Subject: Can't use MT to capture console output
Posted by cbpporter on Thu, 28 Nov 2024 20:41:02 GMT
View Forum Message <> Reply to Message

Hi!

So I have an ancient code I'm rewriting with modern U++ and parts of it execute commands in the console and capture this output to feed it to the GUI. Nothing special.

This code used to work perfectly, including when the process hangs or prints out stuff in an infinite loop. Then the GUI can kill it.

But something has changed with the MT implementation of U++ or I'm doing something very stupid and I'm not aware of it, but now the same code hangs the UI. I looked over the docs and samples and found nothing changed.

I added a zip with a package to test.

Just change the path within:
Thread().Run(callback3(ExecutableThread, this, "c:\\temp\\test.pak\\test4.exe", false));

If the exe prints something like "0\n" in a loop, the GUI hangs.

Thank you!

PS: wow, it has been years since I posted one of these: a ZIP with a package to minimally illustrate an issue :d.

File Attachments
1) testmt.zip, downloaded 185 times

Subject: Re: Can't use MT to capture console output
Posted by cbpporter on Thu, 28 Nov 2024 21:07:51 GMT
View Forum Message <> Reply to Message

Hmmm, adding a Sleep(5) or 10 in ExecutableThread makes the GUI responsive again.

```
while (globalExecutor.Read(t)) {
 if (t.GetCount()) {
  //DUMP(t);
  PostCallback(callback1(test, &Test::AddOutputLine, t));
  Sleep(5);
 }
}
```

The goal is for it to be responsive and the user to figure out that it hanged, so they can stop the process.

Subject: Re: Can't use MT to capture console output
Posted by cbpporter on Sat, 30 Nov 2024 13:51:44 GMT
View Forum Message <> Reply to Message

For now I'm running with the Sleep option but this used to be one of strong points of U++. You could capture giant MB long console outputs tot he GUI with virtually no delay or lag.

I'll take my sample and try to use some U++ form 5+ years I have around and see if it still behaves the same.

Subject: Re: Can't use MT to capture console output
Posted by cbpporter on Wed, 04 Dec 2024 09:34:45 GMT
View Forum Message <> Reply to Message

I've tested on U++ 12587 and the test case I uploaded, unmodified, with an exe that prints 100k lines of code finishes almost instantly and gets captured, no Sleep needed.

With latest U++ this hangs or lags, as described.

With 1000 times as many outputs, both hang. But a Sleep of (1) on the old U++ version makes it super responsive, much better than Sleep(10) on the new U++.

Also, the old U++ is on an old PC. Read slow. I had to revive it from storage just for this test. The new one is on a cutting edge (from 3-4 years ago) Intel CPU. So the new U++ on new PC is more laggy than old U++ on old PC.

This reminds me of many years ago, when I was first trying to get this going and it was a complete mess. My solutions didn't work. If I remember correctly, I took inspiration from TheIDE sources and this single line fixed all the issues:

```
 void Append(const String& s) {
  Insert(total, s); // THIS
 }
```

Inserting to "total" position and only there, made it work. I'm guessing that turns it into an "append", not an "insert".

I'll try and find an ever older U++ than 12587, and maybe one in the middle :). But I doubt I'll find one.

Subject: Re: Can't use MT to capture console output

Posted by cbpporter on Wed, 04 Dec 2024 09:55:24 GMT
View Forum Message <> Reply to Message

Ha, the issue is more complicated than it seems.

I have two programs outputting 100k numbers, one with STD C++ and one with a custom "cout" solution.

One is much slower than the other for no good reason, just a loop printing numbers from 0 to 99K.

The size of the Output console is another variable. With STD C++ there are no noticeable differences, but with the custom, Console capture is very laggy.

I'll dig into this more.

This is probably something very stupid, like it usually is with MT issues...

Subject: Re: Can't use MT to capture console output
Posted by cbpporter on Sat, 02 Aug 2025 11:57:37 GMT
View Forum Message <> Reply to Message

I finally had time to do another full run of investigations.

While the PostCallback method is still not performing acceptably without a Sleep, I switched over to not using it and adding a GuiLock __.

This solved many of the problems. Still might be a bit slower than it used to be in the old days and builds, but barely.

Now I just need to rewrite the Thread stop conditions for this new approach, since there is a menu option to kill the thread and this one crashes when using GuiLock. Because the thread still keeps going.

A bit hacky my current method, need to rethink it. I have found no way to kill a Thread directly.