Tips for cross-compiling C on Mac for Windows

Matt Wong

July 9, 2015

Introduction

Presenter (me)

- Founder of very early-stage software company (Guarnerix)
- Former investment banker. Before that, worked in programming and similar roles

Intended audience

- Haven’t done much, or any, cross-compiling for C before
- Have basic familiarity with C and “configure + make” build process

Overview

- Why do it
- Toolchain
- Basic executables
- Make + configure: a few library examples
- Shared libraries and VBA

Why do it

Cross-compile

- Time / productivity
- Cost / complexity
- Strategic flexibility

Mac for Windows

- Many developers prefer to work on Mac
- For B2B, Windows is still customer platform of choice
- Mac / OSX and Linux generally work the same with little or no modification

Toolchain

- MinGW (64-bit)
  - Building your own is tedious. I used the latest darwin auto-build
- Environment script for Make. For example:

```
#!/bin/sh

PREFIX=x86_64-w64-mingw32
export CC=$PREFIX-gcc
export CXX=$PREFIX-g++
export CPP=$PREFIX-cpp
export RANLIB=$PREFIX-ranlib
export PATH="/path/to/mingw64_autobuild/bin:/opt/mingw64/bin:$PATH"
export AR=$PREFIX-ar
export C_INCLUDE_PATH=/path/to/mingw64_autobuild/include:/opt/mingw64/include
export CPLUS_INCLUDE_PATH=$C_INCLUDE_PATH
export LIBRARY_PATH=/path/to/mingw64_autobuild/lib:/opt/mingw64/lib64:/opt/mingw64/lib
export LD_LIBRARY_PATH=/path/to/mingw64_autobuild/lib:/opt/mingw64/lib64:/opt/mingw64/lib

exec "$@"
```

Basic executable: hello world

C

```
#include <stdio.h>

int main() {
  printf("Hello World");
}
```

Makefile

```
ifneq ($(findstring w64,$(CC)),) # win64
        EXE=64.exe
endif

all: hello_world${EXE}

hello_world${EXE}: hello_world.c
        ${CC} hello_world.c -o $@

clean:
        rm hello_world${EXE}
```

OSX build (hello_world)

```
make
```

Win build (hello_world64.exe)

```
~/mingw64_env make
```

Other Makefile setting examples

```
ifneq ($(findstring w64,$(CC)),) # win64
        EXE=64.exe
        DOT_O=.w64o
        WINLIB=-lwsock32 -loleaut32
        WINDEFS=-D__USE_MINGW_ANSI_STDIO -D_ISOC99_SOURCE
else
        DOT_O=.o
endif


xxx${DOT_O}: xxx.[ch]
        ${CC} ${CFLAGS} ${WINDEFS} xxx.c -o $@


yyy${EXE}: ...
        ${CC} ... ${WINLIB}
```

Make + configure: a few library examples

- libiconv

```
~/mingw64_env ./configure --host=x86_64-w64-mingw32 --prefix=/opt/mingw64
```

- openssl

```
~/mingw64_env ./Configure --prefix=/opt/mingw64 no-idea no-mdc2 no-rc5 no-
shared mingw64

~/mingw64_env make depend

~/mingw64_env make
```

- libtool

```
~/mingw64_env ./configure --host=x86_64-w64-mingw32 --prefix=/opt/mingw64

~/mingw64_env make install
```

Shared libraries and VBA: introduction

- Why shared libraries?
  - Use your C code via VBA to extend closed-source third-party apps
- Why VBA?
  - Can use to extend MSOffice (Excel, Word, Powerpoint etc)
- Why MSOffice?
  - Extending a widely-used application suite is often a better solution from customer's perspective than a separate, standalone application

Shared libraries and VBA: challenges

- Returning integer-type values is easy
- Returning strings and arrays takes more work, especially if your binary must run on both Windows and Mac
  - If it only needs to run on Windows, use SysAlloc and SafeArray functions
  - Cross-platform compatibility becomes more complicated because SysAlloc / SafeArray functions are not available

Shared libraries and VBA: cross-platform string workaround

- Problem: no SysAlloc family of functions. Rolling our own is complicated
- Workaround: don't use SysAlloc
  - Call DLL function to do something and return resulting string size
  - Allocate memory
  - Call DLL function to copy result into allocated memory
- Could also use similar workaround for arrays

Shared libs and VBA: cross-platform string workaround (C)

```
char *saved_string = NULL;
int32_t saved_len = -1;

int32_t __declspec(dllexport) WINAPI
myfunc(...) {
  ...
  saved_string = malloc(result_length + 5);
  ...
  saved_len = len + 5; /* added 5 bytes just for padding */
  return saved_len;
}

int32_t __declspec(dllexport) WINAPI
get_saved_string(LPSTR pszString, int cSize) {
  int32_t old_saved_len = saved_len;
  if(saved_len > 0 && cSize >= saved_len)
    memcpy(pszString, saved_string, saved_len);
  if(saved_string) {
    free(saved_string);
    saved_string = NULL;
    saved_len = -1;
  }
  return old_saved_len;
}
```

Shared libs and VBA: cross-platform string workaround (VBA)

```
Public Declare Function _
  myfunc Lib "path:to:test.dylib" (...) As Long

Public Declare Function _
  get_saved_string Lib "path:to:test.dylib" _
    (ByVal s As String, ByVal csize As Long) As Long


Public Function getDLLString(string_size As Long) As String
    Dim s As String
    If string_size > 0 Then
        s = Space$(string_size)
        get_saved_string s, string_size
    End If
    getDLLString = s
End Function


Public Sub test()
    Debug.Print getDLLString(myfunc(...))
End Sub
```

C cross-platform minutiae: beware of sizes (example: printf)

- OSX without __USE_MINGW_ANSI_STDIO or _ISOC99_SOURCE:

Variable declaration	sizeof	Format spec	Result
long double	16	%Lf	0.000000
double	8	%Lf	0.000000
float	4	%f	12345.669922
float	4	%Lf	0.000000

(use “%f” instead, and don’t use “float”)

- OSX with __USE_MINGW_ANSI_STDIO and _ISOC99_SOURCE

Variable declaration	sizeof	Format spec	Result
float	4	%f	12345.669922
float	4	%Lf	12345.669922

(use “double” instead)

C cross-platform minutiae: examples

- File paths and separators, realpath() vs _fullpath()
- File writing, reading, flushing, concurrent access
- Size: “long” vs “long long”, printf family, etc
- mkstemp(): google “mkstemp source”
- mktime() family
- asprintf(), vasprintf(): roll your own asprintf using vasprintf (google “vasprintf source” and choose a suitable version+license)
- Atomic functions e.g. sync_add_and_fetch / OSAtomicAdd64Barrier: inconsistent signatures, return values

VBA cross-platform minutiae: examples

```
#If Mac Then
...
#Else
...
#End If
```

- Kernel functions such as CopyMemory: libc.dylib in lieu of kernel32.dll
- File paths: Mac (colon) vs Unix (slash) vs Windows (backslash)
- File handling: existence, creating, deleting
- HTTP: WinHttp vs curl + “do shell script” + VBA.MacScript()

Resources

- Mingw64 auto-builds:
  http://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win64/Automated%20Builds/

- DLL/VBA string workaround sample code:
  http://c-programming-blog.blogspot.com/2014/09/dlls-written-in-c-eg-to-be-called-from.html

- Examples of vasprintf and mkstemp source:
  http://www.ncbi.nlm.nih.gov/IEB/ToolBox/CPP_DOC/doxyhtml/vasprintf_8c_source.html
  http://www.opensource.apple.com/source/lukemftp/lukemftp-13/tnftp/libnetbsd/mkstemp.c

Questions / comments

matt@guarnerix.com