DOC-GHOSTSCRIPT-GS.DYLIB

RICHARD KOCH

1. Ghostscript

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

2. Overview

The TeX Live program dvisvgm converts dvi, eps, and pdf files to the XML-based vector graphic format SVG. Many users never use this utility.

The program dvisvgm must link to a Ghostscript dynamic library at runtime to perform a small number of operations. By deliberate design, MacTeX does not install any libraries in the system, since unexpected libraries can cause hard-to-diagnose problems if they become out of date. For those relatively few users who require these special dvisvgm operations, this package will install the required library.

3. Building Binaries

The main Ghostscript package documentation explains how to obtain and compile the Ghostscript source code. A section at the very end explains how to compile the Ghostscript dynamic library, and add its three files to the Ghostscript-9.50/Binary directory.

Copy from this directory the file named "libgs.dylib.950" and place it in the Binaries directory of this folder. The ghostscript directory will also contain symbolic links named "libgs.dylib" and "libgs.dylib.9", but these can be ignored. In future versions of Ghostscript, these names may change; a bug report was filed for Ghostscript 9.50 complaining that the names should end with ".dylib".

Note that the gs dynamic library requires X11 support.

4. Preparing the Root Folder

Next edit the file "buildPackage.sh" to reflect the current version of Ghostscript. Currently this file contains "9.50", referring to folders installed by Ghostscript. Change this number appropriately.

In the Ghostscript-9.50-libgs folder type

Date: March 12, 2020.

```
sudo sh buildPackage.sh
```

This step will build the appropriate root file for the installation.

When the package installs the library libgs, it will put libgs.dylib.9.50 in

```
/usr/local/share/ghostscript/9.50/libs
```

and will place two symbolic links pointing to this location in /usr/local/lib:

```
libgs.dylib
libgs.9.dylib
```

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. 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-9.50libgsStart.pkg by

```
sudo sh pkgbuildscript.sh
```

This script reaches up a level to find the root folder containing the package material.

All other items in this build directory are optional and never needed, because only Ghostscript-9.50libgs-Start will be used to create the final Ghostscript package.

However, if desired create Ghostscript-9.50libgs-Temp.pkg by running a second script

```
sudo sh productbuildscript.sh
```

Finally, created a signed package Ghostscript-9.50libgs.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-9.50libgsStart.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.ghostscript9.50libgs". And it contains the restriction that the package can only be installed on machines running system 10.13 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.

7. Notarizing the Final Install Package

Scripts to notarize Ghostscript-9.50libgs.pkg are in the folder "NotarizePackage" inside "CreateInstallPackage." The actual Ghostscript-9.50libgs.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-9.50.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
```

This command will return 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.

8. Testing

To test, run

```
dvisvgm -l
dvisvgm -V1
```

Both commands will produce a list of items. The libgs install was successful if the first list includes the first item below, and the second list contains the second item.

```
ps     dvips Postscript specials
Ghostscript 9.50
```