot anything suspicious in the callback code of the button handling.
It is the following:

```
 btnStart <<= callback(this, &RegionalCtrl::SpawnTasks);




void RegionalCtrl::SpawnTasks()
{
 //Start Thread Remote Request; The Server Portion of the UDP TCP Service for Incoming Service
 //Request
 Thread udprpc;
 udprpc.Run(callback(this, &RegionalCtrl::UDPRpcServerThread));

 Thread tcprpc;
 tcprpc.Run(callback(this, &RegionalCtrl::TCPRpcServerThread));

 btnStart.Disable();
}

// Listenning here to figure any execution is needed
void RegionalCtrl::UDPRpcServerThread()
{
 UrrServer urr;
 urr.Create(m_nServerPort);
 //Cout() << "URR Ping server\n";
 //while(false)
 for(;;)
 {
  UrrRequest r;
  if(urr.Accept(r))
  {
   UdpRpcCmd(r);
  }
  Sleep(500);
```

```
 }
}

void RegionalCtrl::TCPRpcServerThread()
{
 TcpSocket server;
 if(!server.Listen(m_nServerPort+100, 5)) {
  Cout() << "Unable to initialize server socket!\n";
  SetExitCode(1);
  return;
 }
 Cout() << "Waiting for requests..\n";
 for(;;)
 {
  TcpSocket s;
  if(s.Accept(server)) {
   String w = s.GetLine();
   TcpRpcCmd(w);
//   Cout() << "Request: " << w << " from: " << s.GetPeerAddr() << '\n';
   if(w == "time")
    s.Put(AsString(GetSysTime()));
   else
    s.Put(AsString(3 * atoi(~w)));
   s.Put("\n");
  }
  Sleep(500);
 }
}
```

I stepped into these 2 pieces of code. It didn't return back when the "if" statement is stepped into to get input strings. Comment either one of these 2 threads didn't help, unless comment both of them.

I have to kill the program every time when it is launched outside theIDE. It can exited properly if I launch it within theIDE for debug though. I checked back to the original program. These pieces of code are the same. Headache. Not sure what's the general advise to troubleshoot this kind of "Fail to exit completed from OS" problem.

Thank you very much!

Best Regards
David

Subject: Re: Program didn't exit in Task Manager after click [X]

Posted by mirek on Mon, 26 Oct 2020 07:47:19 GMT
View Forum Message <> Reply to Message

sinpeople wrote on Mon, 26 October 2020 02:36Hi folks,

I have a dialog program which contains a "Start" button to start 2x thread services. It is copied and modified from an existing program. The original program does not have this program but this program has. I tried to isolate the program as the following:
1) Launch program, click [X] directly, program can exit
2) Launch program, click "Start" button, then click [X], program cannot exit.
I could not spot anything suspicious in the callback code of the button handling.
It is the following:

```
 btnStart <<= callback(this, &RegionalCtrl::SpawnTasks);




void RegionalCtrl::SpawnTasks()
{
 //Start Thread Remote Request; The Server Portion of the UDP TCP Service for Incoming Service
 //Request
 Thread udprpc;
 udprpc.Run(callback(this, &RegionalCtrl::UDPRpcServerThread));

 Thread tcprpc;
 tcprpc.Run(callback(this, &RegionalCtrl::TCPRpcServerThread));

 btnStart.Disable();
}

// Listenning here to figure any execution is needed
void RegionalCtrl::UDPRpcServerThread()
{
 UrrServer urr;
 urr.Create(m_nServerPort);
 //Cout() << "URR Ping server\n";
 //while(false)
 for(;;)
 {
  UrrRequest r;
  if(urr.Accept(r))
  {
   UdpRpcCmd(r);
  }
  Sleep(500);
 }
}
```

```
void RegionalCtrl::TCPRpcServerThread()
{
 TcpSocket server;
 if(!server.Listen(m_nServerPort+100, 5)) {
  Cout() << "Unable to initialize server socket!\n";
  SetExitCode(1);
  return;
 }
 Cout() << "Waiting for requests..\n";
 for(;;)
 {
  TcpSocket s;
  if(s.Accept(server)) {
   String w = s.GetLine();
   TcpRpcCmd(w);
//   Cout() << "Request: " << w << " from: " << s.GetPeerAddr() << '\n';
   if(w == "time")
    s.Put(AsString(GetSysTime()));
   else
    s.Put(AsString(3 * atoi(~w)));
   s.Put("\n");
  }
  Sleep(500);
 }
}
```

I stepped into these 2 pieces of code. It didn't return back when the "if" statement is stepped into to get input strings. Comment either one of these 2 threads didn't help, unless comment both of them.

I have to kill the program every time when it is launched outside theIDE. It can exited properly if I launch it within theIDE for debug though. I checked back to the original program. These pieces of code are the same. Headache. Not sure what's the general advise to troubleshoot this kind of "Fail to exit completed from OS" problem.

Thank you very much!

Best Regards
David

Would it possible to post the whole package? I do not even know which "if" statement do you have in mind...

Mirek

Subject: Re: Program didn't exit in Task Manager after click [X]
Posted by sinpeople on Mon, 26 Oct 2020 09:01:17 GMT
View Forum Message <> Reply to Message

Hi

Thank you for your reply. I have attached the zip package. The .xml configurations are also included.

Extract the zip to MyApp folder to run the RegionalCtrl

Best Regards
David

File Attachments

1) MyApps.zip, downloaded 257 times

Subject: Re: Program didn't exit in Task Manager after click [X]
Posted by peterh on Thu, 05 Nov 2020 12:51:43 GMT
View Forum Message <> Reply to Message

While in the debugger, the program doesnt stop either.
If I click [x], the window is closed, but the stop button of the debugger stays red and CPU-load in Windows Task manager is high.
In the "threads" section of the debugger it can be seen, several threads are still running and waiting inside windows e.g. "WaitForMultipleObjects".
If the red stop button of the debugger is clicked, the threads are terminated, so the debugger can do this, but the program obviously does not successfully terminate all threads.

Maybe, some threads are blocked on Kernel objects by blocking I/O? In this case these cannot be immediately terminated.
Because the threads can be killed by Windows Task Manager and/or debugger, there is some hope you can kill them programmatically before the main GUI thread terminates.
This is of course a brute force solution, maybe there are better solutions, but I cannot give advice, because I am not specialist for this matter.

Good luck!
P.S. If the programm is running normally it consumes a lot of CPU while doing nothing than waiting.
Maybe this is a side effect of my system configuration or of debugger, I dont know.

Subject: Re: Program didn't exit in Task Manager after click [X]
Posted by peterh on Sat, 07 Nov 2020 06:53:23 GMT
View Forum Message <> Reply to Message

For Windows, this solves the problem, so far I can see:

```
GUI_APP_MAIN
{
 MainWindow().Sizeable().Run();
 ExitProcess(0);
}
```

ExitProcess() is a Windows system call that terminates all blocking IO and kills all threads, including main thread and process.
https:// docs.microsoft.com/en-us/windows/win32/procthread/terminatin g-a-process

---

Subject: Re: Program didn't exit in Task Manager after click [X]
Posted by Klugier on Sat, 07 Nov 2020 10:37:41 GMT
View Forum Message <> Reply to Message

Hello Peterh,

The trick with ExitProcess(0) seems like the hack and workaround. In good written application it is never needed. At the end of the program you should clean all resources, threads and processes. If for some reasons you can not close your application then you need to do the investigation and fix your code. Moreover ExitProcess(0) is Windows only and using it you will lose portability.

Klugier

Subject: Re: Program didn't exit in Task Manager after click [X]
Posted by peterh on Sat, 07 Nov 2020 11:01:57 GMT
View Forum Message <> Reply to Message

So far I understand it, Windows will not terminate a thread forcefully, but will inject code to the thread, so the thread will terminate itself cleanly, when it runs this code.
This doesnt work, if the thread is IO waiting, because the thread doesnt run. In this case, if there is no IO timeout, the thread cannot self terminate.
So first the IO must be unblocked and ExitProcess does this for all IO in the process and threads.
Also DLLS are detached cleanly from the threads.

I think, there is nothing wrong with this. Yes, it is system specific, but multithreading is always somewhat specific to the OS.
For console processes ExitProcess() is called automatically by Windows when Ctrl C is pressed, but not for GUI processes.
This is not a call from the debug API but from normal process API.

A cleaner way would be to set a termination flag which is polled by the threads, and I already tried this with no success.

The threads are blocked in some TCP Network call which obviously has no timeout. Maybe this can be improved, but I dont know.
I dont want to dig so deep into it.

Subject: Re: Program didn't exit in Task Manager after click [X]
Posted by peterh on Sat, 07 Nov 2020 13:03:39 GMT
View Forum Message <> Reply to Message

I have tracked down this by singlestep to some degree:

Core, Socket.cpp
Comments starting "//!!!" are by me.

```
bool TcpSocket::Accept(TcpSocket& ls)
{
 Close();
 Init();
 Reset();
 ASSERT(ls.IsOpen());
 int et = GetEndTime();
 for(;;) {
  int h = ls.GetTimeout();
  bool b = ls.Timeout(timeout).Wait(WAIT_READ, et);
  ls.Timeout(h); //!!! If I step into this thread is switched, the window appears and I am unable to step further
  if(!b) // timeout
   return false;
  socket = accept(ls.GetSOCKET(), NULL, NULL);
  if(socket != INVALID_SOCKET)
   break;
  if(!WouldBlock() && GetErrorCode() != SOCKERR(EINTR)) { // In prefork condition, Wait is not enough, as other process can accept
   SetSockError("accept");
   return false;
  }
 }
 mode = ACCEPT;
 return SetupSocket();
}
```

Im unable to singlestep this somewhere inside this code there happens a threadswitch and the window appears.
This blocks the thread. The problem could be here.
I admit, I dont understand it fully have no clue about Networking code. :blush:

So far I found, timeout in the socket is set to zero or negative - maybe uninitialized - , this means

infinite timeout or blocking, so this is a problem in the user code.
I am not sure about it.

Subject: Re: Program didn't exit in Task Manager after click [X]
Posted by peterh on Sat, 07 Nov 2020 19:25:35 GMT
View Forum Message <> Reply to Message

While debugging I find something suspicious in Core Socket.cpp at line 287 in the constructor of TcpSocket:

```
TcpSocket::TcpSocket()
{
 ClearError();
 Reset();
 timeout = global_timeout = start_time = Null; //<---- Confusion about NULL and Null?
 waitstep = 10;
 asn1 = false;
}
```

"Null" is the special "Nuller" of upp and evaluates to 0x8000 0000 here.
If "(int) Null" is propagated to signed 64 bit it gives 0xffff ffff 8000 0000, if "NULL" is propagated it gives 0x0000 0000 0000 0000.
(int) Null / 10000000 will NOT give zero microseconds. There are wrong timeout calculations in this program, and possibly in Upp source.
This is very dangerous, probably you must turn on warnings about unintended implicite int int64 conversions for compilation.
This leads to different behaviour (which I have observed) , when the program is compiled to 32 or 64 bit and I believe there is a bit of confusion about NULL and Null.
However fixing this, improves, but does not fix the current problem.

Subject: Re: Program didn't exit in Task Manager after click [X]
Posted by Oblivion on Sun, 08 Nov 2020 12:03:28 GMT
View Forum Message <> Reply to Message

Hello peterh,

Just to clarify TcpSocket behaviour and its relation to Upp::Null:

Upp::TcpSocket uses a "pseudo-blocking" mechanism.
This means, the platform specific, underlying socket is always in non-blocking mode.
This is by design. From the user's POV, this allow us to use the TcpSocket in blocking, non-blocking, or in a time-constrained mode, easily, depending on our specific use-case.

Enter "Null" value.

Upp::Null, as you've noticed, is a Upp-specific value. It is already defined for int, int64, double and bool types (and can be used wtih other types that define a Nuller). So you should make no assumptions about it.

Null / 100000 can semantically be considered an "undefined behaviour", because we are trying to divide something that doesn't exist.

Hence, timeout = Null, means that there is no timeout.

```
int   foo1 = Null; // Ok
int64 foo2 = (int) Null; // Not OK! This can be considered a user error.
int64 foo3 = Null; // OK.
```

Null, assigned to timeout (int), in this context, is used to put the TcpSocket in blocking mode.
OTOH, "0" (int) is used to put the TcpSocket into non-blocking mode.
timeout > 0 && timeout <= INT_MAX is a time-constraint (in miliseconds) for a given TcpSocket operation.

Best regards,
Oblivion

Subject: Re: Program didn't exit in Task Manager after click [X]
Posted by peterh on Sun, 08 Nov 2020 12:39:45 GMT
View Forum Message <> Reply to Message

Thank you Oblivion,
Maybe another name instead of "Null" e.g. "Empty" or "Nil" or "Void" would be better.
NULL and Null are easily confused by beginners and due to the polymorphic nature of Null, this semantical error is not captured by the compiler.
This Naming opens all doors for beginner errors. And I think even proffessionals are not immune against these type of error :cry:
This happens for example in Urr.cpp
Watchout to the calculations about timeout here:

Factum is: The timeout in this program doesnt work, the network communication blocks and the program is therefore unable to terminate the threads.
Another factum is: the timeout behaviour changes (but is not correct) when the progam is compiled for a 32 bit target.

I cannot judge were the reason for this is, in Upp or in RegionalCtrl or in both.

File Attachments
1) RegionalCtrl.png, downloaded 633 times

Subject: Re: Program didn't exit in Task Manager after click [X]
Posted by Oblivion on Sun, 08 Nov 2020 13:32:52 GMT
View Forum Message <> Reply to Message

Hello peterh,

You are definitely right, it can be confusing sometimes. That is why we have a warning for beginners. :)

https://www.ultimatepp.org/srcdoc$Core$Caveats_en-us.html

Maybe we should expand the explanation...

About Urr package: I have never used it but indeed there seems to be a problem in Rcv() function. According to ln: 18, it expects Upp::Null. But on ln 14-15, it does not check the timeout value for Null.
From what I understand, they should be:

```
time_out.tv_sec = Nvl(timeout, 0) / 1000;
time_out.tv_usec = Nvl(timeout, 0) % 1000 * 1000;
```

As a side note, I personally wouldn't consider a good practice assigning NULL to anything other than pointers.
(I don't even use NULL fpr pointers anymore as we have nullptr) :)

Best regards,
Oblivion