TUG-TEXLIVE-2019

RICHARD KOCH

1. TeXLive-2019

This is the portion of the TUG documentation which deals with making the TeXLive-2019 and TeXLive-2019-DVD packages. This is the key component of the build directory.

2. Creating the Package

The compile system must contain X11. Get it at X11Quartz if it isn't there.

On the compile system, issue the commands

```
cd /usr/local
sudo mv texlive texlive-old
```

Then make sure local configuration files for TeX Live are disabled, perhaps by writing

```
cd
cd Library
mv texmf texmfold
rm -R texlive
```

Then create a folder containing the tlpretest, which will later be updated with the same command. To do that, make a directory named texlive2019pretest and then

```
cd
cd texlive2019pretest
rsync -a --delete --exclude="mactex*" ftp.math.utah.edu::tlpretest .
```

The last line here obtains a copy from the utah mirror site for pretest releases. If time is of the essence, but not otherwise, download directly from the tug home site:

```
cd
cd texlive2019pretest
rsync -a --delete --exclude="mactex*" ftp.tug.org::tlpretext .
```

When first building binaries, it is necessary to submit them to Karl, wait for him to install them in the pretest and wait overnight for the pretest to make its way to mirror sites.

Then install TeX Live from the svn directory by

```
TEXLIVE_INSTALL_NO_CONTEXT_CACHE=1
export TEXLIVE_INSTALL_NO_CONTEXT_CACHE
cd
cd texlive2019pretest
sudo env  TEXLIVE_INSTALL_NO_CONTEXT_CACHE=$TEXLIVE_INSTALL_NO_CONTEXT_CACHE ./install-tl
```

This will run the graphic Unix install script. If not, change "./install-tl" to "./install-tl –gui tcl". The default values will be correctly chosen for the Macintosh. Change the paper size option to "letter". Then start installation.

When installation is complete, switch the default tlmgr repository to the final value using the command

```
sudo /usr/local/texlive/2019/bin/x86_64-darwin/tlmgr option repository \
     http://mirror.ctan.org/systems/texlive/tlnet
```

After these steps, the final TeXLive directory "root" can be built, using the command:

```
sudo sh buildPackage.sh
```

Finally repair /usr/local by

```
cd
cd /usr/local
sudo rm -R texlive
sudo mv texlive-temp texlive
```

This root file is all that is needed to make MacTeX.

3. 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/texlive/2019basic using the script "sudo sh liposcriptsignallapps". This script signs lz4, wget, and xz in tlpkg/installer, and over 100 apps in the app directory. Note that mf and xdvi require third party library exceptions because they link with X11.

We also remove four pieces of documentation which causes notarize problems in April and May of 2019. It would be good to discuss this with Apple and eventually remove these fixes.

4. 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 TeXLive-2019Start.pkg by

```
sudo sh pkgbuildscript.sh
```

This script reaches up a level to find the root folder containing the package material. Next create TeXLive-2019-Temp.pkg by running a second script

```
sudo sh productbuildscript.sh
```

Finally, created a signed package TeXLive-2019.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 TeXLive-2019Start.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.texlive2019". 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.

5. Notarizing the Final Install Package

Scripts to notarize TeXLive-2019.pkg are in the folder "NotarizePackage" inside "CreateInstallPackage." The actual TeXLive-2019.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 TeXLive-2019.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.