

TUG-GHOSTSCRIPT

RICHARD KOCH

1. GHOSTSCRIPT

This is the portion of the TUG documentation which deals with making the Ghostscript package.

2. OVERVIEW

From 2017 on, MacTeX will support the platforms for which Apple provides security updates, and thus the three most recent versions of macOS. MacTeX is built on these oldest of these versions. In 2018, MacTeX is built on El Capitan and supports El Capitan, Sierra, and High Sierra.

Building Ghostscript is done in three steps. The first builds the binaries, which are placed in the folder “Binaries.” Two binaries are built El Capitan, one with X11 support and one without this support.

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 combined build binaries in Binaries.

The script makes modifications for TeX Fonts by Arnold-Voisin and for Japanese fonts by Kuroki-Yusuke, Bruno Voisin, and Norbert Preining. The Arnold-Voisin additions were constructed in 2012. The Japanese additions were constructed in 2015 with very difficult testing by Bruno and Norbert.

In the third stage, this root folder is used to construct an install package using the `GhostscriptBuild` template.

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.

In 2015, very extensive work was done to improve support for CJK fonts by Bruno Voisin, Norbert Preining, and others. Various documents created during that work are preserved in the folder `Bruno-Documentation`, in case it is necessary to refer back to it.

Date: May 13, 2019.

3. BUILDING BINARIES

Build binaries on Sierra. Build a version with X11 support and a version without this support.

Before building, move `/usr/local/bin` to `/usr/local/bin-temp` and `/usr/local/share` to `/usr/local/share-temp`. Leave `/opt` alone for X11 build, but change it to `/opt-temp` for non-X11 build.

To build:

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

Move all of these binaries to the “Binaries” folder on the machine used to build the actual install package.

4. PREPARING THE ROOT FOLDER

Next edit the file “`buildPackage.sh`” to reflect the current version of Ghostscript. Currently this file contains “9.27”, referring to folders installed by Ghostscript. Change this number appropriately.

Next 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 this latest TeX Live distribution.

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 --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 file. It will also make modifications from Kuroki-Yusuku, Bruno Voisin, and Norbert Preining so Ghostscript will understand Japanese files.

5. ADJUSTING FILES IN THE GHOSTSCRIPT INSTALL PACKAGE

This package 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.

None of these files appear in MacTeX, but they appear if you make an optional Ghostscript Install Package.

6. SIGNING, TIMESTAMPING, AND ADOPTING HARDENED RUNTIMES FOR BINARIES

Binaries in this package must be signed and notarized, and must adopt a hardened runtime. We do this for the binaries in `root/usr/local/bin` using the script “`sudo sh liposcripts/signal-lapps`”. This script signs `gs-noX11` and `gs-X11`. Note that `gs-X11` requires a third party library exception because it links with `X11`.

7. BUILDING THE FINAL INSTALL PACKAGE

The install package is constructed using scripts in the folder “`CreateInstallPackage`.” This folder also contains Resources and Scripts which will become part of the package. We first build `Ghostscript-9.27Start.pkg` by

```
sudo sh pkgbuildscript.sh
```

This script reaches up a level to find the root folder containing the package material. Next create `Ghostscript-9.27-Temp.pkg` by running a second script

```
sudo sh productbuildscript.sh
```

Finally, create a signed package `Ghostscript-9.27.pkg` by running the script

```
sh signPackage.sh
```

This script contains my Apple signing identity; future builders will first have to join Apple's Developer Support and get their own signing identity.

The `CreateInstallPackage` folder contains two other files and one other script. The files are `requirement.xml` and `distribution.xml`, and the script is `create_distribution.sh`. In future years, it may be enough to edit `distribution.xml` by hand. This file contains the `Ghostscript-9.27Start.pkg` name, and references to the background image, Welcome dialog, ReadMe dialog, and License dialog. It contains the identifier of the package, currently “`org.tug.mactex.ghostscript9.27`”. And it contains the restriction that the package can only be installed on machines running system 10.12 or higher.

But if necessary, the file can be recreated as follows. The system requirement is contained in `requirement.xml`. Edit this first. Then run the script `create_distribution.sh`. This script provides some information in `distribution.xml`, and outputs a sample `distribution.xml` file. But other items must be added by directly editing the file.

8. NOTARIZING THE FINAL INSTALL PACKAGE

Scripts to notarize `Ghostscript-9.27.pkg` are in the folder “NotarizePackage” inside “CreateInstallPackage.” The actual `Ghostscript-9.27.pkg` created above can be left where it is; it need not be moved to the new folder.

To notarize,

```
sh sendnotarizerequest.sh
```

There will be a delay when nothing is printed in Terminal as the package is uploaded to Apple. When the upload is complete, a message of the form

```
RequestUUID = 96f0932f-b50f-4d73-ad65-1220fd9c8efe
```

will be printed. This magic number can be used to discover what went wrong if notarization fails, so I recommend copying it to the file `requestUUID.tex`. There will be a pause of about ten minutes and then a notification from Apple and an email from Apple will arrive. If all is well, this message says that notarization succeeded. To add the certificate to the install package

```
sh stapleresult.sh
```

The package `Ghostscript-9.27.pkg` is then ready for uploading and release.

In case of failure, open the script `detailedinfo.sh` and notice that the magic UUID number is part of this script. Copy and paste the new number from `requestUUID.tex` to this script. Then run it.

```
sh detailedinfo.sh
```

The script will ask for a password. More on that in a second. Then it will report a web url. Use Safari to go to that url, where a very detailed error report will clearly state what went wrong.

The `detailedinfo` script runs a command named `xcrun` which communicates with developer support. Apple has adopted two-factor authentication security for such communication, but this security is not very convenient when running shell scripts. So developers can log into their Apple developer account and assign a password to `xcrun` and a few similar commands. Then when the script runs, typing this password is enough to run the command. Note that the `detailedinfo` script contains my account name and password. When someone else makes these packages, they will replace this information with their own personal information.