Ruby - Bug #1230

ESP Stack Corruption on Windows

03/01/2009 05:17 PM - cfis (Charlie Savage)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:	1.9.1		
ruby -v:	ruby 1.9.2dev (2009-03-01) [i386-mswin32_90]	**Backport:**	

Description

=begin
Build Ruby with -RTC1 flag on Windows with VC2008 (same issues will also apply with Mingw).

You'll almost immediately get a runtime assertion failure:

Run-Time Check Failure #0 - The value of ESP was not properly saved across a function call. This is usually a result of calling a function declared with one calling convention with a function pointer declared with a different calling convention.

Looking into this, at startup Ruby calls rubygems/config_file which int turn calls the WindowsAPI method SHGetFolderPath. It doesn't work because of three different bugs:

1. Win32API uses the cdecl calling convention to call the WindowsAPI. This is wrong, the windows API uses the STDCALL convention for the most part. The problem is in Win32api.rb, where def initialize method has this line:

   @func = DL::CFunc.new(handle[func], TYPEMAP[export.tr("VPpNnLlIi", "0SSI")], func)

The fix is easy:

```
@func = DL::CFunc.new(handle[func], TYPEMAP[export.tr("VPpNnLlIi", "0SSI")], func, :stdcall)
```

1. Event with that fix, the problem still occurs. Its because DL::CFunc does not call STDCALL methods correctly.

For STDCALL methods, it defines this macro:

DECL_FUNC_STDCALL(f,void,DLSTACK_PROTO##n) = cfunc->ptr; \

When this macro gets expanded, it adds "..." to the end of the argument list. That works for CDECL, but not STDCALL because STDCALL does not support variable argument lengths. The correct code is:

DECL_FUNC_STDCALL(f,void,DLSTACK_PROTO##n##_) = cfunc->ptr; \

1. That then runs into the third bug in dl.h. That file defines:

#define DLSTACK_PROTO0
#define DLSTACK_PROTO1_ DLSTACK_TYPE
#define DLSTACK_PROTO2_ DLSTACK_PROTO1_, DLSTACK_TYPE

Notice the first line is wrong, it should be:

#define DLSTACK_PROTO0_

With all three changes applied, the assertion error is fixed.
=end

History

#1 - 03/01/2009 07:07 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

- % Done changed from 0 to 100

=begin
Applied in changeset r22690.
=end

#2 - 03/02/2009 03:59 AM - cfis (Charlie Savage)

=begin
Hi Nobu,

Thanks for applying the patch. Two comments though.

First, you changed:

SHGetFolderPath = Win32API.new 'shell32', 'SHGetFolderPath', 'LLLLP', 'L'

To:

SHGetFolderPath = Win32API.new 'shell32', 'SHGetFolderPath', 'PLPLP', 'L', :stdcall

But looking at Win32api it only takes 4 parameters in its initializer:

def initialize(dllname, func, import, export = "0")

So you can you pass the extra :stdcall?

Second, since the windows api almost always uses the stdcall convention, it seems like changing win32api to assume that would be the right thing to do. Currently it assumes cdelc.

Thoughts? Are you worried about backwards compatibility. What does Ruby 1.8.x do?

Charlie
=end

#3 - 03/02/2009 02:49 PM - nobu (Nobuyoshi Nakada)

=begin
Hi,

At Mon, 2 Mar 2009 03:58:32 +0900,
Charlie Savage wrote in [ruby-core:22613]:

> But looking at Win32api it only takes 4 parameters in its initializer:
>
> def initialize(dllname, func, import, export = "0")
>
> So you can you pass the extra :stdcall?

See r22695.

--
Nobu Nakada

=end

Files

stdcall.patch	3.06 KB	03/01/2009	cfis (Charlie Savage)