TUG-GHOSTSCRIPT

RICHARD KOCH

1. Ghostscript

This is the portion of the TUG documentation which deals with making the Ghostscript package. A clean package file will contain

- "About" folder with public documentation explaining how the package was constructed
- "Binaries" folder with empty subfolders "10.3–10.4", "10.5", "10.6" and the file "liposcript"
- Files "buildPackage.sh" and "buildPackage1.sh"
- Files "Description.plist" and "Info.plist"
- "resources" folder containing various files
- "source" folder containing "ghostscript-fonts-std-8.11.tar" and a folder of fonts named "fonts". The tar folder contains a packed version of these fonts. (This folder is not needed in the current package because Ghostscript now contains the 35 standard fonts. It will eventually be removed.)
- "TUG" folder with this document

To construct the package, download the latest Ghostscript tar file from `http://www.ghostscript.com/` and put the file in the "source" folder. Then unzip this file; the resulting folder is the source code for Ghostscript.

2. OverView

This folder contains an "About" folder giving an overview of the construction of the Ghostscript package. You may want to read that document, because otherwise the compilation steps below will seem unnecessarily baroque.

3. Leopard binaries

Build Ghostscript on Leopard Intel and Leopard PPC. On each of these operating systems, built a Ghostscript binary with X11 support, and a binary without X11 support. Before building, make sure that Apple's X11 package is installed on both systems.

We'll use Intel as an example. Move a copy of the Ghostscript source code to the Intel Leopard system. Using Terminal, switch directories to /usr and rename local to local-temp. Then in the Ghostscript source code type

```
make clean
./configure
make
sudo make install
```

Use the Finder's "Go" menu to go to /usr/local/bin and copy gs to the desktop. Rename it gs-X11-Intel. Next go to /usr and

```
sudo rm -R local
sudo mv X11R6 X11R6-temp
```

If there is a folder named X11, rename it as well. Then rebuild Ghostscript as above, move gs in /usr/local/bin to the desktop, and name it gs-noX11-Intel.

Finally, clean up /usr by moving /usr/local-temp back to /usr/local, moving X11R6-temp to X11R6, and similarly for X11 if it existed.

Repeat this entire process on PowerPC Leopard, calling the resulting binaries gs-X11-PPC and gs-noX11-PPC.

Finally, move the four resulting binaries to the Binaries/10.5 directory of this MacTeX Ghostscript build tree.

4. Tiger binaries

Repeat the entire process on Intel Tiger, and on PowerPC Panther. On both of these systems, Ghostscript must be compiled without cups support. So in the build step, write

```
make clean
./configure --disable-cups
make
sudo make install
```

This will give four binaries. Name them gs-X11-tigerIntel, gs-noX11-tigerIntel, gs-X11-pantherPPC, and gs-noX11-pantherPPC and put them in the Binaries/10.3–10.4 directory of this MacTeX Ghostscript build tree.

5. Snow Leopard binary

Build one binary on Snow Leopard (Intel only, of course). Since X11 is installed by default on this system, it is only necessary to build a version of Ghostscript supporting X11. Repeat all of the steps in the Leopard build above. Call the resulting binary gs-X11-64 and put it in the Binaries/10.6 directory of this MacTeX Ghostscript build tree.

6. Putting It All Together

On whatever platform you use to build install packages, switch to the MacTeX build tree's Ghostscript folder; change to the Binaries directory, and type

```
sh liposcript
```

This will lipo the various binaries together. The script will automatically throw away old copies and then create new copies of gs-noX11-tiger, gs-X11-tiger, gs-noX11, gs-X11, and gs-X11-64. Leave them in the directories where the liposcript put them.

Then rename the package to reflect the current ghostscript version number. This step is common to all packages in MacTeX, so the method for doing so is described in the root TUG document for the entire build system.

Next change directory to /usr and rename /usr/local to /usr/local-temp. Copy the Ghostscript source folder to your machine and install Ghostscript using

```
make clean
./configure
make
sudo make install
```

In this step we are not interested in the binaries, but instead in the support files for Ghostscript.

Then change to the main Ghostscript directory of the MacTeX build system and type

```
sudo sh buildPackage.sh
```

This step will copy Ghostscript support files from /usr/local to the root directory of MacTeX's Ghostscript build folder. It will add the binaries constructed earlier, and add the Ghostscript fonts from source/fonts. Finally, it will build the final install package.

To finish, remove /usr/local and move /usr/local-temp back to /usr/local.

7. Adjusting the Ghostscript Install Package

The folder resources/English.lproj contains some files which are shown to the user during installation. One of these files is a brief Welcome document, a second is a longer ReadMe

file, and a third shows the License for the code. After the Ghostscript install package is constructed, you may need to revise these documents. For example, the Welcome document contains the release date. Usually TeX Live authors are overly ambitious and predict a release several months earlier than it actually occurs, and you'll need to change the release date from time to time.

A special script is provided to make changes in these support files without rebuilding everything. The script is called buildPackage1.sh. This script uses the already constructed root folder, and makes a new install package using this root, and possible new versions of Description.plist, Info.plist, and the documents in resources/English.lproj.