Creating Binary Extension Modules

Description

Lua 5.0 introduces the 'loadlib' function, which allows Lua to load binary extensions at runtime (the implementation is platform specific). This document shows a simple example of an extension module, and shows how to build Lua for use with extension modules under Windows.

Example

Under Windows, 'loadlib' is implemented in terms of DLLs (dynamic link libraries). Here's complete source for an extension DLL that provides a Lua function called 'msgbox':

msgbox.c

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

/* Pop-up a Windows message box with your choice of message and caption */
int lua_msgbox(lua_State* L)
{
  const char* message = luaL_checkstring(L, 1);
  const char* caption = luaL_optstring(L, 2, "");
  int result = MessageBox(NULL, message, caption, MB_OK);
  lua_pushnumber(L, result);
  return 1;
}

int __declspec(dllexport) libinit (lua_State* L)
{
 lua_register(L, "msgbox",  lua_msgbox);
 return 0;
}
```

After this is built into a Windows DLL (msgbox.dll), you can use it in a Lua script like this (error checking omitted):

```
libinit = loadlib("msgbox.dll", "_libinit")
```

```
libinit()
msgbox("Hey, it worked!", "Lua Message Box")
```

It makes is easy to add extensions to a Lua-5.0-enabled app *at runtime* -- no need to rebuild or even restart the application -- a very hand feature. However, I've glossed over an important detail.

When building our extension, the linker must resolve the addresses of Lua/Luax API functions like 'lua_pushnumber' or 'luaL_checkstring'. If those functions exist within host application (like lua.exe), we cannot link our extension module. We must move the Lua API into a DLL, so that the host application and the extension DLL can share the same copy of the Lua API code.

[* Correction: under windows, functions can be exported from an exe as if it were a dll. Therefore you could export functions from your lua.exe (using the instructions below) and use the resulting export library to link into the new DLL *]

[* How? The instructions below don't work. Using __declspec(dllexport) in prototypes doesn't result in those functions being exported from the executable. What do I do? *]

Putting the Lua API into a DLL

These instructions are for Borland's excellent (and free) command line ANSI C|C++ compiler for Windows [download here].

From the root of the lua distribution:

```
lua-5.0.2> bcc32 -elua.dll -WD -Iinclude;C:\Program\borland\Include -LC:\Program\borland\lib -DLUA_API=__declspec(dllexport) src\
lua-5.0.2> implib lua lua.dll
```

This gives us `lua.dll`, which contains the Lua API and standard libraries; and `lua.lib`, an *import library* that we can statically link to other applications to give them access to the Lua API via `lua.dll`.

Note: `__declspec(dllexport)` is Microsoft language extensions (supported by Borland) that simplifies the creation and usage of DLLs. It eliminates the need to provide a "module definition file" that explicitly enumerates every function exported by our DLL. The `LUA_API` macro gives us the opportunity to insert `__declspec(dllexport)` into the function prototype of every Lua API function, automatically exporting them from the resulting DLL.

Building a host application

For example, to build a version of `lua.exe` (in the Lua distribution) so that it will use `lua.dll`, we would say (from the root of the distribution):

```
bcc32 -elua.exe -Iinclude -DLUA_API=__declspec(dllimport) lua.lib src\lua\*.c
```

Note we changed `LUA_API` to '__declspec(*dllimport*)', and we linked in `lua.lib`. Now `lua.exe` will dynamically link in the Lua API code from `lua.dll` at runtime.

Building an extension DLL

Assuming the `msgbox.c` (above) is the root of the lua distribution:

```
bcc32 -w -tWD -Iinclude -DLUA_API=__declspec(dllimport) lua.lib msgbox.c
```

Again, we set `LUA_API` to '__declspec(dllimport)' and link in `lua.lib`. That's it. We now have `msgbox.dll`, a binary extension ready for use with 'loadlib'.

Summary

Building binary extensions and using them with loadlib is easy, but we must make sure that the Lua API is in it's own DLL where it can be shared by the host application and any extension modules.

FindPage · RecentChanges · preferences
edit · history
Last edited April 19, 2004 10:04 am PDT (diff)