One of the most infuriating problems i have come across in my programming career is **COMPLICATED** descriptions of how to do something in a programming language , take a look at any help file in most programming language's and try to figure out a description , sometimes i wonder if they're there to confuse.

I made the transition from Visual Basic to C/C++ because i needed to do things that were just not possible in VB or were far too slow eg :- in VB 3.0 try to find the free space on a given drive , not easy , try the same from C++ and its dead simple.

So i started to learn C , not easy but i made progress then i realised that in a lot of cases i would need to write DLL's , which i thought would be really easy to do , not so , mainly because of the lack of SIMPLE examples , (I find that i learn a language much much quicker (I currently have experience in 4 programming language's , Assembly , Visual Basic , C/C++ & Pascal/Delphi and this in itself can lead to problem's) , if i can work from a simple example then modify it to suit my own need's.

To this end i have wrote this paper to help anyone who is trying to write a 32 bit dll using the C/C++ language , i do not have the time to write a paper fully explaining all of the intricacies of 32bit dll's youll have to gleam that information from somewhere else , what i will do is make it easy for you to have a working dll in just a few minute's with minimal fuss.

Note this next section assumes you are using the Borland compiler , minimal alterations should be needed to the CPP file to have it compile successfully on other compilers.

There is one important consideration to decide upon before even writing a line of code , that decision is wether you intend to use c or cpp node (c node = C source file or cpp node = C++ source file) , i would always recommend you to use c node unless you have a specific reason for wanting to use object orientated code , using C node simplifies the way that function declarations are made and minimises the risk of mistakes , using C node is explained first then CPP node.

First off as in all C/C++ windows programming you need the following declaration

```
#include < windows.h >
```

Next forward declare any functions to be used , only 1 in this example

```
int WINAPI _export TimesTwo(int x);
```

Next the 32bit Dll should have a DllEntryPoint function this replaces the WinMain function in an EXE file or if youv'e written a 16bit dll previously LibMain.
(DllEntryPoint is called by the system when processes and threads are initialized and terminated , 16Bit dll's use LibMain and WEP.)

```
BOOL WINAPI DllEntryPoint(HINSTANCE  hinstDLL,DWORD  fdwReason,LPVOID  lpvReserved)
{
```

Now you should place a switch statement like so

```
        switch (fdwReason)
        {
        case DLL_PROCESS_ATTACH:
                {
                    // place here any code that initialises the dll as its first mapped into process's adress
                  break;
                }
           case DLL_THREAD_ATTACH:
                {
                  //place here code that initialises the dll when a new thread is created
```

```
                    break;
                    }
        case DLL_PROCESS_DETACH:
                {
                     //place here code that cleans up when the dll is removed
                    break;
                    }
          case DLL_THREAD_DETACH:
                {
                     //place here code that cleans up when a thread is finished
                    break;
                    }
}
        return TRUE;

}


//   end of  dllEntryPoint
```

For 32-bit programs, Windows calls DllEntryPoint each time the DLL is loaded and unloaded , each time a process attaches to or detaches from the DLL, or each time a thread within the process is created or destroyed.

Now place a function callable by visual basic , let's keep it simple and to the point , make the function multiply a given number by 2

```
int WINAPI _export TimesTwo(int x)
{
return x * 2;
}
```

Next decide wether you want a Module Definition file or not , if you do one like below will suffice (for 16bit programmers a DEF file is NOT required although you can have one if you want eg to place a description in the compiled file).

```
LIBRARY          PCE32
DESCRIPTION   'SAMPLE 32BIT DLL © PC-Enterprises / Paul Collishaw 1996/1997'
CODE              PRELOAD MOVABLE DISCARDABLE
DATA             PRELOAD SINGLE
EXPORTS
                 ; The names of the Dll functions , just 1 in this case
                 TimesTwo
```

That's all there is to writing a simple 32bit dll , it's really very easy , but all of the manual's or help files make it look really hard , in fact when i made the transition from 16 bit to 32 bit dll's i must admit i was baffled for quite a while , but once i'd seen a simple example and deciphered the manual's it all fell into place.

Now a word of warning , or rather two word's of warning NAME MANGLING , the cause of me banging my head against my desk as i was trying to figure out how to write a (16 bit) dll for the very first time and shouting out "why the !@!**!!!!!!!!**@   wont it work"

At the time i tried all sorts of fixes , none of which worked , until i read a simple explanation which went something like this.

If you select C++ as your target language , the compiler will indulge in a neat little trick called Name Mangling , what this mean's is that the compiler adds extra bits of information onto the name of a function to record the parameter types that have been used , now normally this doesn't matter as C++ programs that call the function know and use name mangling themselves so a problem doen't arise.

What does matter is when the function is called outside of C++ , for example VB , in this case VB will report that it can't find the specified function , all is not lost though there is a solution to the problem , turn off Name Mangling.

You turn off Name Mangling by adding extern "C" to the beginning of the function declarations.

(The forward declaration)

EG :- change `int WINAPI _export TimesTwo(int x);`

to `extern "C" int WINAPI _export TimesTwo(int x);`

(and in the function itself)

FROM

```
int WINAPI _export TimesTwo(int x)
    {
```

TO

```
extern "C" int WINAPI _export TimesTwo(int x);
    {
```

Visual Basic will now recognise the function , that's Name Mangling , seem's reall'y easy once it's explained simply but try deciphering it from a manual.

Ive included a sample program (C node) based on the above explanation called PCE32.DLL , also included is a simple Visual Basic program called Call32dll.vbp , use this to call the dll.

Try them out they both work and have comment's where needed to help you along.

HAPPY PROGRAMMING.