Guide To Building ImageMagick For MacOSX

Written by Kevin Gale
Email: keving@ncs-plc.co.uk
Version: Monday, 23rd September 2002

Thanks to:
The ImageMagick developers for creating a great product.
Ilya Goldberg, Bob Friesenhahn, and Glenn Randers-Pehrson for helping me to get this far.

Introduction

The purpose of this document is to help MacOSX users (and possibly other Unix users) trying to build ImageMagick and the delegates from the source code.

Why not use the pre-built binaries/packages I here you say?

Well, a project I am involved in requires that all of the ImageMagick files (delegates / command line tools etc..) are distributed in my application's directory so that users can install the whole system by simply dragging it from a server / CD. The pre-built binaries for MacOSX require that files are installed into various paths which are normally protected meaning that the user has to use their root password to install the ImageMagick libraries. This document is my ongoing struggle to try to duplicate the structure of the pre-built ImageMagick binaries for Windows as they contain all of the files within the same directory.

Before building ImageMagick the first thing to decide is whether to build Static or Shared libraries.

When building ImageMagick using Static libraries the actual code from the libraries is actually added to various executables (convert, mogrify etc..). The main advantage when using Static libraries is the fact that there are less files to keep track of as all of the required code will be within the executable. However, it does mean that the command line tools become very large in size quickly and require more resources to run. If you are building ImageMagick with support for one or two image formats than this will probably be okay but if you are planning to build ImageMagick with all of its delegates then Static libraries will probably not be the way to go.

When building ImageMagick using Shared libraries the actual code from the libraries stay in their separate files meaning that the various executables are quite small. As the command line tools share the same libraries it makes sense to build ImageMagick this way.

Unfortunately, at this point in time I have not been successful in building some of the delegates as shared libraries (JPEG for starters) meaning that this document currently only covers building ImageMagick using Static libraries. I will, of course (hopefully with the help of others), update this document when I manage to solve the various problems I have encountered.

Note to PerlMagick users
The version of Perl supplied with OSX has been built using shared libraries.
This means that the build instructions within this document will not allow you to use PerlMagick unless you build a static version of Perl yourself.

Static Libraries

This following shows how to build ImageMagick and some of its delegates using static executables under OSX.
The example requires that the various source files are located in specific paths. You can, however, use any path as long as you substitute the paths at the required point.

1) The ImageMagick source files need to be at *$home/ImageMagick-5.4.9*. The source code can be downloaded from the ImageMagick FTP site into *$home* and de-compressed by dropping the archive onto Stuffit Expander. ImageMagick 5.4.9-1 was used when creating this document.

2) The ImageMagick delegate source files need to be at *$home/IMDelegates*. The delegates can be downloaded from the ImageMagick FTP site into this path and de-compressed by dropping them onto Stuffit Expander.

3) ImageMagick will be built at *$home/im*.

Building The Output Structure

Before we start building the delegates we need to create various directories to hold the ImageMagick and delegate binary files.

Launch Terminal and enter each one of these commands followed by a carriage return.

```
cd $home

mkdir $home/im

cd $home/im

mkdir tmp

mkdir tmp/lib

mkdir tmp/include

mkdir tmp/bin

mkdir tmp/man

mkdir tmp/man/man1
```

Building The Delegates

The first delegates we build are actually libraries which will be used by other delegates. Some delegates have test programs which can be ran after the build process has completed. It is not required that any test programs are executed to build ImageMagick.

Execute the commands listed below within Terminal to build each delegate.

BZIP2 (libbz2)

```
cd $home/IMDelegates/bzip2-1.0.2

make install PREFIX=$home/im/tmp

make test                                (if you wish to test the library)

ranlib $home/im/tmp/lib/libbz2.a
```

ZLIB (libz)

```
cd $home/IMDelegates/zlib-1.1.4

./configure --prefix=$home/im/tmp
```

If you wish to test ZLIB after compilation you will have to edit *$home/IMDelegates/zlib-1.1.4/makefile* and change line

```
LDFLAGS=-L. -lz
```

to

```
LDFLAGS=-L. libz.a
```

(see *http://www.gzip.org/zlib/zlib_faq.html* for more information on this)

```
make

make install

make test                                (if you wish to test the library)
```

JPEG (libjpeg)

```
cd $home/IMDelegates/jpeg-6b

./configure --prefix=$home/im/tmp
```

If the *./configure* script reports an error stating that it does not recognise the target system you will have to execute the following commands and then re-execute the *./configure* command.

```
             cp /usr/libexec/config.sub .

             cp /usr/libexec/config.guess .

make

make install

make install-lib

make test
```

(if you wish to test the library)

```
ranlib $home/im/tmp/lib/libjpeg.a
```

JBIG (libjbig)

```
cd $home/IMDelegates/jbigkit

make

make test
```

(if you wish to test the library)

```
cp libjbig/libjbig.a $home/im/tmp/lib

cp libjbig/jbig.h $home/im/tmp/include

ranlib $home/im/tmp/lib/libjbig.a
```

JASPER (libjasper)

```
cd $home/IMDelegates/jasper-1.500.4

./configure --enable-static --disable-shared --prefix=$home/im/tmp

make

make install
```

<u>**TIFF**</u> (libtiff)

```
cd $home/IMDelegates/tiff-v3.5.7
```

```
./configure --prefix=$home/im/tmp
```

You will be asked if the installation paths are correct. If you do not need the manual pages then you can just type *y* and ignore the *Permission Denied* error messages which are displayed when you execute *make install*. If you do want the manual pages then you are supposed to press *4* and then enter the full path where you wish to install the files. Unfortunately, this does not appear to work as they are still installed into */usr/local/man*.

```
make
```

`make install` (you may see Permission denied errors if you have not modified the manual pages path)

```
ranlib $home/im/tmp/lib/libtiff.a
```

<u>**PNG**</u> (libpng)

```
cd $home/IMDelegates/libpng-1.2.4
```

```
cp scripts/makefile.macosx makefile
```

edit *$home/IMDelegates/libpng-1.2.4/makefile* and change the following lines:

`prefix=/usr/local`
to
`prefix=`

`ZLIBLIB=../zlib`
to
`ZLIBLIB=`

`ZLIBINC=../zlib`
to
`ZLIBINC=`

```
make install DESTDIR=$home/im/tmp
```

Building ImageMagick

Once the delegates have been built it is simply a matter of telling the ImageMagick *./configure* script the location where we wish to build ImageMagick along with the location of the delegate's headers and libraries.

Execute the commands listed below within Terminal to build ImageMagick.
NOTE The *.configure* command below is one command even though it is split over two lines.

```
cd $home/ImageMagick-5.4.9
```

```
./configure --prefix=$home/im --enable-delegate-build
LDFLAGS='-L/pathto/tmp/lib' CPPFLAGS='-I/pathto/tmp/include'
```

e.g: if your *$home* path is */users/keving* then the following example will build ImageMagick:

```
./configure --prefix=$home/im --enable-delegate-build
LDFLAGS='-L/users/keving/im/tmp/lib' CPPFLAGS='-I/users/keving/im/tmp/include'
```

The output of the *./configure* script will list all of the delegates which have been identified. If you have followed all of the above instructions correctly then the configuration should be similar to the following:

```
Host system type : powerpc-apple-darwin6.0

Option           Configure option              Configured value
-----------------------------------------------------------------
Shared libraries  --enable-shared=no           no
Static libraries  --enable-static=yes          yes
GNU ld            --with-gnu-ld=no             no
LZW support       --enable-lzw=no              no
Quantum depth     --with-quantum-depth=8       16

Delegate Configuration:
BZLIB             --with-bzlib=yes             yes
DPS               --with-dps=yes               no
FlashPIX          --with-fpx=yes               no
FreeType 2.0      --with-ttf=yes               no
Ghostscript       None                         gs
Ghostscript fonts --with-gs-font-dir=default   none
Ghostscript lib   --with-gslib=yes             no
HDF               --with-hdf=no                no
JBIG              --with-jbig=yes              yes
JPEG v1           --with-jpeg=yes              yes
JPEG v2           --with-jp2=yes               yes
LCMS              --with-lcms=yes              no
MPEG v2           --with-mpeg2=yes             no
Magick++          --with-magick-plus-plus=no   yes
PERL              --with-perl=yes              /usr/bin/perl
PNG               --with-png=yes               yes
TIFF              --with-tiff=yes              yes
Windows fonts     --with-windows-font-dir=     none
WMF               --with-wmf=yes               no
X11               --with-x=                    no
XML               --with-xml=yes               no
ZLIB              --with-zlib=yes              yes

X11 Configuration:

Not using X11.

Options used to compile and link:
CC       = gcc
CFLAGS   = -g -O2 -Wall
CPPFLAGS = -D_REENTRANT -I/users/keving/im/tmp/include
CXX      = g++
CXXFLAGS = -g -O2
LDFLAGS  = -L/users/keving/im/tmp/lib
LIBS     = -ljbig -ltiff -ljasper -ljpeg -lpng -lbz2 -lz -lpthread -lm
```

```
make

make install
```

Tidying Up

Once installed we can move the files to more convenient locations and remove the files / directories which are not required anymore.

Execute the commands listed below within Terminal to tidy up the build directory.

`cd $home/im/`	
`cp $home/ImageMagick-5.4.9/*.txt .`	(copy the text files to the parent directory)
`cp -r $home/ImageMagick-5.4.9/www .`	(copy the documentation to the parent directory)
`mv bin/* .`	(move the command line tools to the parent directory)
`rm -d -r bin`	(remove the *bin* directory as it is not required anymore)
`mv lib/ImageMagick/*.mgk .`	(move the *.mgk* config files to the parent directory)
`rm -d -r share`	(remove the *share* directory as it is not required anymore)

If the tools which come supplied with the delegates (not the ImageMagick tools) are required execute the following command to move the files into a more appropriate location.

`mkdir delegatetools`	
`cp -r tmp/bin/* delegatetools/`	(copy the contents of *tmp/bin* into *delegatetools*)

If the c/c++ api is required execute the following commands to move the files into more appropriate locations.

`cp -r tmp/include/* include/`	(copy the contents of *tmp/include* into *include*)
`cp -r tmp/lib/* lib/`	(copy the contents of *tmp/lib* into *lib*)
`cp -r tmp/man/* lib/`	(copy the contents of *tmp/man* into *lib*)
`rm -d -r -f tmp`	(remove the *tmp* directory as it is not required anymore)

If the c/c++ api is not required the headers and libraries can be removed by executing the following commands:

`rm -d -r -f tmp`	(remove the *tmp* directory as it is not required anymore)
`rm -d -r include`	(remove the *include* directory as it is not required anymore)
`rm -d -r lib`	(remove the *lib* directory as it is not required anymore)

If the manual pages are not required execute the following command to remove the directory:

`rm -d -r -f man`	(remove the *man* directory as it is not required anymore)