TUG-TEXLIVE-2020

RICHARD KOCH

1. MinimalTeX-2020

This is the portion of the TUG documentation which deals with making MinimalTeX-2020.

MinimalTeX was requested by Vafa Khalighi. It is based on the "minimal" TeX Live scheme. I make it available on my web site, but do not upload it to TUG or CTAN.

2. Creating the Package

Although the binaries are compiled on High Sierra in 2020, the actual package is built on Catalina. To construct the package, begin by renaming /usr/local/texlive. Then install TeXLive using the TeX Live install script.

cd texlive2020pretest

sudo ./install-tl –gui text

Select the "minimal" scheme

Change TEXDIR to /usr/local/texlive/2020minimal

Change TEXMFVAR to $HOME/Library/texlive/2020minimal/texmf-var

Change TEXMFCONFIG to $HOME/Library/texlive/2020minimal/texmf-config

Under Options, check "use letter size" and uncheck everything else

Then install.

After these steps, the "root" directory in the MinimalTeX folder is built using the commands

```
sudo sh buildPackage.sh
```

Most binaries will be signed, but some will not, so sign and adopt a hardened runtime for everything using

```
sudo sh liposcriptsignallapps
```

Date: March 12, 2020.

3. 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 MinimalTeX-2020-Start.pkg by

```
sudo sh pkgbuildscript.sh
```

This script reaches up a level to find the root folder containing the package material. Next create MinimalTeX-2020-Temp.pkg by running a second script

```
sudo sh productbuildscript.sh
```

Finally, created a signed package MinimalTeX.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 MinimalTeX-2020-Start.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.minimaltex2020". 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.

4. Notarizing the Final Install Package

Scripts to notarize MinimalTeX-2020.pkg are in the folder "NotarizePackage" inside "CreateInstallPackage." The actual MinimalTeX-2020.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 MinimalTeX-2020.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.