

Creating DLL's for Lisp-Stat on Windows

B. Narasimhan
Department of Statistics
Stanford University
Stanford, CA 94305

Revision : 1.2 of May 19, 1998

Abstract

This is a short document that I wrote for myself (and Forrest Young) with Luke Tierney's help on how to create DLL's for use with Lisp-Stat.

1 Introduction

This document is a short introduction to creating DLL's on Windows. I expect it to evolve into a document covering how to create dynamically loadable modules on all platforms for Lisp-Stat. I use Microsoft Visual C++ version 5.0 for all the examples below.

2 Example

Consider an almost identical example to the one described in Tierney's book[1].

```
1  <C subroutine 1>≡  
    void foo (int *n, double *x, double *sum) {  
        int i;  
  
        *sum = 0.0;  
        for (i = 0; i < *n; i++) {  
            *sum += x[i];  
        }  
    }
```

Defines:

foo, used in chunks 2-4.

Suppose we wish to create a DLL in Windows to load this program in `foo.c`, how can we do it?

1. Download the Lisp-Stat sources onto your machine from the University of Minnesota, Department of Statistics ftp server. You need the actual sources, not just the binaries for Windows.

The Lisp-Stat sources are supposed to contain a file called `wxls32.def` in the subdirectory `msdos` for creating DLL's but this is not the case at present. Tierney will be fixing this in the future, so for now, you need to use the file he supplied me.

Move the file `wxls32.def` supplied here to the Lisp-Stat source subdirectory `msdos`. You only need to do this once regardless of how many DLL's you create.

2. Create a subdirectory for creating windows code, say `win`. Move the file `foo.c` In this subdirectory. Edit the C program and prepend each function you want to call in Lisp-Stat with a declaration that will ensure that it is exported.

- 2 *<Export declaration 2>*≡
`__declspec(dllexport) void foo(int *n, double *x, double *sum);`
 Uses `foo 1`.

3. Copy the `makefile.vc` present in the Lisp-Stat sources in the subdirectory `Extras/sockets` to the current directory and edit it to suit your needs. For example, here is the one I use for the above example. Note how you have to quote directory names containing spaces.

```
3  <Makefile for Visual C++ 5.x >≡
    # Change the next line to point to where your xlispsstat sources are.
    XLSDIR = d:\xls-3-52.5
    WXLSDIR = $(XLSDIR)\msdos
    XLSLIB = vcwxls32.lib

    CC=cl.exe
    LINK32=link.exe
    LIB32=lib.exe

    DLL_CFLAGS=/nologo /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_Windows" /FD \
        /I $(WXLSDIR) /I $(XLSDIR) /I "c:\Program Files\DevStudio\Vc\include" \
        /D far=
    DLL_LDFLAGS=/LIBPATH:"c:\Program Files\DevStudio\Vc\lib" $(STDLIBS) \
        /nologo /subsystem:windows /dll /incremental:no /machine:I386

    STDLIBS=kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib\
        advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib\
        odbccp32.lib wsock32.lib

    .c.obj:
        $(CC) $(DLL_CFLAGS) -c $<
    {$(WXLSDIR)}.c.obj:
        $(CC) $(DLL_CFLAGS) -c $<

    OBJECTS = $(WXLSDIR)\dllstub.obj foo.obj

    foo.dll : $(OBJECTS) $(XLSLIB)
        $(LINK32) <<
        $(DLL_LDFLAGS) /out:foo.dll dllstub.obj foo.obj $(XLSLIB)
        <<

    vcwxls32.lib: $(WXLSDIR)/wxls32.def
        $(LIB32) /def:$(WXLSDIR)/wxls32.def /out:vcwxls32.lib

    clean :
        -@erase *.obj
        -@erase *.dll
        -@erase *.exp
        -@erase *.lib
        -@erase *.exe
        -@erase *.idb
```

Uses foo.l.

The makefile also assumes that the compiler tools will be found in your path.

4. Bring up an MSDOS window and in the win subdirectory make the DLL.

4a *<Command to create the DLL 4a>*≡
`nmake foo.dll`
 Uses foo 1.

That's it. Now the DLL can be used. As an example, let's try the call-cfun stuff. Note how we need the oldcfun stuff described on Tierney's project page.

4b *<Use the DLL 4b>*≡
`(require "oldcfun")`
`(def lib (shlib::shlib-open "foo.dll"))`
`(def w (shlib::old-call-cfun "foo" lib 5 (float (iseq 1 5)) 0.0))`
 Uses foo 1.

3 Concluding Remarks

One can also use a def file instead of using the export declaration, but that is clearly more inconvenient.

List of code chunks

This list is generated automatically. The numeral is that of the first definition of the chunk.

<C subroutine 1>
<Command to create the DLL 4a>
<Export declaration 2>
<Makefile for Visual C++ 5.x 3>
<Use the DLL 4b>

Index

Here is a list of the identifiers used, and where they appear. Underlined entries indicate the place of definition. This index is generated automatically.

foo: 1, 2, 3, 4a, 4b

References

- [1] Luke Tierney. *LISP-STAT: An Object-oriented Environment for Statistical Computing and Dynamic Graphics*. John Wiley & Sons (New York, Chichester), 1990.