

TUG-TEXLIVE-2019

RICHARD KOCH

1. BASICTEX-2019

This is the portion of the TUG documentation which deals with making the BasicTeX-2019.

2. CREATING THE PACKAGE

To construct the package, begin by renaming /usr/local/texlive on Sierra and above. Then install TeXLive using the TeX Live install script. Select x86_64 binaries and the scheme “TeX Live Small Scheme”.

Under Options, check “use letter size” and uncheck everything else. In particular, uncheck “make format files.”

Change TEXDIR to /usr/local/texlive/2019basic

Change TEXMFHOME to \$HOME/Library/texmf

Change TEXMFVAR to \$HOME/Library/texlive/2019basic/texmf-var

Change TEXMFCONFIG to \$HOME/Library/texlive/2019basic/texmf-config

Then install.

After these steps, the “root” directory in the BasicTeX folder is built using the commands

```
sudo sh buildPackage.sh
```

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 30 apps in the `app` directory. Note that `mf` and `xdvi` require third party library exceptions because they link with `X11`.

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 `BasicTeX-2019Start.pkg` by

```
sudo sh pkgbuildscript.sh
```

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

```
sudo sh productbuildscript.sh
```

Finally, created a signed package `BasicTeX.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 `BasicTeX-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.basictex2019`”. 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 `BasicTeX-2019.pkg` are in the folder “`NotarizePackage`” inside “`CreateInstallPackage`.” The actual `BasicTeX-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 BasicTeX-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.