READ-ME

RICHARD KOCH

1. Ghostscript-9.53.3; Overview

Ghostscript is built both with and without X11 support. It is built on Big Sur for Arm and on Mojave for Intel, and then lipo is used to create universal binaries. So make certain that X11 is installed on both the Big Sur and the Mojave machines. The current official release of xQuartz supports only Intel and is used on Mojave. But version 2.8.0 rc1 is available and supports both Arm and Intel; we use it on the Big Sur machine.

Building Ghostscript is done in three steps. The first builds the binaries, which are placed in the folder "Binaries." Two binaries are built on Mojave, one with X11 support and one without this support. In addition, we build the Ghostscript dynamic library libgs.dylib.9.53.3, but only once because this library requires X11 support. These elements are placed in Ghostscript-9.53.3/Binaries/Intel with the names gs-noX11-intel, gs-X11-intel, and libgs.dylib.9.53

We repeat this step on Big Sur, again building two binaries and a library. These are placed in Ghostscript-9.53.3/Binaries/Arm with names gs-noX11-arm, gs-X11-arm, and libgs.dylib.9.53

In the second stage, Ghostscript is built and installed in /usr/local. Then a buildPackage script copies the resulting installation to the "root" folder in the Build Directory. This script replaces the binaries in this root folder with the carefully constructed binaries in Binaries.

The script makes modifications for TeX Fonts by Arnold-Voisin. The Arnold-Voisin additions were constructed in 2012.

In 2015, very extensive word was done to improve support for CJK fonts by Bruno Voisin, Norbert Preining, and others. Ultimately that led to a script by Norbert Preining called cjk-gs-integrate. See the ReadMe document in MyResources for details.

In the third stage, the root folder is used to construct an install package using the Ghostscript-Build template.

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

Date: February 23, 2021.

2. Building Binaries on Intel

To build binaries on Mojave, move /usr/local/bin to /usr/local/bin-temp and /usr/local/share to /usr/local/share-temp. Leave /opt alone for the X11 build, but change it to /opt-temp for the non-X11 build.

To build:

```
make clean
./configure --disable-compile-inits
make
```

Find gs in the source folder and rename it gs-X11-intel or gs-noX11-intel. Just after building the X11 version, build the library using the following instructions:

```
make so
```

Then go to the file sobin in the ghostscript-9,.53.3 source folder. Move the library "libgs.dylib.9.53" to the desktop.

Then move gs-X11-intel, gs-noX11-intel, and ligbs.dylib.9.53 to the machine where the final Ghostscript will be built and put all three in the folder Ghostscript-9.53.3/Binaries/Intel.

3. Building Binaries on Arm

The libpng problem raises its head on arm: ghostscript links with libpng, which has arm assembly code, but only for 32 bit arm and Apple's machines have 64 bit arm. So before building the sources on arm, find libpng/pngpriv.h in the ghostscript source folder, and add near the top the line

```
#define PNG_ARM_NEON_OPT 0
```

This fix is not required on Intel, and only required on arm when building with X11 support.

After this step, repeat the entire Intel build sequence, this time creating gs-X11-arm, gs-noX11-arm, and libgs.dylib.9.53. Move these to the machine where the full Ghostscript will be created and put all three in the folder Ghostscript-9.53.3/Binaries/Arm.

We finish by using lipo to construct universal binaries and then signing these binaries. Do this by issuing the following commands in Terminal:

```
sudo sh liposcript.sh
sudo sh liposcriptsign.sh
```

The result will be a folder named Universal-Binaries in the Binaries folder containing gs-noX11, gs-X11, and libgs.dylib.9.53.

4. Preparing the Root Folder

Final preparation is done on the most recent system, currently Big Sur..

Make sure the latest TeX Live is installed on this machine and type

```
sh buildTeXfonts.sh
```

This step will create a folder named "fonts" in "source/TeXfonts/ containing symbolic links to the pfb font files in the latest TeX Live distribution.

Find the folder source/TeXfonts/fonts. Rename the final folder to "Font" and copy it to BrunoContributions/Bruno-TeXLive/Resource/Font, possibly replacing the Font folder already there.

Next change directory to /usr/local and rename bin to bin-temp and and share to share-temp. Then switch to the ghostscript source folder and install Ghostscript using

```
make clean
./configure --disable-compile-inits
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. Then it will make modifications from Arnold-Voisin so Ghostscript will understand TeX files.

5. Constructing the Install Package

Make sure the items in MyResources are up to date, and in Terminal change directory to CreateInstallPackage and

```
sudo sh pkgbuildscript.sh
sudo sh productbuildscript.sh
sh signPackage.sh
cd NotarizePackage
sh sendnotarizerequest.sh
sh stapleresult.sh
```