

Re: SetUnhandledExceptionFilter

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.kernel/2004-05/0582.html>

From: Mikko Noromaa (*mikkon_at_excelsql.com*)

Date: 05/25/04

Date: Tue, 25 May 2004 14:57:17 +0300

Hi,

> *An access violation (or unhandled exception) may often cause
> functionality in the add-in not to work properly.*

If you add `__try .. __except` blocks around all the code of your exported functions, you should be able to catch 99% of exceptions that happen because of your code. I use this approach in an add-in that is loaded as a COM object. I use ATL so that part of the functionality is not covered by the `__try .. __except` blocks. This is normally enough, but if you don't trust ATL code, then you might have to code everything yourself (Unknown methods etc).

When an unhandled exception occurs (caught by my top-level handlers), I create a simple stack trace without using any debugging functions. This trace works by checking all `DWORD`'s in the stack for executable code in my module. The function (`OutputSimpleStackDump`) is included at the end of this post. This function has the advantage of working on optimized release code and is operating-system independent (not tested though).

When I get a crash report, I load the same version of the program in my debugger, and start looking at the reported addresses. With a little help from the MAP file, I usually find the offending function pretty quickly. The simple stack dump has items that are not real addresses, but they are usually easy to skip when examining the code. Obviously, you need to have some skills in reading assembly code to be able to do all this...

`OutputSimpleStackDump` function follows. You may have to modify some parts (e.g. `hModules[]` & `0xFFFF0000`) depending on your application.

```
void CDebug::OutputSimpleStackDump(PCONTEXT Context)
{
    DWORD Esp, StartFrame, PrevFrame;
    DWORD hModules[2];
    WCHAR TraceLine[2048]; // Well enough for 20 * 18 characters...
    int TraceLineLen=0;
    int NumFrames=0;
```

```
hModules[0]=(DWORD)GetModuleHandle(_T(IMAGE_NAME));
hModules[1]=(DWORD)GetModuleHandle(_T("My_other_module.dll"));

if (Context) {
    Esp=Context->Esp;
    TraceLineLen+=swprintf(&TraceLine[TraceLineLen], L"0x%08X", Context->Eip);
}
else {
    __asm {
        mov Esp, esp;
    }
}

// Limit our stack trace to 20 levels, or 100 kB.
StartFrame=PrevFrame=Esp;
while (NumFrames<20 && Esp-StartFrame<100*1024 && !IsBadReadPtr((void
*)Esp,4)) {
    DWORD Ptr=*(DWORD *)Esp;
    if ((Ptr&0xFFF00000)==(hModules[0]&0xFFF00000) ||
(Ptr&0xFFF00000)==(hModules[1]&0xFFF00000)) {
        MEMORY_BASIC_INFORMATION MemInfo={0};

        VirtualQuery ((void *)Ptr, &MemInfo, sizeof(MemInfo));
        if (MemInfo.State==MEM_COMMIT && MemInfo.Type==MEM_IMAGE &&
(MemInfo.Protect&(PAGE_EXECUTE|PAGE_EXECUTE_READ|PAGE_EXECUTE_READWRITE|PAGE
_EXECUTE_WRITECOPY))) {
            TraceLineLen+=swprintf(&TraceLine[TraceLineLen], L" (%d) 0x%08X",
Esp-PrevFrame, Ptr);
            NumFrames++;
            PrevFrame=Esp;
        }
    }

    Esp+=4;
}

if (TraceLineLen>0) {
    TraceLine[TraceLineLen]=0;
    PrintOut (TraceLine);
}
}

--
Mikko Noromaa (mikkon@excelsql.com)
- SQL in Excel, check out ExcelsQL! - see http://www.excelsql.com -
"Werner B. Strydom" <bloudraak@hotmail.com> wrote in message
news:14047B11-D0AC-4852-A998-44B5FF33F665@microsoft.com...
> Hi Pavel,
>
> We want to know whether it's our add-ons that cause the third party
application to misbehave. It opens possibilities to diagnose and fix the
issue. Perhaps it can save us some money.
```

microsoft.public.win32.programmer.kernel: Re: SetUnhandledExceptionFilter

>
> An access violation (or unhandled exception) may often cause functionality in the add-in not to work properly. Support logs a defect that the add-in intermittently doesn't work. We have a hard time reproducing it, because we don't have the exact environment of the user. We may lose the sale, or in turn lose any profit diagnosing the problem. An access violation in our code is not common, however they are the most expensive and damaging of all defects to diagnose and fix.
>
> It may be safe to assume that if one of our dlls is in the call stack, that we have something to do with it, hence using StackWalk64. We'll log that fact. If it's not the case, the default behaviour of windows comes to effect.
>
> Take care
> Werner
>