MACTEX OVERVIEW

RICHARD KOCH

1. Introduction and History

This is the root document for a series of related documents which explain how to construct MacTeX.

Wendy McKay conceived the idea of a Macintosh install package which would provide everything needed to use TeX on a Macintosh. After advocating the idea at a couple of earlier TUG conferences, she organized a lunch for Macintosh users at the TUG Practical TeX Meeting, 2005, in Chapel Hill, North Carolina. At that lunch, she pointed to Jonathan Kew and assigned the construction of the installer to him (how could Jonathan refuse!). Jonathan had to leave the conference the next morning, and we expected that he would construct the package over several weeks after he returned to England. Instead, Jonathan stayed up all night and had a package the next morning. Over breakfast, he willed maintenance of it to me.

Originally, MacTeX installed a TeX Distribution created by Gerben Wierda, based on teTeX. But in 2006, Thomas Esser, the author of teTeX, announced that the project was ending and recommended that users switch to TeX Live. Gerben Wierda then began revising his distribution to contain a mixture of teTeX and TeX Live, announcing the new distribution, gwTeX, in November, 2006 at the annual TUG meeting in Marrakesh, Morocco. But at that same meeting, Gerben held up a sign containing the words "I quit" and announced that he would no longer support his distribution. I then constructed test versions of MacTeX, one using the old teTeX, one using gwTeX, and another using the full TeX Live. To my delight, the TUG authorities pushed for the package based on the full TeX Live, and since 2007, that is what MacTeX installs.

2. Binaries

This document is provided inside a build tree for MacTeX. Do not rearrange folders in this tree. As MacTeX is constructed, these folders will gradually be filled with bits and pieces of the package, until finally the complete package is inside one of the folders. Some folders contain a subfolder named MiscDocs. This is extra material that might be useful, so I kept it but don't suggest reading it.

MacTeX supports the operating systems for which Apple is currently providing security patches. These are the current system and the two previous systems. Thus in 2021 we

Date: February 23, 2021.

support Mojave, Catalina, and Big Sur. Around September Apple will release the next operating system, and we always support that as well by working carefully with the beta release starting in June.

Binaries must be compiled twice, once on an Arm machine (this year running Big Sur) and once on an Intel machine (this year running Mojave). They are then combined using lipo, signed, and zipped up to be sent to TeX Live. It is useful to do the construction step on the most recent system, even if the compiles were done on older systems.

The first step in creating MacTeX and TeX Live for a yearly update is to compile the TeX Live Macintosh binaries. Two folders in the documentation cover this step: "Asymptote Build" and "Binary-2021'. We first describe obtaining the TeX Live source code and compiling the binaries for this system.

Inside the "Binary" folder is a TextEdit document named BuildStatus-2021. This contains the full instructions for making TeX Live binaries. It contains actual commands which can be copied and pasted into Terminal. The first step is to create an empty folder in your home directory named texlive2021dev. Then an rsync command in the document populates it with the current sources. This same command updates the directory when the sources change. The original step takes several minutes, but updating typically takes 30 seconds or less. Here is that command

```
cd
cd texlive2021dev
rsync -a --delete --exclude=.svn tug.org::tldevsrc/Build/source .
```

After the source is obtained, compiling is straightforward. But a few preliminary steps are required. Go to /usr/local and temporarily rename the bin, lib, include, share, and texlive folders so the system will not accidentally use some other code on the machine.

A small number of these binaries link with X11 libraries. They are pdfopen, pdfclose, xdvi-xaw, and xindy. Originally, Apple supported X11 on macOS with an install package named xQuartz. This package was only updated when new systems were introduced, and X11 users complained, so Apple abandoned the project and it became an independent open source project. The releases can by obtained by Googling xQuartz and going to their site. The release version on the site was last worked on in 2016. But on the left hand side of the page a button labeled "Releases" leads to version 2.0.8.beta4 which is universal with both Arm and Intel code. This will probably soon replace the 2016 code as the release version. In 2021, we used it on Arm and used the release version on Intel. TeX Live links a few programs with the X11 Library, so X11 must be installed but need not be running.

Now you are ready to compile. On an Arm machine, execute

```
cd
cd texlive2021dev/source
export STRIP="strip -u -r"
./Build --disable-arm-neon
```

The flag about arm-neon is needed when compiling on Arm because libpng has arm assembly code, but it only works on 32 bit processors and the Apple processor has 64 bits. It is likely that libpng will be updated in the future and this flag may no longer be necessary.

If there is an error, then it must be reported to TeX Live and an unpleasant debugging session will commence. But if there is no error and the build passes all tests automatically done at the end, then there will be a folder in texlive2021dev/source/inst/bin with a name like aarch64-apple-darwin20.3.0. Move the entire folder to the directory Binary/Binary2021/RawCode on the machine used for the final construction of MacTeX.

On the Intel machine, the instruction to compile will be

```
cd
cd texlive2021dev/source
export STRIP="strip -u -r"
./Build --enable-xindy CLISP=/Users/koch/clispold/clisp-build/clisp
```

This command will produce a folder in texlive2021dev/source/inst/bin with a name like x86_64-apple-darwin18.7.0. Move this entire folder to the directory Binary/Binary2021/RawCode.

Note that this command builds xindy. That build requires Lisp. For reasons explained next, we only compile xindy on Intel ; it still runs on Arm using Rosetta.

3. Lisp

When xindy was first added to the TeX Live toolchain, we were told to first compile clisp-2.49 and put it in a spot which could be called by the build system. This procedure worked.

Shortly afterward, we needed to support both PowerPC and Intel processors. Compiling xindy produces both xindy and two support files xindy.mem and xindy.run. It turned out that the support files depended on the processor. I suspect that is because the PowerPC and Intel chips had different endedness. The author of Xindy had to modify the code to use the correct support file when run.

Later we dropped support for PowerPC and this issue went away. But soon a new version of clisp appeared and the author of xindy asked us to use it. I could never get that version to compile. So I continued to use clisp-2.49. This year I tried to compile the later clisp on Arm, ran into troubles, and confirmed from the internet that others had this trouble as well.

So I decided to only compile xindy on Intel and continue to use clisp-2.49. Then all the issues just described go away.

I no longer have notes about making clisp-2.49. I still have a zip file with the source code, which is about 10 megs in size. Rather than adding it to these notes, I recommend that whoever takes over this project from me face the issue of clisp again from scratch, asking first whether enough people use xindy to make spending time on the issue worth it.

4. Asymptote

One other binary requires special attention, asy or Asymptote. There is a folder named Asymptote in the folder containing the document you are reading. You should make a copy of this folder in your home directory. Currently the folder contains an instruction sheet, "AsymptoteBuild-2021", and a Sample folder to test the final result. Follow the instructions in the instruction sheet. They will explain how to download and compile four libraries used by asymptote. This needs to be done only once a year. The asymptote source itself is in the TeX Live source and obtained when you obtain that source. "AsymptoteBuild-2021" explains how to compile it. On Arm, rename the final binary "asy-Arm" and on Intel name it "asy-Intel". Move these binaries to the machine used to create MacTeX and its Binary/Binary-2021/RawCodeAsymptote folder.

5. Packaging Universal-Darwin

After binaries are built on Arm and Intel machines, it is time to package them into a universal-darwin directory. We will produce universal-darwin.tar.xz, which is then sent to Karl Berry to add to TeX Live.

This will involve using some scripts in the MacTeX build system. Each year it is important to examine these scripts and modify them appropriately. Some contain the year, currently 2021, or the version number of an operating system, like 20.3.0, or restrictions on which systems can use MacTeX, like "Mojave or higher".

The required scripts for the current task are in Binary/Binary-2021. They are named lipomain.sh and lipoharden.sh. Run these scripts in Terminal by changing to the directory they are in and typing

```
sudo sh lipomain.sh
sudo sh lipoharden.sh
```

This will create universal-darwin.tar.xz in the directory, which can be sent to TeX Live.

How does lipomain work? A new folder named universal-darwin is created. The lipomain script lists all files in x86_64-darwin. Some are binaries, some are scripts, and some are symbolic links. If a file is a binary, the script looks in aarch64-apple-darwin for a file with the same name, and if found, then the two files are lipoed together and the result is placed in universal-darwin. If the Arm directory doesn't have the file, then a copy of the Intel version is placed in universal-darwin. If a file is a script or link, then a copy is placed in universal-darwin.

The script lipoharden has three tasks. It signs all binaries, and indicates that they have hardened runtimes. Then if a file requires exceptions, it assigns these exceptions. Finally it tars universal-darwin and then compresses it with xz. Note that xz is the standard compression form used by the TeX Live folks.

Since the Macintosh does not come with a copy of xz, it is necessary to compile it and place a copy of the binary in the Binary-2021 directory so the script can find it. We'll describe compiling xz in the next section.

Open the lipoharden script in an editor. The first thing you notice is that it uses the "file" command to determine whether a file is a link, a script, an Intel-only binary, or a universal binary. Only the last two types of files are signed.

Next notice that for Intel-only binaries like xindy, file(xindy) outputs "xindy: Mach-O 64-bit executable x86_64". But for universal binaries like tex, file(tex) outputs a longer string which only starts with the script "tex: Mach-O universal binary" and then continues with text which we ignore.

These files are signed using the command

```
codesign -s "Developer ID Application: Richard Koch"
```

Another person running this script would have to pay Apple $100 a year to become an Apple Developer. They would then obtain two licenses called "Developer ID Application" and "Developer ID Installer". The first license is used for applications, and the second will be used soon for MacTeX and other install packages. When these licenses are obtained, information about them is stored in the computer's Keychain. This information is accessed to sign the code. Therefore as written, the script will only work for me, and another person will need to change to their own name and insure that the keychain has appropriate information from Apple. Incidentally, there is a fancy way to move developer keychain information from one computer to another. Thus if I buy and use a new machine, the scripts won't work until I move my keychain developer information using the fancy method.

Notice that the signing commands have a flag

```
--options=runtime
```

This flag causes the application to adopt Apple's "hardened runtime." Such an application is not allowed to perform certain dangerous tasks like accessing the camera, or the user's location, or the user's address book. The application also cannot link with a third party library, or execute JIT-compiled code. A full list of prohibited actions can be found in the document "Hardened Runtime Entitlements" in the Binary-2021 directory.

However, for each prohibition, the application can file an "exception." These exceptions are always automatically granted. So the idea is that Apple doesn't restrict programs in any way, but if a programmer doesn't intend to use a dangerous feature, they the programmer can make sure that even illegal code inserted by an attacker cannot use that feature.

After signing all binaries, the script lipoharden considers a few applications that require exceptions. These are signed again, and this time an additional flag "–entitlements" lists a file containing the required entitlements for that program. Three binaries (mf, xdvi-xaw, and dvisvgm), link with an X11 library which is created by a third party, so they need an entitlement for that purpose. Four applications related to luatex require an entitlement to run JIT code.

6. xz and wget

Certain portions of the TeX Live Unix Install Script, and the tlmgr code in TeX Live, use wget and curl to download code, and use xz to compress and decompress files. The Mac already has curl, so nothing has to be done about it. But early each year, before seriously working on the binaries, we must compile wget and xz and send the resulting universal code to Karl. The folder xz_and_wget in the Binary directory contains all scripts and tools to do this. Incidentally, TeX Live requires only limited operations, so the programs are configured with many flags which turn off features. I cannot compile wget without these flags.

To compile, search the internet for source code and obtain wget-1.21 and xz-5.2.5 as tar files. Uncompress. Compile xz using the instructions

```
./configure --disable-nls --disable-shared --disable-threads
make
strip src/xz/xz
cp src/xz/xz xz.arm64
```

On Intel, replace the last xz.arm64 with xz.x86_64-darwin. Place these two binaries in the RawCode directory.

Compile wget using the instructions below. All the flags for configure should be on the same line.

```
./configure --enable-ipv6 --disable-iri --disable-nls --disable-ntlm
--disable-pcre --disable-pcre2 --without-libiconv-prefix
--without-libintl-prefix --without-libpsl --without-libuuid
--without-ssl --without-zlib
make
strip src/wget
cp src/wget wget.arm64
```

On Intel replace the last line with wget.x86_64-darwin. Place these two binaries in the RawCode directory.

Add the xz binary to the directory xz_and_wget so the final script can use it. Then execute

```
sudo sh lipomain.sh
sudo sh lipoharden.sh
```

This will create a file universal-darwin.tar.xz. Send it to Karl Berry. Examine the scripts. Notice that they contain nothing new.

7. Install Packages

A glance at the MacTeX-Doc-2021 directory reveals that most folders install one or another version of TeX, or a piece of the full MacTeX. All of these folders are constructed in the same way, so it suffices to describe the specifics in a single case. Let us concentrate on BasicTeX-2021.

The folder for this item contains a directory named "root". This directory will eventually contain a duplicate of part of the Mac file tree, containing exactly the new material to be installed on that machine. For example, the root folder for BasicTeX-2021 will contain root/usr/local/texlive/2021basic/... and the string of dots will be the full contents of BasicTeX. The BasicTeX subfolder of MacTex-Doc-2021 contains scripts which construct this root folder, and then scripts which create the install package that installs this root material, and finally scripts which sign and notarize this package.

To create this root folder, we will install BasicTeX on our machine using the TeX Live Unix Install Script. This script is available at `https://tug.org/texlive/acquire-netinstall.html` by clicking the link titled "install-tl-unx.tar.gz". The pretest of Tex Live contains a similar script.

Start by renaming /usr/local/texlive to, say, /usr/local/texlive-temp. The install script will create a new /usr/local/texlive which we will copy to our root directory. The Unix install script has a textual form and a somewhat primitive gui form. We will use the textual form because it gives us more control. So in Terminal change to the directory containing the install script and type

```
sudo ./install-tl --gui=text
```

On recent versions of macOS, two dialogs appear claiming that wget and xz cannot be verified as from known developers. Click "OK" in each case and the installation will proceed normally. The following text will appear in Terminal:

```
<B> set binary platforms: 1 out of 16

 <S> set installation scheme: scheme-full

 <C> set installation collections:
     40 collections out of 41, disk space required: 7189 MB

 <D> set directories:
   TEXDIR (the main TeX directory):
     /usr/local/texlive/2021
   TEXMFLOCAL (directory for site-wide local files):
     /usr/local/texlive/texmf-local
   TEXMFSYSVAR (directory for variable and automatically generated data):
     /usr/local/texlive/2021/texmf-var
   TEXMFSYSCONFIG (directory for local config):
     /usr/local/texlive/2021/texmf-config
   TEXMFVAR (personal directory for variable and automatically generated data):
     ~/Library/texlive/2021/texmf-var
   TEXMFCONFIG (personal directory for local config):
     ~/Library/texlive/2021/texmf-config
   TEXMFHOME (directory for user-specific files):
     ~/Library/texmf

 <O> options:
   [ ] use letter size instead of A4 by default
   [X] allow execution of restricted list of programs via \write18
   [X] create all format files
   [X] install macro/font doc tree
   [X] install macro/font source tree
   [ ] create symlinks to standard directories

 <V> set up for portable installation

Actions:
 <I> start installation to hard disk
 <P> save installation profile to 'texlive.profile' and exit
 <Q> quit

Enter command:
```

We first change any default answers that are wrong for us. After that, the installation proceeds without further help. It downloads and installs a large number of packages over the network.

Let us look at the questions. The install script will pick the binaries suitable for the machine being used. In past years it would pick x86_64-darwin but starting in 2021 it will pick universal-darwin. Using the first set of options, we can add additional binary sets to the installation, but we won't.

The second "S" choice picks an installation scheme. By default, everything is installed and that is what MacTeX will do. But for BasicTeX we select a smaller set of packages. So type S and push return and the display will change to the following:

```
Select scheme:

 a [X] full scheme (everything)
 b [ ] medium scheme (small + more packages and languages)
 c [ ] small scheme (basic + xetex, metapost, a few languages)
 d [ ] basic scheme (plain and latex)
 e [ ] minimal scheme (plain only)
 f [ ] ConTeXt scheme
 g [ ] GUST TeX Live scheme
 h [ ] infrastructure-only scheme (no TeX at all)
 i [ ] teTeX scheme (more than medium, but nowhere near full)
 j [ ] custom selection of collections

Actions: (disk space required: 7189 MB)
 <R> return to main menu
 <Q> quit

Enter letter to select scheme:
```

Type "c" to pick the small scheme, which gives Basic TeX. As a matter of fact, Basic TeX came first in the Macintosh world, and then TeX Live copied our choices to form this scheme. Then push "R" to return to the main choices.

The next question can be ignored. But we need to change some of the directory choices. Actually the TeX Live people have been kind to the Macintosh and the default choices in our list are the correct ones for the full MacTeX.

I won't show the choice dialog, but will walk our way through it. The first directory is the location where TeX will be installed. MacTeX installs in a directory named 2021, but BasicTeX installs in a directory named 2021basic so both installations can coexist. So change the first option to /usr/local/texlive/2021basic. The script automatically changes several other names. But we must change options 5 and 6 from " /Library/texlive/2021/texmf-var" to " /Library/texlive/2021basic/texmf-var" and from " /Library/texlive/2021/texmf-config" to " /Library/texlive/2021basic/texmf-config".

Next we change the "options". Select "use letter size instead of A4 by default" and "allow execution of restricted list of programs via write18". Deselect all other options.

Now start the installation. When it completes, a message will be printed asking you to set up a symbolic link to the binaries. Ignore that message.

We now have BasicTeX and we want to move it to the "root" directory and then prepare that location to be used by our install package. This is done by running two scripts:

```
sudo sh buildPackage.sh
sudo sh liposcriptsignallapps
```

As usual, these scripts should be read at the start of a building season to make obvious changes for the new year. The first script is rather straightforward, so I won't comment on it. The second script is trickier. It signs all binaries in the distribution. There are three binaries just for tlmgr, iz4, wget, and xz. These are signed at the start of the script. The binaries are currently named lz4.x86_64-darwin, wget.x86_64-darwin, xz.x86_64-darwin because other binaries are present for Linux, etc. I haven't seen the 2021 pretest. It is possible that these names have changed to lz4.universal-darwin, wget.universal-darwin, and xz.universal-darwin. If so, the script needs a slight edit.

Next the script signs all universal TeX binaries and all Intel-only TeX binaries.

Finally, it resigns a few binaries which require entitlements.

8. Special Notes for Pretest

During the pretest period, it is convenient to download the entire pretest to your own computer. Then all versions of TeX can be created rapidly without downloading packages over and over from the internet. Updating the downloaded pretest takes little time. Create a folder named "texlive2021pretest" and populate it (and update it) using

```
cd
cd texlive2021pretest
rsync -a --delete --exclude="mactex*" ftp.math.utah.edu::tlpretest .
```

It takes a day or so for material to reach these servers from TUG, so in a pinch creating the local folder directly from TUG is allowed:

```
cd
cd texlive2021pretest
rsync -a --delete --exclude="mactex*" ftp.tug.org::texlive/tlpretest .
```

After these steps, texlive2021pretest will contain an appropriate install-tl script.

9. Important Note On BuildPackage Script

The TeX binary directory for Macintosh contains a symbolic link

```
man --> ../../texmf-dist/doc/man
```

The reason is that man packages on the Mac use a heuristic to find some such pages: look for a man entry in the directory containing the actual binary.

In 2019 and 2020, the configuration software which created BasicTeX and MacTeX added a third entry to texmf-dist/doc/man. This third entry was a symbolic link "man" which again pointed to ../../texmf-dist/doc/man. There is no such location.

This spurious entry was probably created by trying to add a new "man link" to the binary directory, but in such a way that the existing link was followed and a new link was added to its end.

Make sure this spurious link is not created in 2021.

10. Install Packages Part 2

Now we need to create the Basic TeX install package using the "root" directory we have just prepared. It turns out that the techniques we use for BasicTeX work the same way for all other packages. So this section explains the final steps once and for all for everything.

Change directory from MacTeX-Doc-2021/BasicTeX-2021 to MacTeX-Doc-2021/BasicTeX-2021/CreateInstallPackage. This directory contains two important folders: MyResources and scripts. The first folder contains several pages of text which the installer displays at the start of an install process. The second folder contains the postinstall script which runs at the end, and possibly auxiliary files used by this script. So both folders need editing at the start of each new year.

Creating an install package from root is very simple; just change to the CreateInstallPackage folder and run three scripts as follows:

```
sudo sh pkgbuildscript.sh
sudo sh productbuildscript.sh
sh signPackage.sh
```

The first line creates BasicTeX-2021-Start.pkg, the second creates BasicTeX-2021-Temp.pkg, and the final line creates BasicTeX-2021.pkg. This third item will be the release package after it is notarized.

As usual, these scripts must be edited for each new year. Examine the three scripts now. The final script is completely straightforward. The first script is also mostly straightforward. It locates the root folder and the Scripts folder, and identifies the package with a reverse-url, in this case org.tug.mactex.basictex2021. This url must change each year and serves to keep various yearly updates separate.

The more complicated script is productbuildscript. It builds an unsigned copy of the final install package, locates the startup text in MyResources, and uses requirement.xml and distribution.xml to "design the structure of the package." So we need to talk about these two xml files.

The file requirement.xml lists the lowest system on which the package can be installed. In 2021, that is Mojave and the file lists 10.14

The file distribution.xml gives the detailed structure of the distribution and will become more complicated when we deal with MacTeX, which is built from several subpackages. Because distribution.xml is complicated, there is a script named create_distribution.sh which can create a skeleton from the information in the Start package. This need only be run when packages are first created; later it is easiest just to edit distribution.xml. Read this file. The purpose of most lines will be clear, and other lines can just be copied and retained without full understanding (!!).

11. Notarizing

The final step sends the install package to Apple. It is not touched by human hands there, but it is searched for viruses and checked to make sure all included binaries are signed and have adopted a hardened runtime. If the package passes these tests, then an official "notarization" is "stapled" to the package, which can then be released.

Notice that install packages are flat files which have already been compressed, so no compression is needed. The file can be placed directly on a server. Notarization is straightforward:

```
sh sendnotarizerequest.sh
sh stapleresult.sh
```

The first of these commands sends the result to Apple. This can take a long time for large packages like the complete TeXLive. When sending is complete, the command will print a message with a code like

```
39ae5363-520c-4b15-abb6-7094332242e
```

Copy this result in the file requestUUID.tex, because it will be needed if notarization fails.

Then wait from an email from Apple that notarization was successful. If so, run the staple command and notarization is complete.

If the email reports that notarization failed, edit the script detailedinfo.sh to contain the requestUUID number, and then

```
sh detailedinfo.sh
```

Apple will send back a url. Use your browser to access this url, which will yield a very, very detailed error report leading to the exact reason that notarization failed.

There are two other final tricky points. Notice that one parameter of sendnotarizerequest.sh is

```
--username "koch@uoregon.edu"
```

This is the Apple ID of my developer account. Apple checks that the message is coming from a computer it knows about, so change to your own developer Apple ID.

Notice that the sendnotarization script calls altool and one parameter is

```
--password "xxxx-xxxx-xxxx-xxxx"
```

Here altool is a script which calls my developer account asking that a task be performed. All communication with Apple developer accounts is protected by Two Factor authentication, so a message should pop up asking for the magic number that was just sent to my iPhone. Obviously this is inconvenient for calls embedded in scripts. So Apple allows developers to ask for passwords for a small number of utilities. This is done by accessing the developer site. If a developer changes their main machine or their developer password, then all of these passwords become invalid and must be reset. Thus xxxx-xxxx-xxxx-xxxx is my password and must be reset by you.

12. Read-Me Files

This documentation should suffice for most of the remaining packages. So instead of continuing here, I'll write small Read-Me files for these package folders explaining any idiosyncrasies.

13. Overview

We put three packages on CTAN: MacTeX-2021, BasicTeX-2021, and Ghostscript-9.53.3. In addition, we create Macintosh material for the TUG DVD.

The final MacTeX is created in the folder MacTeX-2021, but this is a very easy task because the package is formed by combining GUI-Applications-Start.pkg, Ghostscript-9.53.3-Start.pkg, Ghostscript-9.53.3-libgs-Start.pkg, and TeXLive-2021-Start.pkg. Notice that only the "Start" packages are required, so it is not necessary to notarize these four packages. In particular, TeXLive-2021-Start.pkg is enormous and notarizing it would take hours for the upload. All that can be skipped. On the other hand, notarizing the smaller pieces can save time by catching mistakes before the giant upload needs to occur.

TeXLive-2021 is easy to create because it is just like BasicTeX except that we install the full TeX Live.

The "root" folder for GUI-Applications is constructed in a different way explained in the Read-Me for that piece. It is an easy package to create.

The difficult step with the two Ghostscript packages is compiling the binaries, but the Read-Me files explain exactly how to do that. The two Ghostscript packages then form

two of the four pieces of MacTeX, but they also form the two pieces of the Ghostscript-Full package.

The remaining pieces are for the DVD. We cannot just put MacTeX on the DVD, because MacTeX has a complete copy of TeX Live, and the DVD also has another complete copy of TeX Live for Linux, Unix, and Windows. There isn't room for both.

So we use the TeX Live install script to install TeX Live from the DVD. Our instructions make this task as simple as possible, with only one easy choice for the user. Doing things this way has two additional advantages:

- Mojca Miklavec compiles TeX Live for older versions of macOS. These binaries are called x86_64-darwinlegacy. The Unix installer selects them if it discovers that it is installing to an older machine. So more people can install TeX when we use the Unix installer.

- The Unix installer allows users to select the install location. So the DVD can be used to install to the user's home directory if they do not have root permission.

However, this creates the problem that the MacTeX installer creates a Tex-Dist structure and the link /Library/TeX/texbin, while the Unix installer does not. So we create a very small install package named TeXDist for the DVD which merely adds the appropriate TeXDist structure for TeX Live. This package is also on the web page for users who install TeX Live using the Unix script but still want the advantages of /Library/TeX/texbin.

We also avoid the GUI-Applications package on the DVD and instead provide TeXShop, LaTeXiT, TeX Live Utility, and BibDesk (together with a little documentation) directly, so users can drag these to /Applications/TeX. In past years this had the advantage that we could put LaTeXiT, TeX Live Utility, and BibDesk on the DVD even though they were not signed and thus could not be part of a notarized MacTeX. However in 2021 LaTeXiT and TeX Live Utility are signed. It is possible that BibDesk can also be signed; we'll determine that in a few days. Nevertheless, we stick with putting these programs directly on the DVD.

We do put Ghostscript-9.53.3-Full on the DVD.

The folder "MacTeX-for-DVD" creates the full set of tools for the DVD. This simple task just gathers the pieces together without any scripts or fancy work.

Finally, we create the Macintosh portion of the DVD in the folder DVD. Herbert Schulz creates MacTextures, which forms the majority of the material. We add the material from MacTeX-for-DVD, and then use a simple script to create an iso which is sent to Manfred Lotz. He adds it to the material for other platforms and negotiates with a manufacturer to create the DVD.

14. GUI Programs

MacTeX distributes four GUI programs, TeXShop, LaTeXiT, TeX Live Utility, and BibDesk. Just before building, it is best to make sure all are up to date. I recommend doing this in the following order.

- The location GUI-Applications-apps should contain all four applications. Make sure each is up to date. Then rebuild the GUI-Applications package.
- The four applications should also be placed in MacTeX-for-DVD/MacTeX-Install. Nothing has to be rebuilt here.
- The entire MacTeX-Install folder should be moved to DVD/MacTeX-Install. Then rebuild the DVD.

15. Summary

This completes the documentation. Look for those small READ-ME folders in the less documented packages. Good luck.