theForger's Win32 API Programming Tutorial

Getting Started

What this tutorial is all about

This tutorial is intended to present to you the basics (and common extras) of writing programs using the Win32 API. The language used is C, most C++ compilers will compile it as well. As a matter of fact, most of the information is applicable to any language that can access the API, inlcuding Java, Assembly and Visual Basic. I will not however present any code relating to these languages and you're on your own in that regard, but several people have previously used this document in said languages with quite a bit of success.

This tutorial *will not* teach you the C language, nor will it tell you how to run your particular compiler (Borland C++, Visual C++, LCC-Win32, etc...) I will however take a few moments in the appendix to provide some notes on using the compilers I have knowledge of.

If you don't know what a *macro* or a *typedef* are, or how a `switch()` statement works, then turn back now and read a good book or tutorial on the C language first.

Important notes

Sometimes throughout the text I will indicate certain things are IMPORANT to read. Because they screw up so many people, if you don't read it, you'll likely get caught too. The first one is this:

The source provided in the example ZIP file is not optional! I don't include all the code in the text itself, only that which is relevant to whatever I'm currently discussing. In order to see how this code fits in with the rest of the program, you *must* take a look at the source provided in the ZIP file.

And here's the second one:

Read the whole thing! If you have a question during one section of the tutorial just have a little patience and it might just be answered later on. If you just can't stand the thought of not knowing, at least skim or search (yes computers can do that) the rest of the document before asking the nice folks on IRC or by email.

Another thing to remember is that a question you might have about subject A might end up being answered in a discussion of B or C, or maybe L. So just look around a little.

Ok I think that's all the ranting I have to do for the moment, lets try some actual code.

The simplest Win32 program

If you are a complete beginner lets make sure you are capable of compiling a basic windows application. Slap the following code into your compiler and if all goes well you should get one of the lamest programs ever written.

Remember to compile this as C, not C++. It probably doesn't matter, but since all the code here is C only, it makes sense to start off on the right track. In most cases, all this requires if you add your code to a `.c` file instead of a `.cpp` file. If all of this hurts your head, just call the file `test.c` and be done with it.

```
#include <windows.h>

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
    LPSTR lpCmdLine, int nCmdShow)
{
    MessageBox(NULL, "Goodbye, cruel world!", "Note", MB_OK);
    return 0;
}
```

If that doesn't work, your first step is to read whatever errors you get and if you don't understand them, look them up in the help or whatever documents accompany your compiler. **Make sure you have specified a Win32 GUI (NOT "Console") project/makefile/target, whatever applies to your compiler.** Unfortunately I can't help much with this part either, as errors and how to fix them vary from compiler to compiler (and person to person).

You may get some warnings about you not using the parameters supplied to `WinMain()`. This is OK. Now that we've established you can in fact compile a program, lets go through that little bit of code....

```
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
   LPSTR lpCmdLine, int nCmdShow)
```

`WinMain()` is windows equivalent of `main()` from DOS or UNIX. This is where your program starts execution. The parameters are as follows:

`HINSTANCE hInstance`
: Handle to the programs executable module (the .exe file in memory)

`HINSTANCE hPrevInstance`
: Always `NULL` for Win32 programs.

`LPSTR lpCmdLine`
: The command line arguments as a single string. NOT including the program name.

`int nCmdShow`
: An integer value which may be passed to `ShowWindow()`. We'll get to this later.

`hInstance` is used for things like loading resources and any other task which is performed on a per-module basis. A module is either the EXE or a DLL loaded into your program. For most (if not all) of this tutorial, there will only be one module to worry about, the EXE.

`hPrevInstance` used to be the handle to the previously run instance of your program (if any) in Win16. This no longer applies. In Win32 you ignore this parameter.

Calling Conventions

`WINAPI` specifies the calling convention and is defined as `_stdcall`. If you don't know what this means, don't worry about it as it will not really affect us for the scope of this tutorial. Just remember that it's needed here.

Win32 Data Types

You will find that many of the normal keywords or types have windows specific definitions, `UINT` for `unsigned int`, `LPSTR` for `char*` etc... Which you choose is really up to you. If you are more comfortable using `char*`

instead of `LPSTR`, feel free to do so. Just make sure that you know what a type is before you substitute something else.

Just remember a few things and they will be easy to interpret. An `LP` prefix stands for *Long Pointer*. In Win32 the *Long* part is obsolete so don't worry about it. And if you don't know what a pointer is, you can either 1) Go find a book or tutorial on C, or 2) just go ahead anyway and screw up a lot. I'd really recommend #1, but most people go with #2 (I would :). But don't say I didn't warn you.

Next thing is a `C` following a `LP` indicates a `const` pointer. `LPCSTR` indicates a pointer to a const string, one that can not or will not be modified. `LPSTR` on the other hand is not `const` and may be changed.

You might also see a `T` mixed in there. Don't worry about this for now, unless you are intentionally working with *Unicode*, it means nothing.