

Re: Windows Procedural Programming

Source: <http://coding.derkeiler.com/Archive/General/comp.programming/2005-07/msg00677.html>

- *From:* "Phlip" <phlip_cpp@xxxxxxxxxx>
 - *Date:* Mon, 25 Jul 2005 05:12:55 GMT
-

Joe Butler wrote:

- > One minute [Alf is] saying that a Windows program does not have a WinMain
- > instead of a main, and the next you are saying that main is a requirement
- > of
- > the standard and that it must be used?
- >
- > You seem to be confused.
- >
- > For GUI apps, I've always used WinMain.
- > for CUI app, I've always used main.

It's a gross simplification to say WinMain is for GUIs and main is for CUIs.
(I'm aware you are not saying that. Alf is saying the simplification is gross.)

- > Simple as that.

Not really. In Release mode, I prefer my Win32 GUI programs to divorce from the command line, so I link them with `/subsystem:WINDOWS`. This, in turn, makes the linker look for one of the precursors for WinMain. Then when I compile in Debug mode, I prefer 'cout' for trace statements, so I link with `/subsystem:CONSOLE`. The linker then looks for the wrong kind of main, and I must supply another one.

This is incredibly simpler than the systems that link with `/subsystem:WINDOWS`, then when they need a console for trace statements they bend over backwards to reconstruct one from scratch, and then either ignore `<iostreams>` or construct an alternate 'cout' from scratch.

And sometimes I write a normal `main()`, and then generate windows within it, in between console operations. They work. And because I use a good library, not MFC, I can treat windows as objects without MFC eating my `main()` and limiting my options.

Either way, `main()` and `WinMain()` are only parts of the system that fit together to make a program consular or windular.

Then there's the question which of these situations is C++ Standard

Re: Windows Procedural Programming

compliant. That's only a question: You get the correct opcodes either way. C++ programmers must at least understand the Standard, to continuously track their program's compliance, so they know when they are pushing its envelop. This is a point of pride among C++ programmers.

> You're not actually trying to be helpful, are you. You're just trying to be
> smart. I've had the misfortune of working with people like you in the
> past.
> Sorry.

Joe, this is Alf. Alf, Joe.

> I understand what you are saying. I'll get around to looking at this ruby
> thing you keep pushing – on a virtual machine. Does it hook into Windows,
> or is it self-contained in one's installation?

What are you saying? Ruby.exe is a dumb program that eats text files and follows their instructions. No sandbox, virtual machine, or other fru-fru involved.

But one can substitute "Python", "PHP", "TCL", or other light but OO scripting languages for any of my Ruby rantings.

> Are there any non-trivial apps that you can point to as demonstrations of
> it's usefulness?

No. Only trivial ones.

<a beat>

;–)

> Afterall, in my experience of, say MFC (which MFC'ers
> swear by), anything that you want to do that's 'outside of the box' can
> become a right royal pain trying to work around MFC's internals. So, I
> wonder if it's the same with Ruby.

This is apples-and-oranges. Ruby is a language. MFC is a library. Ruby, like other scripting languages, supports several GUI toolkits, each with strengths and weaknesses. I think that Ruby/Tk lets you drill down to an HWND and run raw window commands on it. You can also call raw Win32 directly.

> Looks good to managers cos you can go:
> click, click, clickety click. And there's the basis of our app. "Bloody
> hell! This will save us tons of development effort.", thinks manager.
> Hmm... first mistake has been made on that project.

That's wizard-oriented programming. Per all my other posts, I prefer unit tests, even for GUIs. They help a project `_sustain_` a project as it grows,

Re: Windows Procedural Programming

so you can follow your demo up with results.

- > Also, you might like to know that with quite a small class (or module in C),
- > about 60 lines of non-trivial code, I can create a window with 1 line line
- > of code in my main app and add controls to it with 1 line of code too
- > (dynamically) – there really is NO magic involved.

Uh, except someone else wrote Tk for you, and a huge community supports it. Same for FOX, Qt, FLTK, etc.

--

Phlip

<http://www.c2.com/cgi/wiki?ZeekLand>

- *Follow-Ups:*

- ◆ **[Re: Windows Procedural Programming](#)**
◇ *From: Joe Butler*

- *References:*

- ◆ **[Windows Procedural Programming](#)**
◇ *From: Snooze*
- ◆ **[Re: Windows Procedural Programming](#)**
◇ *From: Joe Butler*
- ◆ **[Re: Windows Procedural Programming](#)**
◇ *From: Alf P. Steinbach*
- ◆ **[Re: Windows Procedural Programming](#)**
◇ *From: Joe Butler*
- ◆ **[Re: Windows Procedural Programming](#)**
◇ *From: Phlip*
- ◆ **[Re: Windows Procedural Programming](#)**
◇ *From: Joe Butler*

- Prev by Date: **[Re: Windows Procedural Programming](#)**
- Next by Date: **[Re: Windows Procedural Programming](#)**
- Previous by thread: **[Re: Windows Procedural Programming](#)**
- Next by thread: **[Re: Windows Procedural Programming](#)**
- Index(es):
 - ◆ **[Date](#)**
 - ◆ **[Thread](#)**