Using the "gu" debugger command to find the infinite loop

devblogs.microsoft.com/oldnewthing/20070426-00

April 26, 2007

Raymond Chen

Somebody says, “Your program is consuming 100% CPU” and hands you a debug session. Usually, this happens because one thread has gotten stuck in an infinite loop. And if you’re lucky it’s the type of infinite loop that’s easy to diagnose because it’s just one function that isn’t returning. (The more complicated types are where a function does some work and then returns, and then some of that work has a delayed effect that causes the function to run again, and so on.) Let’s assume we’re lucky because, well, debugging is an exercise in optimism.

The first step is to find the thread that is using all the CPU. That’s actually pretty easy with the help of the `!runaway` debugger extension.

```
0:011> !runaway
 User Mode Time
 Thread    Time
 192c      0 days 0:05:22.457
 1384      0 days 0:00:16.063
 14ac      0 days 0:00:08.392
 48c       0 days 0:00:03.955
 1db0      0 days 0:00:00.010
 1888      0 days 0:00:00.010
 1078      0 days 0:00:00.000
 1470      0 days 0:00:00.000
 1f84      0 days 0:00:00.000
 1d60      0 days 0:00:00.000
 1850      0 days 0:00:00.000
 134c      0 days 0:00:00.000
 19fc      0 days 0:00:00.000
 1b4       0 days 0:00:00.000
```

Wow, thread 0x192c has sure used a lot of CPU time, but that doesn’t mean that it’s the thread that is in a 100% CPU loop, because the CPU time is cumulative over the lifetime of the thread. Maybe that thread has a lot of CPU time because it’s been around the longest. What you need to do is resume execution for a little while, then break in again and see whose CPU time has **increased**.

```
0:011> g
^C
(1928.1d34): Break instruction exception - code 80000003 (first chance)
eax=7ffd9000 ebx=00000001 ecx=00000002 edx=00000003 esi=00000004 edi=00000005
eip=7c901230 esp=0124ffcc ebp=0124fff4 iopl=0         nv up ei pl zr na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=0038  gs=0000             efl=00000246
ntdll!DbgBreakPoint:
7c901230 cc              int     3
0:011> !runaway
 User Mode Time
 Thread    Time
 192c      0 days 0:05:23.679
 1384      0 days 0:00:16.063
 14ac      0 days 0:00:08.392
 48c       0 days 0:00:03.955
 1db0      0 days 0:00:00.010
 1888      0 days 0:00:00.010
 1078      0 days 0:00:00.000
 1470      0 days 0:00:00.000
 1ea4      0 days 0:00:00.000
 1d60      0 days 0:00:00.000
 1850      0 days 0:00:00.000
 134c      0 days 0:00:00.000
 19fc      0 days 0:00:00.000
 1b4       0 days 0:00:00.000
```

Aha, we see that thread 0x192c is the only one who gained any noticeable amount of CPU time. That's probably the one that's responsible for the 100% CPU usage.

```
0:011> ~~[192c]s
eax=00000000 ebx=77d5e581 ecx=0012daa0 edx=0000000c esi=01d18140 edi=00000000
eip=77d5e590 esp=0012da78 ebp=0012da88 iopl=0         nv up ei pl zr na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
USER32!FindWindowA+0xf:
77d5e590 50              push    eax
ChildEBP RetAddr
0012da88 1000714b USER32!FindWindowA+0xf
0012dbc8 100061f3 ABC!CAlpha::FindTarget+0x27f
0012dbf4 603517e8 ABC!CBeta::TransferData+0x18a
0012dc28 10002d9d DEF!CGamma:TransferData+0xc
0012dc48 00505303 ABC!CBeta::BeginAsync+0x51
0012dc5c 0090a21a GHI!CPrintSession::Open+0x2a51
0012dd20 009099d8 GHI!CPrintSession::Init+0x252
0012e060 009097e6 GHI!CPrintOptions::GetSettings+0x24a
0012e0a4 0090973d GHI!CPrintOptions::OpenSettings+0x248
0012e130 00909664 GHI!CDocumentMenu::OnInvoke+0x24
...
```

Now this is where the magical "gu" command comes in. You type "gu" to run the current function until it returns. If you get another prompt back, then type "gu" again, to run **that** function until it returns. And so on, until you find the function that doesn't return.

That’s the one with the infinite loop.

Now you can start investigating why that function is stuck.

Raymond Chen

Follow