

Compiling Mathcad extensions with GCC

Simon Prince

August 15, 2001

1 Introduction

This document briefly describes how to compile a dynamic link library for Mathcad using the FREE compiler available from [GNU](#). The compiler is normally a unix beast (as is most of the GNU software) but has been ported to windows. Two flavours exist in windows: Cygwin and Mingw32. I've got both installed on my computer, and they seem to coexist happily, but the one that I normally use is the Mingw32 version, primarily since it can be run from a dos window.

Why use gcc? Because it's free: you don't need to buy the microsoft compiler and mess around with the development environment.

2 Installation

Installation of GCC: both cygwin and mingw are pretty straightforward to install. Either download the cygwin installation file from [Cygwin](#) or get the Mingw installation executable from [Mingw](#). Download the latest *stable* version and the installation should be quite straightforward.

I would recommend that you use the Mingw installation for a number of reasons:

1. It's a smaller installation than Cygwin
2. You'll probably be more familiar with a dos box than a unix shell
3. It's what I'm using and the following makefiles will work for it!

3 Compilation

To check that your compiler is installed, type in the following command:

```
gcc -v
```

The response should be something like:

```
Reading specs from c:/mingw/bin/./lib/gcc-lib/mingw32/2.95.3-4/specs
gcc version 2.95.3-4 (mingw special)
```

If this doesn't work then the most likely cause is that your path isn't set. Either set it temporarily in the dos box with (I installed it in c:\mingw - amend appropriately):

```
path=%path%;c:\mingw\bin\;
```

or set it permanently (under NT/2000 at least) by right clicking on 'My computer', select properties→advanced→environment variables. Select 'Path', click edit and then add the path given above.

The next thing to do is to check with a very simple 'hello world' program that the compiler's working, so cut/paste the following program into a text file called test.c. Then compile with the following command:

```
gcc -o test.exe test.c

#include <stdio.h>

void main{void}
{
printf('Hello world\n');
}
```

Execute the program and the immortal words of Kernighan and Ritchie should appear on the screen.

4 Mathcad DLL compilation

This, of course, is the bit you've been waiting for. DLL creation is a pain and it took me ages to figure out what was the correct syntax to create a DLL that mathcad would interface to. Mathsoft supply code for three compilers which confuses the issue a bit... and also supply a .lib file for compiling against with microsoft and borland C. I tried to compile against it with gcc but couldn't get it to work. Eventually, I used the Watcom code which works. So...

1. Copy the file multiply.c from the userrefi\watcom\sources\simple\ directory of your mathcad installation.
2. Copy multiply.h from directory userrefi\watcom\include\
3. Create a text file called multiply.def containing the following text:

```
EXPORTS
MultiplyRealArrayByRealScalar
```

4. Enter the following compilation commands:

```

gcc -c multiply.c
gcc -mdll -o junk.tmp -Wl,--base-file,base.tmp multiply.o
del junk.tmp
dlltool --dllname multiply.dll --base-file base.tmp --output-exp
temp.exp --def multiply.def
del base.tmp
gcc -mdll -o multiply.dll multiply.o -Wl,temp.exp
del temp.exp
copy multiply.dll "C:\Program Files\MathSoft\Mathcad 8 Professional\userefi\"

```

I'm a bit unclear as to what all this does (sort of thing you understand for about 10 seconds, then promptly forget). Line 1 compiles the source to an object, line 2 compiles it to a dll, but the dll is not yet relocatable. Line 3 (I understand that one) deletes some junk file. Lines 4 and 5 should be joined together and package the dll with the header definitions in multiply.def. Line 6 deletes some more junk. Line 7, I think, allows the dll to be relocatable. So, after running all that on your source file, a DLL should be produced which will work in mathcad. Line 9 copies this dll into the appropriate directory in mathcad. This can, of course, all be cut and pasted into a batch file to automate the process.

It's also worth noting that gcc can compile fortran code with a command like:

```
g77 -c -o abc.o abc.f
```

So, with a lot of luck, fortran subroutines could be included. I haven't done this yet with a DLL, but it works with a simple executable!

5 Thanks

Thanks to the MingW list serve group who frequently answer questions I hadn't thought to ask yet, in particular Lloyd Dupont who helped me with compiler options when I was particularly stuck.

Any errors/comments: email me at simon.m.prince@hotmail.com

6 Bibliography

[Mathsoft Software - purveyors of mathcad](#)

[Mingw homepage](#)

[Colin Peter's Mingw compiler page](#)