

Re: Frame-based exception handling problem on Server 2008

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.kernel/2008-02/msg00255.htm>

- *From:* "Ivan Brugiolo [MSFT]" <ivanbrug@xxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Fri, 22 Feb 2008 08:24:31 -0800
-

Can you elaborate on the NX options of the bcdedit configuration ?
My take is that the server SKU has NX enabled by default, that implies strict validation of the SEH-Handler against the table generated by the compiler,
while the client SKU has a looser validation, and your code gets by.

Then, could you use a ntsd/cdb/windbg based debugger to report the stack ?
Those debuggers have the ability to use public PDBs to give meaningful stack traces.

--

--

This posting is provided "AS IS" with no warranties, and confers no rights.
Use of any included script samples are subject to the terms specified at
<http://www.microsoft.com/info/copyright.htm>

"Corinna Vinschen" <corinna@xxxxxxxxxxxxxxxxxxxx> wrote in message
[news:fpmnif\\$7i4\\$1@xxxxxxxxxxxxxxxxxxxx](mailto:news:fpmnif$7i4$1@xxxxxxxxxxxxxxxxxxxx)

Hi Jeffery,

"Jeffrey Tan[MSFT]" wrote:

Hi Corinna,

It seems like you have done some hack for the Windows SEH which is a bit like the Visual C++ runtime SEH wrapper(which also wraps all the exception list entries in one function in one place for table-driver dispatch).

Mmh, that's possible, but I'm not quite sure about this. Cygwin's using GCC as compiler and Cygwin is its own runtime environment, so you probably know indefinitely more about the Visual-C++ runtime. :-)

Re: Frame-based exception handling problem on Server 2008

Now, it seems that the problem is that the Windows2008 exception dispatcher did not call your registered exception_list entry at FS:[0], yes?

That seems to be the case, yes. If I set a breakpoint to our exception handler, it's not called. Instead, 2008 hangs, feeding on the CPU. As soon as I change the exception_list entry so that the prev pointer points to the default handler, our exception handler is called just fine.

But the drawback is, it's only called once for a given exception. If the exception handler returns without rectifying the cause for the exception, the next time the default handler is called. That's not what we want for hopefully obvious reasons.

Have you tried to set a breakpoint in the user-mode exception dispatcher(should be in ntdll) in debugger? If it is called, then you can step through to understand its algorithm and find out why your handler is not called.

I'm sorry, but I don't know exactly how to do that. I can't set a breakpoint in ntdll from within GDB. I get an I/O error. However, what I did was to step through and it appears that an endless loop is run in this area of ntdll.dll:

Dump of assembler code from 0x76f00d6d to 0x76f00da2:

```
0x76f00d6d <ntdll!RtlTimeToTimeFields+74226>: cmp -0x8(%ebp),%ebx
0x76f00d70 <ntdll!RtlTimeToTimeFields+74229>: jb 0x76eedf10
<ntdll!EtwSetMark+495>
0x76f00d76 <ntdll!RtlTimeToTimeFields+74235>: lea 0x8(%ebx),%eax
0x76f00d79 <ntdll!RtlTimeToTimeFields+74238>: cmp -0xc(%ebp),%eax
0x76f00d7c <ntdll!RtlTimeToTimeFields+74241>: ja 0x76eedf10
<ntdll!EtwSetMark+495>
0x76f00d82 <ntdll!RtlTimeToTimeFields+74247>: test $0x3,%bl
0x76f00d85 <ntdll!RtlTimeToTimeFields+74250>: jne 0x76eedf10
<ntdll!EtwSetMark+495>
0x76f00d8b <ntdll!RtlTimeToTimeFields+74256>: mov 0x4(%ebx),%eax
0x76f00d8e <ntdll!RtlTimeToTimeFields+74259>: cmp -0x8(%ebp),%eax
0x76f00d91 <ntdll!RtlTimeToTimeFields+74262>: jb 0x76f00d9c
<ntdll!RtlTimeToTimeFields+74273>
0x76f00d93 <ntdll!RtlTimeToTimeFields+74264>: cmp -0xc(%ebp),%eax
0x76f00d96 <ntdll!RtlTimeToTimeFields+74267>: jb 0x76eedf10
<ntdll!EtwSetMark+495>
0x76f00d9c <ntdll!RtlTimeToTimeFields+74273>: mov (%ebx),%ebx
0x76f00d9e <ntdll!RtlTimeToTimeFields+74275>: cmp %edi,%ebx
0x76f00da0 <ntdll!RtlTimeToTimeFields+74277>: jne 0x76f00d6d
<ntdll!RtlTimeToTimeFields+74226>
```

Re: Frame-based exception handling problem on Server 2008

Is it possible for you to provide a simple sample project to reproduce this problem? If I can reproduce, we can work on it more efficiently. Also, I will try to setup a Windows2008 machine for testing it. Thanks.

Below you'll find the source code for a self-contained testcase, which allows to reproduce the behaviour. It just installs the exception handler on the stack in the main() function. That should do it for testing purposes.

However, it's using GCC syntax. I built it under Cygwin like this:

```
gcc -g -mno-cygwin -o exc_test exc_test.c
```

The same should work fine without `-mno-cygwin` under MingW. I don't know how you can build it in Visual-C++. I assume it should just work if you replace the GCC assembler syntax in the line

```
extern exception_list_t *_except_list __asm__ ("%fs:0");
```

with the appropriate Visual-C++ syntax.

Without argument, the `exception_list` entry is installed pointing the `prev` pointer to itself. With any argument, the `exception_list` entry is installed with `prev` pointing to the previous exception list entry (the default handler).

On 2008, running the example without argument does never call the exception handler, but it hangs. On (for instance) XP it calls the own exception handler over and over again, which is the desired behaviour. Running the example with any argument calls the own exception handler once, but calls the default handler the next time the division by zero occurs.

Ok, here's the self-contained testcase:

```
===== SNIP =====  
#include <stdio.h>  
#include <windows.h>  
  
extern DWORD __stdcall RtlUnwind (void *, void *, EXCEPTION_RECORD *, void  
*);  
  
struct _exception_list;  
  
typedef int (*exception_handler_t) (EXCEPTION_RECORD *,  
struct _exception_list *,  
CONTEXT *,  
void *);
```

Re: Frame-based exception handling problem on Server 2008

```
typedef struct _exception_list
{
    struct _exception_list *prev;
    exception_handler_t handler;
} exception_list_t;

typedef struct _cygtls
{
    exception_list_t el;
} cygtls_t;

extern exception_list_t *_except_list __asm__ ("%fs:0");

void
init_exception_handler (cygtls_t *tls, exception_handler_t eh, int
do_loop)
{
    tls->el.handler = eh;
    if (do_loop)
        tls->el.prev = &tls->el; // loop back to itself
    else
        tls->el.prev = _except_list; // Just point to default handler
    _except_list = &tls->el;
}

int
handle_exception (EXCEPTION_RECORD *e, exception_list_t *frame,
CONTEXT *c, void *dummy)
{
    fputs ("In exception_handler\n", stderr);
    RtlUnwind (frame, (void *) c->Eip, e, 0);
    return 0;
}

int
func ()
{
    return 1/0;
}

int
main (int argc, char **argv)
{
    cygtls_t _my_tls;
    init_exception_handler (&_my_tls, handle_exception, argc <= 1);
    fputs ("Before func\n", stderr);
    func ();
    fputs ("After func\n", stderr);
    return 0;
}

===== SNAP =====
```

Re: Frame-based exception handling problem on Server 2008

Corinna

--

Corinna Vinschen
Cygwin Project Co-Leader
Red Hat