Building a gretl disk image for OS X

Allin Cottrell, December 2007

1 Objective

To build a stand-alone disk image (dmg) of gretl, including a suitably configured version of gnuplot, for Mac OS X. The final user should be able to download the dmg file, double-click to mount it, and drag the `Gretl.app` folder (found "inside" the image) to an Applications folder. You'll use Fink in the build process but the final dmg should not be dependent on Fink in any way; it will, however, be dependent on Apple's X11.

2 Overview

Here are the basic prerequisites:

- A fully functional installation of OS X.
- Apple's X11 and the Xcode development package. If these are not already installed, they should be found on the OS X installation DVDs.
- A basic installation of Fink.
- Source code for gretl and gnuplot.
- A skeleton for `Gretl.app` plus some auxiliary scripts.

The method is as follows:

1. Install the `Gretl.app` skeleton. This provides the "space" into which you'll install gretl and gnuplot.
2. Under Fink, install various required third-party packages (including the "dev" or developer components). This includes GTK+ and friends (glib, atk, gdk, pango).
3. Configure and build gnuplot; install gnuplot into the `Gretl.app` folder.
4. Configure and build gretl; install gretl into the right place inside `Gretl.app`; delete some extraneous files and add some extras.
5. Tar up various run-time files from your Fink installation and dump them into the appropriate place in `Gretl.app` (hence removing the dependency on Fink at run time). This is the trickiest part.
6. Grab the latest gretl documentation and dump it into place.
7. Create a compressed disk image containing `Gretl.app`.

Steps 1, 2, 3 and 5 only need to be done once; thereafter you can update the disk image with just steps 4, 6 and 7.

The following sections expand on each of the steps.

3 The Gretl.app skeleton

I'm making a gzipped tar file available. This is mostly an empty directory tree, but it includes some "generic" files that shouldn't depend on the particular OS X build platform (though see the final section below). This should be unzipped in some suitable location; on the OS X system to which I have access I've put it under `/Users/allin/dist`.

`http://ricardo.ecn.wfu.edu/~cottrell/gretl-osx/Gretl.app.tar.gz`

4 Required Fink packages

The exact line-up of these packages depends somewhat on the specific OS X variant. If a given package is available via OS X itself, then you don't need to, and probably don't want to, install the corresponding Fink package. A case in point is libxml2, which is supplied on recent OS X (but was not supplied in earlier variants).

The required packages will presumably include gtk+2, gtk+2-dev and gmp3 and fftw3; recode may also be required; gnuplot is not required since we'll be building that ourselves. Libxml2 will hopefully be supplied by OS X, and dlcompat doesn't seem to be needed any longer. It may be helpful to install wget via Fink for build purposes. I have also installed mpfr via Fink: this is not strictly required for gretl, but if it's present it enhances the functionality of the multiple precision plugin.

Since we're building gnuplot, the libraries to be installed via Fink also include those needed by gnuplot (see the following section).

5 Building gnuplot

First, why do we bother building gnuplot? Well, briefly, gnuplot is rather integral to gretl, and it's preferable to have a gnuplot version that is as functional as possible. We want good PNG support, provided via libgd. Unfortunately the standard libgd installation on Fink has a problem with finding TrueType fonts; we can work around this if we build libgd ourselves. In addition, OS X is highly PDF-oriented, so it's nice to be able to provide PDF graph support, which I don't think is in a stock Fink gnuplot build. Finally, the gnuplot source referenced below is patched so as to provide accurate bounding box information for PNG files; this improves the quality of mouse-over interaction with graphs under gretl.

Fink stuff: To support the gnuplot build, install libpng3, libpng-shlibs, pdflib, and pdflib-shlibs. Although we'll build libgd separately, it may be handy to install gd2 and gd2-shlibs via Fink, for build-time convenience.

libgd: Grab patched source from

`http://ricardo.ecn.wfu.edu/~cottrell/gretl-osx/libgd-2.0.35.tar.gz`

Untar and configure with

```
export CFLAGS="-O2 -I/sw/include"
export CPPFLAGS=$CFLAGS
export LDFLAGS="-L/usr/X11R6/lib"
./configure --prefix=/Users/allin/misc --disable-rpath \
--disable-static --without-jpeg
```

Note that I'm installing to a temporary location outside of Gretl.app. When we're ready, we'll just grab the appropriate dylib file from the lib directory in that temporary tree (here `/Users/allin/misc`).

Then `make` and `make install`.

Grab the patched source for gnuplot 4.2.2, `http://ricardo.ecn.wfu.edu/~cottrell/gretl-osx/gnuplot-4.2.2-ac.tar.gz`. Untar and configure with

```
./configure --prefix=/Users/allin/dist/Gretl_Folder && \
echo "#define PNG_COMMENTS 1" >> config.h
```

Again, `make` and `make install`.

6 Configuring and building gretl

There's a file `myconf` in the osx subdirectory of the gretl source. You should use this, or a variant of it, to configure gretl. Here's what it looks like:

```
export CFLAGS="-O2 -I/sw/include"
export LDFLAGS=-L/sw/lib
export CPPFLAGS=$CFLAGS
export PKG_CONFIG_PATH="/usr/lib/pkgconfig:/usr/X11R6/lib/pkgconfig:/sw/lib/pkgconfig"
export PATH=/Users/allin/dist/Gretl.app/Contents/Resources/bin:$PATH
./configure --prefix=/Users/allin/dist/Gretl.app/Contents/Resources \
  --disable-rpath --enable-build-doc
```

The "`export PATH`" line is designed to ensure that the version of gnuplot installed at the previous step is found during the gretl configuration process. This may not be necessary if a version of gnuplot that supports PNG output is already in your path.

After doing `make` and `make install` we run a script named postinst to clear out unnecessary files and add a few extra things needed for OS X. This is also in the `osx` subdir of the source.

```
#!/bin/sh
# postinst: run this in the gretl build directory

# The directory above Gretl.app
TOPDIR=/Users/allin/dist
PREFIX=$TOPDIR/Gretl.app/Contents/Resources

rm -f $PREFIX/bin/gretl
rm -rf $PREFIX/include
rm -rf $PREFIX/share/aclocal
rm -rf $PREFIX/share/info
rm -rf $PREFIX/info
rm -rf $PREFIX/lib/pkgconfig
rm -f $PREFIX/lib/*.la
rm -f $PREFIX/lib/gretl-gtk2/*.la
rm -rf $PREFIX/share/emacs

install -m 644 osx/README.pdf $TOPDIR
install -m 755 osx/gretl.sh $PREFIX/bin/gretl
```

7 Copying Fink run-time files

As mentioned above, this is a bit tricky. The general idea is that we want to identify all the files provided by Fink that are necessary to support gretl and/or gnuplot, and copy these into the right places under `Gretl.app`. To keep the disk image as compact as possible, we want to try to ensure

that we copy *only* those files that are really necessary. This includes all shared libraries that are not provided by OS X itself; it also includes some additional run-time files required by GTK+.

Libraries: You can do most of the work on the libraries by running `libs.sh` from the osx subdirectory of the gretl source. This finds dependencies that live in `/sw/lib` using `otool -L` (think ldd on Linux) and copies the files into place. Here's `libs.sh`:

```
TOPDIR="/Users/allin/dist"
PKGDIR="$TOPDIR/Gretl.app/Contents/Resources"

cd $PKGDIR

otool -L ./bin/gretl_x11 | awk '{ print $1 }' | grep /sw/lib > liblist
otool -L ./bin/gnuplot | awk '{ print $1 }' | grep /sw/lib >> liblist

for f in `cat liblist` ; do
  cp $f ./lib
done
```

But having done this, we need to add a few things: `libpangoft2-1.0.0.dylib` (not picked up, maybe an indirect dependency); `libgd.2.dylib` from the custom build of gd2 (see section 5), `libmpfr.1.dylib` from Fink to support the gretl plugin. On the other hand, if `libiconv.2.dylib` has been picked up from the Fink installation it can be deleted since it's provided by OS X.

Other files:

Besides the basic dylibs, we need to borrow from Fink some module files that live under `lib`, and configuration files under `etc` (the installation for both sets of files will be relative to `Gretl.app/Contents/Resources`). I'll illustrate with my current setup; filenames may differ slightly if the GTK version differs.

Under `lib`:

```
lib/gtk-2.0/2.4.0/immodules/
lib/gtk-2.0/2.4.0/loaders/libpixbufloader-png.so
lib/gtk-2.0/2.4.0/loaders/libpixbufloader-xpm.so
lib/pango/1.4.0/modules/pango-basic-fc.so
lib/pango/1.4.0/modules/pango-basic-x.so
```

The `immodules` directory is empty; we don't need any input modules. The directory may be redundant, I'm not sure. The GTK `loaders` directory and the pango `modules` directory contain a small subset of the full content of those directories under Fink; this is all we need to support gretl.

Second, config files under `etc`:

```
etc/gtk-2.0/gtk.immodules
etc/gtk-2.0/gdk-pixbuf.loaders
etc/pangorc
etc/pango/pango.modules
etc/pango/pangox.aliases
```

The first of these is an empty file; the second is an edited version of the corresponding Fink file. We need to use relative paths to the loaders, as in

```
# GdkPixbuf Image Loader Modules file
"../lib/gtk-2.0/2.4.0/loaders/libpixbufloader-png.so"
"png" 1 "gtk20" "The PNG image format"
"image/png" ""
```

```
"png" ""
"\211PNG\r\n\032\n" "" 100

"../lib/gtk-2.0/2.4.0/loaders/libpixbufloader-xpm.so"
"xpm" 0 "gtk20" "The XPM image format"
"image/x-xpixmap" ""
"xpm" ""
"/* XPM */" "" 100
```

(Only the paths to the `.so` files need to be changed.)

The third file, `etc/pangorc`, tells pango where to find the other files:

```
[Pango]
ModuleFiles=../etc/pango/pango.modules
ModulesPath=../lib/pango/1.4.0/modules
[PangoX]
Aliasfiles=../etc/pango/pangox.aliases
```

The content of `pango.modules` is again edited to use relative paths:

```
# Pango Modules file
#
../lib/pango/1.4.0/modules/pango-basic-fc.so Basic...
../lib/pango/1.4.0/modules/pango-basic-x.so Basic...
```

As for `pangox.aliases`, I'm not sure it's needed, but anyway I think it can be copied straight.

Testing: Once you've got all this stuff in place (remember, you only need to do it once!) you have to check that that `Gretl.app` is really self-contained. This means running gretl with Fink disabled. To help with this I use two little scripts, as follows:

```
# disable Fink
sudo mv /sw /hidden.sw
hash -r

# enable Fink
sudo mv /hidden.sw /sw
hash -r
```

The `hash -r` command is required to ensure that common utilities such as `cp`, which are present in both `/bin` and `/sw/bin`, are found after the switch.

I recommend starting the test by running `./gretl` in the `bin` directory under `Gretl.app/Contents/Resources`: then if there's stuff missing you'll hear about it on stderr. Once that's working, try launching gretl via the `Gretl.app` icon.

8 Documentation files

The canonical PDF documentation for gretl is available from `ricardo.ecn.wfu.edu`. You should do something like the following (wget can be installed via Fink; you could use curl instead if you prefer):

```
TOPDIR=/Users/allin/dist
DOCDIR=$TOPDIR/Gretl.app/Contents/Resources/share/gretl/doc
```

```
rm -f gretl-guide.pdf
wget http://ricardo.ecn.wfu.edu/pub/gretl/manual/PDF/gretl-guide.pdf
cp gretl-guide.pdf $DOCDIR
rm -f gretl-ref.pdf
wget http://ricardo.ecn.wfu.edu/pub/gretl/manual/PDF/gretl-ref.pdf
cp gretl-ref.pdf $DOCDIR
```

That is, you grab the current PDF files and drop them into the right place under `Gretl.app`.

9 Creation of dmg

Once everything's in place, we create the final compressed .dmg file. This is actually quite straightforward; below is a shell script to do the job. There's a copy in the gretl source package, in the osx subdirectory (called `dmg.sh`). Obviously, you'll need to edit the line that defines `TOPDIR`; hopefully the rest should be portable.

This script should be run from some "neutral" location outside of the distribution tree; you don't want to get a recursive thing going, whereby the dmg is included within itself. I run `dmg.sh` from `~/bin`.

```
#!/bin/bash

# the directory above Gretl.app
TOPDIR=/Users/allin/dist

HERE=‘pwd‘
KB=‘du -ks $TOPDIR | awk ’{ print $1 }’‘
KB=$((KB+1024))
hdiutil create -size ${KB}k tmp.dmg -layout NONE
MYDEV=‘hdid -nomount tmp.dmg‘
sudo newfs_hfs -v gretl $MYDEV
hdiutil eject $MYDEV
hdid tmp.dmg
cd $TOPDIR && \
/sw/bin/cp -a Gretl.app /Volumes/gretl && \
/sw/bin/cp -a README.pdf /Volumes/gretl
cd $HERE
hdiutil eject $MYDEV
hdiutil convert -format UDZO tmp.dmg -o gretl.dmg && rm tmp.dmg
```

This script uses the `-a` flag to the `cp` command: that is not supported by `/bin/cp` under OS X, but it is supported by Fink's `/sw/bin/cp`.

10 Extra: ScriptExec stuff

Something else should be mentioned. To make Gretl.app into a proper OS X Application "bundle" we use the ScriptExec apparatus from gimp.app, at `http://gimp-app.sourceforge.net/`. This apparatus allows for launching gretl from an icon, and automatic startup of X11.

The `Gretl.app` skeleton mentioned above contains all the files generated in association with ScriptExec as built on OS X 10.4.11. So with any luck you should not have to mess with this. On the other hand it's possible that the files generated on Tiger don't work properly with Leopard (or higher), so it may be worth regenerating them.

An account of how to do this (for gimp, but *mutatis mutandis* for gretl) is given at the gimp.app URL above. I can't add much to what's said there as I have no expertise in Xcode and I worked by trial and error, but here are a few hints.

The basic idea is that you have to build ScriptExec as an Xcode project, then copy the generated bits and pieces into place under `Gretl.app`.

The built executable,

```
ScriptExec/build/Default/ScriptExec.app/Contents/MacOS/ScriptExec
```

should be copied as

```
Gretl.app/Contents/MacOS/Gretl
```

A copy of my build of this program can be found in the osx subdir of the gretl code, named `Gretl.intel`.

A copy of my gretlized version of the ScriptExec source is at

`http://ricardo.ecn.wfu.edu/~cottrell/gretl-osx/ScriptExec.tar.gz`