XLISP-STAT for Microsoft Windows 3.1

Luke Tierney
School of Statistics
University of Minnesota

January 4, 1996

1 Basics

This note outlines an implementation of XLISP-STAT for Microsoft Windows 3.1. The executable is called WXLS. WXLS must be run in protected mode and requires at least a 386 processor with at least 2MB of memory. Only one instance of WXLS can be run at one time. A Win32 version is now available as well. The Win32 executable is called WXLS32. It requires Win32s to run under Windows 3.1.

This implementation uses an MDI-style interface. The listener, graphics windows, and modeless dialog windows are subwindows of an MDI client window. Menus appear in the client window's menu bar. The implementation is quite similar to the Macintosh version. This note assumes you are familiar with the description of the Macintosh version given Appendix B.2 of Tierney (1990).

The current implementation is experimental; the current release at the time of writing is *3.39*. I have only tested this version on a 486DX2 computer with 16MB of memory and a math coprocessor. Please let me know if you run into any problems. A detailed description of the problem and the hardware configuration would be most helpful.

1.1 The Listener

The listener provides parenthesis matching. Hitting the *tab* key indents code to an appropriate level. The listener follows standard Windows 3.1 conventions for handling arrow keys, cut and paste accelerators, etc.. In addition, hitting *SHIFT-ENTER* moves the cursor to the end of the current input expression.

Hitting and holding down *CONTROL-BREAK* should interrupt the current calculation. You may need to wait a bit since key events are only checked periodically during calculations.

Closing the listener is equivalent to minimizing it; you cannot remove the listener. Minimizing a graphics window is equivalent to hiding (i.e. sending it the `:hide-window` message). Closing a graph window removes it.

1.2 Editing Files

A separate program, LSPEDIT, provides a very simple Lisp file editor.[1] Like the listener, LSPEDIT supports parenthesis matching and code indentation. In addition, the **Edit**

[1] This editor is based on the code examples in Microsoft (1990).

menu in LSPEDIT contains an item for pasting a selected expression into an XLISP-STAT application. This is accomplished using the Windows Dynamic Data Exchange (DDE) mechanism. It seems to work with one instance of LSPEDIT and one of XLISP-STAT running. It should also work if there are multiple instances of LSPEDIT running. It would get very confused if there were multiple instances of XLISP-STAT, but the current version does not allow this.

1.3 Menus

As with the Macintosh version, menus can be installed in and removed from the menu bar by sending them the `:install` and `:remove` messages. The **File** and **Edit** menus that appear in the menu bar on startup are Lisp menus. They are the values of the global variables `*file-menu*` and `*edit-menu*`. The third menu, the **Windows** menu, is part of the MDI interface and is *not* a Lisp menu; you cannot change or remove it.

When a graphics window is the front window and you select the **Copy** item from the **Edit** menu, a copy of the graph is placed in the clipboard. This is done by turning on buffering, sending the window the `:redraw` message, copying the contents of the buffer to a bitmap, and placing that bitmap in the clipboard.

1.4 User Preferences

User preferences can be set in a file `wxls.ini` (or `wxls32.ini` for the Win32 version) in the windows system directory. This file can contain several variables in sections `[Xlisp]`, `[Listener]`, and `[Graphics]`:

Section	Variables
`[Xlisp]`	`Libdir`, `Workspace`
`[Listener]`	`Font`, `FontSize`
`[Graphics]`	`Font`, `FontSize`, `Color`

The `Libdir` variable should be set to the directory containing the executable and runtime files; this should be done at installation time by loading `config.lsp`. The `Workspace` variable allows an alternate initial workspace to be specified. Both can be overridden on the command line by specifying `-d` or `-w` options, respectively. The `Color` variable can be used to turn color use off if allocating a color background buffer would require too much memory. As an example, a `wxls.ini` file containing

```
[Xlisp]
Libdir=C:\WXLS
[Listener]
Font="Courier New Bold"
FontSize=13
[Graphics]
Color=off
```

specifies a library directory, a listener font, and turns color off.

1.5 Miscellaneous Notes

The feature `msdos` is included in the `*features*` list and can be used for conditional evaluation of DOS-specific code.

Since the DOS operating system uses the backslash character \ as the directory separator and the backslash character is also the Lisp escape character, a file name string that is specified within XLISP-STAT, for example as an argument to the `load` command, must be entered with two backslashes. For example, a file `foo.lsp` in the subdirectory `bar` of the current directory would be loaded using

```
(load "bar\\foo.lsp")
```

or by

```
(load "bar\\foo")
```

if the `.lsp` extension is dropped.

DOS imposes an 8-character limit on base file names. When asked to access a file with a longer base name, file system functions truncate the base name to 8 characters. Thus

```
(load "regression.lsp")
```

is equivalent to

```
(load "regressi.lsp")
```

If you have files that uses a longer name on other systems you can therefore truncate their names as long as they are unique in the first 8 characters.

There are several features that are either missing, not yet implemented, or only partially implemented:

- Clipping is not yet implemented in the graphics system. I have not yet figured out how to remove clip regions from a device context.
- A mouse is essentially required for the graphics windows and dialogs. A proper alternate keyboard interface is not yet available.
- Listener and graphics windows do not have a maximize button. The reason for this is that strange things seem to happen when the menu bar is modified while windows are maximized. You can maximize the MDI frame and then resize windows within the frame.
- WXLS has access to all the memory Windows makes available to it, but because of the segmented architecture of the 80X86 processor, no single item larger than 64K can be allocated. This limits the total number of elements in a Lisp array to 16K and the total size of a numerical array used in the linear algebra functions to about 8K elements. There is little guarantee that violations of these limits will be handled gracefully. This limitation does not exist in the Win32 version WXLS32.
- The Win16 version has a system stack of only 20K. The Win32 version is limited to a system stack of about 70K when run under Win32s, but should have a much larger stack of 250K under Windows NT.

2 More Advanced Features

2.1 Dynamic Link Libraries

Dynamic link libraries can be used from both WXLS and WXLS32, but WXLS can only use 16-bit DLL's and WXLS32 can only use 23-bit DLL's.

Dynamic Link Libraries (DLL's) can be used to provide additional cursor resources created with SDK or other toolkits, or to provide dynamically callable subroutines. The file `xlsx.c` contains startup and cleanup code for a DLL that seems to work with Borland C++. This code can be linked with cursor resources or with code into a DLL.

Once you have a DLL, the function `load-dll` can be used to load a DLL. The return value of this function is an integer representing the DLL handle. The function `free-dll` can be used to release the library.

2.2 Cursors

When the `make-cursor` function is used to load a cursor from a DLL it is given three arguments. The first is a symbol naming the cursor. The second is an integer representing a DLL handle (the DLL must have been loaded with `load-dll`). The third argument can be either an integer resource index or a string naming the cursor resource. If the DLL `stick.dll` contains a cursor resource defined by a resource file line like

```
Stick   CURSOR   STICK.CUR
```

then it can be loaded as a cursor named `stick` using the expressions

```
(setf stick-dll (load-dll "stick.dll"))
```

to load the library and

```
(make-cursor 'stick stick-dll "Stick")
```

to load the cursor.

The function `msw-cursor-size` returns a list of the width and height of a cursor on the graphics device. XLISP-STAT expands or truncates arrays supplied as arguments to `make-cursor` to this size.

2.3 Dynamic Loading

Subroutines in DLL's can also be called from within XLISP-STAT. The subroutine should be defined as for dynamic loading on the Macintosh and should use C, not Pascal, calling conventions. The header file `xlsx.h` contains the required definitions and macros. The `call-cfun` function requires two arguments to identify the routine to be loaded, a DLL handle and an ordinal index or a name string. Suppose the file `foo.c` shown on page 362 of Tierney (1990) is linked with the DLL code `xlsx.c` using the module definition file `foo.def` given by

```
LIBRARY FOO
EXETYPE WINDOWS
CODE    PRELOAD MOVEABLE DISCARDABLE
```

```
DATA    PRELOAD MOVEABLE SINGLE
EXPORTS FOO=_foo
        WEP
```

If the DLL is loaded as

```
(setf foo-dll (load-dll "foo.dll"))
```

then the routine `foo` can be called using either

```
(call-cfun foo-dll "foo" 5 (float (iseq 1 5)) 0.0)
```

or

```
(call-cfun foo-dll 1 5 (float (iseq 1 5)) 0.0)
```

The second approach works since `foo` is the first routine in the exports table.

2.4 Dynamic Data Exchange

A very minimal DDE interface is provided. The interface is very simple and based on the DDEML library. As a server, WXLS allows connections to the topic `XLISP-STAT` under the service name `XLISP-STAT`. It accepts two kinds of transactions:

- Execute transactions in which the command is a sequence of lisp expressions. The LSPEDIT application sends the current selection as an execute transaction when the **Eval Selection** menu item is chosen.
- Request transactions for the item `VALUE`. This returns a string with a printed representation of the value returned by the last expression in the most recent execute transaction of the conversation.

As a client, there are three functions you can use that correspond fairly closely to their DDEML equivalents: `dde-connect`, `dde-disconnect`, and `dde-client-transaction`.

`dde-connect` takes two arguments, strings naming a service and a topic. The topic string is optional; it defaults to the service string. The return value is a descriptor of the conversation if the connection is established, otherwise it is `nil`. At the moment conversation descriptors are integer indices into a table, and the number of concurrent conversation is limited to 30. This may change.

`dde-disconnect` takes a conversation descriptor and attempts to close the conversation. If successful it returns `t`, otherwise `nil`. If dde-disconnect is called with no arguments, then all currently active conversations are terminated. In this case the return value is `t` if any are terminated, `nil` if not.

`dde-client-transaction` requires a conversation descriptor as its first argument. The remaining arguments are keyword arguments:

`:type`: should be `:request` or `:execute`; the default is `:execute`.

`:data`: a string, currently only used by execute transactions.

`:item`: an item name string, currently only used by request transactions.

`:timeout`: a positive integer specifying the timeout in milliseconds. The default is 60000.

The return value is `t` if the transaction is successful, `nil` if not.

As an example, you could have WXLS communicate with WXLS by DDE (why you would want to I do not know, but it works):

```
> (dde-connect "xlisp-stat")
0
> (dde-client-transaction 0 :data "(+ 1 2)")
T
> (dde-client-transaction 0 :type :request :item "value")
"3"
> (dde-disconnect 0)
T
```

You can also communicate with the program manager:

```
> (dde-connect "progman")
0
> (dde-client-transaction 0 :data "[ShowGroup(Main,1)]")
T
> (dde-client-transaction 0 :type :request :item "Groups")
"Main\r
Accessories\r
Applications\r
Microsoft C/C++ 7.0\r
Software Development Kit 3.1\r
Win32 Applications\r
Adobe\r
Borland C++ 4.0 Online Books\r
Borland C++ 4.0\r
StartUp\r
Games\r
Borland C++ 3.1\r
Pocket Tools\r
XLISP-STAT Programs\r
"
> (dde-disconnect 0)
T
```

The first transaction is an execute transaction that opens the Main group's window. The second is a request that obtains a list of the current groups.

This DDE interface is experimental and may change. For the moment it seems adequate for providing configuring the integration of WXLS into Windows during setup.

2.5 Other Functions

The function `msw-free-mem` returns the total amount of free global memory that is available in Windows.

The `system` function takes a command string and an optional state argument and `WinExec`'s the command string. If the command starts a windows application then the application starts iconified if the state arguments is `nil` and normal if the argument is `t`, the default. Thus

```
(system "clock")
```

starts up a clock in normal state and

```
(system "clock" nil)
```

starts up an iconified clock. The result returned is either `t` if the `WinExec` succeeds or `nil` if it fails. On failure, anumerical error code is returned as the second value.

The function `msw-win-help` provides a minimal interface to the windows `WinHelp` function. Two arguments are required, a string naming the help file to be used, and a symbol specifying the type of help requested. The keyword symbols `:help`, `:context`, `:index`, `:key`, and `:quit` are currently supported. Context help requires an additional integer argument, and key help requires an additional key string. For example,

```
(msw-win-help "calc.hlp" :index)
```

opens help for the calculator application at the contents section. No Windows help file for XLISP-STAT is available at this time, but this function provides the necessary hook for using such a file if it becomes available.

The functions `msw-tile`, `msw-cascade`, `msw-close-all`, and `msw-arrange-icons` implement the corresponding actions on the Windows menu. These functions take no arguments.

The functions `msw-get-profile-string` and `msw-write-profile-string` can be used to access and modify user preference information. They require three and four arguments, respectively. The first and second arguments specify the section and item names as strings, and the last argument specifies the preference file name. A preference file name of `nil` refers to the system preference file. For `msw-write-profile-string` the third argument is a new value. This can be a string or `nil`; if it is `nil`, the entry is deleted. This function deletes a section if the item argument is `nil`.

2.6 Recompiling the System

If you want to recompile the system you can do so, but you have to start with a compiled version of the compiler. The system stack available under Windows is very small – only 20K compared to 180K on the Macintosh version – and is not sufficient for recompiling the compiler from `.lsp` files. You can recompile the compiler as long as you start with a compiled version.

References

Microsoft Corporation, (1990). *Microsoft Windows Guide to Programming.* Redmond, WA: Microsoft.

Tierney, L. (1990). *LISP-STAT: An Object-Oriented Environment for Statistical Computing and Dynamic Graphics.* New York, NY: Wiley.