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. The MacTeX Build Tree

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.

The folder BinaryDoc-2022 contains the documentation for actually building the TeX Live Binaries. This folder contains two subfolders named "Binary" and "xz_and_lz4". These two

Date: March 4, 2022.

folders should be at the root level of the MacTeX build tree, aside the folders BasicTeX-2022, DVD, etc. They are temporarily packaged inside BinaryDoc-2022 so all the information about building binaries is in one place. After digesting the documentation, and before building, move them to their proper position.

3. Binaries

We first give an overview of building binaries. More specific information is in three documents in BinaryDoc-2022. The file "BuildStatus-2022" explains how to build the individual Arm and Intel binary sets. These instructions build everything except the application asy, that is, Asymptote.

The folder BinaryDoc-2022 contains a folder named "Binary", which has a subfolder named "AsymptoteBuild". This folder has a document named "AsymptoteBuild-2022" which explains how to build the Arm and Intel asy binaries.

Finally, BinaryDoc-2022 contains a document named "Complete-Build-Instructions" which explains how to combine the Arm and Intel binaries to form universal-darwin binaries, how to sign these binaries, how to adopt a hardened runtime, and how to submit the binaries to TUG. These steps require several scripts which are bundled inside the Binary directory.

Primary documentation for building is in the three documents just listed, but we give an overview here and a miscellaneous collection of facts which may help explain the process.

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 2022 we should support Catalina, Big Sur, and Monterey. Since it was easy, we actually go back a year and support Mojave, Catalina, Big Sur, and Monterey. 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. This year it was done on Monterey.

The TextEdit document BuildStatus-2022 contains the full instructions for making Arm and Intel binaries. It contains actual commands which can be copied and pasted into Terminal. First create an empty folder in the home directory of each machine used to make arm or intel binaries. Name this folder texlive2022dev. Use an rsync command to populate it with the current sources. This same command updates the directory when the sources change. The original step takes many minutes, but updating typically takes a minute or less. Here is that command

```
cd
cd texlive2022dev
rsync -a --delete --exclude=.svn tug.org::tldevsrc/Build/source .
```

After the source is obtained, compiling is straightforward. One preliminary step is 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 be obtained by going to `https://www.xquartz.org`. In 2020-2021, extensive work on xQuartz converted the entire release to native code for both Intel and Arm. Be sure to use this version. TeX Live links a few programs with the X11 Library, so X11 must be installed but need not be running.

The most up to date instructions on making binaries is in BuildStatus-2022. We will summarize the steps here for completeness, but recommend using the instructions in that file.

On an Arm machine, execute

```
cd
cd texlive2022dev/source
export STRIP="strip -u -r"
./Build --disable-arm-neon --enable-xindy CLISP=/Users/koch/CLisp/bin/clisp
```

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. Strangely, libpng in TeX Live has still not been updated in 2022 to solve this problem.

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 texlive2022dev/source/inst/bin with a name like aarch64-apple-darwin20.6.0. Move the entire folder to the directory Binary/RawCode on the machine used for the final construction of MacTeX.

On the Intel machine, the instruction to compile will be

```
cd
cd texlive2022dev/source
export STRIP="strip -u -r"
./Build --enable-xindy CLISP=/Users/koch/clispold/clisp-build/clisp
```

This command will produce a folder in texlive2022dev/source/inst/bin with a name like x86_64-apple-darwin18.7.0. Move this entire folder to the directory Binary/RawCode.

4. Lisp

The program xindy requires CLisp to build, but not to run. Before 2022, we used a prebuilt clisp-2.49 on Intel because we had trouble building later versions of CLisp. But in the summer of 2021, instructions for building the most recent version were sent by Roberto Avanzi, roberto.av anzi@gmail.com. His instructions work on both Intel and Arm, so in 2022 our Xindy uses the latest CLisp and is universal. Details are in the BuildStatus document. Lisp must be compiled once each year, before starting TeX Live builds.

5. asy

The program asy is compiled separately. Follow the instructions in the AsymptoteBuild folder, producing binaries asy-Arm and asy-Intel.

6. Packaging Universal-Darwin

All remaining steps are done on the main computer, which ran Monterey in 2022. We must combine arm and intel binaries to create universal-darwin binaries. The process used to make these final binaries is somewhat complicated because binaries must be signed by a developer, must adopt a hardened runtime, and must be assigned correct dates so Karl's software knows how to process them. These steps are described in detail in the document "Complete-Build-Instructions" in the folder BinaryDoc-2022. We will not repeat them here, because there is a danger that the two sets of instructions will get out of sync. Below we list some miscellaneous points which may explain the process.

There is an Apple script named "lipo" which can combine arm and intel binaries to form a universal-binary. The command

```
lipo -archs pdftex
```

examines the binary, here pdftex, and lists the types of code it contains.

The program xz is a compression utility used often by TeX Live. 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 various build directories.

Files are signed using my developer signature Developer ID Application: Richard Koch. Another person running this script will 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, many scripts 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-2022 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.

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.

Certain portions of the TeX Live Unix Install Script and tlmgr code in TeX Live use curl to download code, and use xz to compress and decompress files. The program wget is often used instead of curl on Windows, but we don't use it on the Mac.

Early each year, before seriously working on the binaries, we must compile xz and lz4 and send the resulting universal code to Karl. The folder xz_and_lz4 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.

When Apple first adopted signing, we had to sign binaries as we constructed install packages like MacTeX and BasicTeX. But now, we sign binaries sent to Karl to install in TeX Live. So by the time we build MacTeX or BasicTeX, TeX Live will contain signed binaries and we can ignore signing issues as we build the final packages. The only exception is that after creating a package, we must sign it as a whole, and send it to Apple for notarization.

7. Install Packages

We now turn to the second phase of building, constructing the actual install packages. This is considerably easier and the various steps are controlled by scripts which do the work automatically.

Let us consider BasicTeX as an example of a typical install package. To create the package, we first install BasicTeX using the TeX Live Unix install script. We then run a script called "buildPackage.sh" in the folder BasicTeX-2022 in the MacTeX build system.

So

```
sudo sh buildPackage.sh
```

This script copies the installation to a folder named "root" in BasicTeX-2022. This "root" contains an exact copy of the software we want our install package to construct. For BasicTeX, for example, root has the file hierarchy

```
root --> usr --> local --> texlive --> 2022basic --> ...
```

where 2022basic then contains all the folders and subfolders of a typical BasicTeX installation.

The BasicTeX-2022 folder contains a subfolder named "CreateInstallPackage." This folder has some scripts and other files and folders. The folder "MyResources" contains four pages which will be displayed to the user during installation. The folder "scripts" contains the postinstall script which runs after the root material is installed. To create a package, run three scripts in order:

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

The first command creates the install package "BasicTeX-2022-Start.pkg". The second command adds the install package "BasicTeX-2022.Temp.pkg". The final command signs this and writes the third package "BasicTeX-2022.pkg". This third package will be the install package shipped to users. If you are curious, open these scripts in TeXShop to see what they do. They essentially call code provided by Apple to create install packages.

The CreateInstallPackage folder contains another folder named "NotarizePackage". This is used for notarization. To send the package BasicTeX-2022.pkg to Apple for notarization, run

```
sh sendnotarizerequest.sh
```

There will be a pause as the package is uploaded, and then a message will be reported by Apple containing a special code:

```
RequestUUID =  d61a8e89-4815-4578-99a0-e3caca961c94
```

Copy this code into the tex source file requestUUID.tex. Then there will be a pause. Finally, an email will arrive from Apple announcing either that notarization succeeded, or that it failed. If it succeeded, issue the command

```
sh stapleresult.sh
```

This will notarize the install package BasicTeX-2022.pkg which already exists up a level, and it will be ready to distribute.

However, if notarization failed, copy the magic code in RequestUUID to "detailedinfo.sh" and run the command

```
sh detailedinfo.sh
```

Apple will then send a very long url. Paste this into your Web browser to see a detailed error report explaining exactly what went wrong.

If you open these notarization scripts, you will see two flags; username and password. This username is the Apple ID of my Developer Account, which happens to be my email address koch@uoregon.edu. The password requires explanation.

Notarization is going to interact with Apple using an Apple program. Logging into your developer account is usually governed by two-factor authentication, but of course that isn't feasible when using a script. So instead Apple requires developers to log into their account and obtain a password for a few programs that contact their developer account. This password is then sent by scripts to allow various operations. This password is stable for months at a time, but can change. If the scripts suddenly fail, check to see if new passwords are required.

The description just given for BasicTeX also applies to MinimalTeX and several other packages. But a few packages like MacTeX-2022 and Ghostscript-9.55 are constructed by combining several smaller install packages. For example, MacTeX-2022 is made of four packages: Ghostscript-9.55, Ghostscript-9.55-libgs, GUI-Applications, and TeXLive-2022. In that case, building the final package is slightly different. Recall that when packages are created, we create three copies in order. For instance, for BasicTeX we create BasicTeX-2022-Start, and then BasicTeX-2022-Temp, and finally BasicTeX-2022. Only the first of these with "Start" in the title is required to create install packages from multiple subpackages. So to create MacTeX-2022, we change directory to its CreateInstallPackage folder. We then add the following four packages to this folder:

```
Ghostscript-9.55-Start.pkg
Ghostscript-9.55-libgs-Start.pkg
Gui-Applications-Start.pkg
TeXLive-2022-Start.pkg
```

We do not need to call the script "pkgbuildscript.sh"; instead we run

```
sudo sh productbuildscript.sh
```

to create MacTeX-2022-Temp.pkg, and then

```
sh signPackage.sh
```

to sign and obtain MacTeX-2022.pkg, and finally we notarize this package.

Notice that both TeXLive-2022 and MacTeX-2022 will be enormous, and thus take a long time to upload to Apple for notarization. But TeXLive-2022 need not be notarized because we only use the TeXLive-2022-Start package. This saves a lot of time.

8. A Special Task for TeXLive-2022

The package TeXLive-2022 contains the entire TeX distribution, including many documentation files. When it first became necessary to notarize our install packages, a (very) small number of files in the distribution caused problems. Luckily, Apple provides very detailed error reports, so we could modify or in some cases eliminate these bad files.

Consequently, when making TeXLive-2022, there is an additional step. The process begins, as usual, by installing TeX Live with the Unix install script. Then run two scripts:

```
sudo sh buildPackage.sh
sudo sh lipofix2022.sh
```

The first of these creates the "root" directory, as usual. The second reaches into this root and makes the desired modifications.

After this step, proceed as usual.

The file "lipofix2022.sh" is quite short. Inspect it to discover what is being done. In 2022 we had other problems and simply applied these fixes without much further investigation. Feel free to investigate.

Here is the list of problems as of 2022: Four of the many alegreya fonts cause problems. These are removed by the script. One of the ctan-o-mat document files has extension pkg; we change the extension to txt. The source/latex/stellenbosch folder contains a file named USlogos-4.0-src.zip. We remove this file.

9. Using the Unix Install Script

By the previous section, construction of an install package begins by installing a distribution using the Unix install script. We'll give an outline of how that works. Consult the TUG site for more details.

The Unix install script downloads the various packages of an installation from the web; this takes a long time. When pretests are being constructed, it is useful to download the entire pretest just once to your own computer. Then the download only needs to be updated when later versions of the package are made, saving time. And installations directly from your hard disk are much, much, much faster than over the network.

So the first step is to create a folder named "texlive2020pretest" in the home directory of your main computer. Populate this directory using

```
cd
cd texlive2022pretest
rsync -a --delete --exclude="mactex*" ftp.tug.org::texlive/tlpretest .
```

This downloads directly from the TUG site rather than from a pretest server, but Karl tells me that it is OK if we are using the result to build MacTeX, BasicTeX, and the rest.

The same command will update the texlive2020pretest folder. The following command is useful for doing a dry run which actually changes nothing if you are curious what will be updated:

```
cd
cd texlive2022pretest
rsync -a  --verbose --dry-run--delete --exclude="mactex*" ftp.tug.org::texlive/tlpretest .
```

To actually install a distribution, first go to /usr/local/texlive and erase any distribution already there with the same name. For instance, if texlive-2022 is being created, it is fine for /usr/local/texlive to contain other folders, but erase 2022 if it is already there. Similarly if you are creating BasicTeX-2022, then other distributions are fine, but erase 2022basic.

The install script is inside texlive2022pretest. Change directory to the contents and then run

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

The installer may see a previous distribution and ask if it should use it (y/n). Reply n.

Then the script will print a list of configuration questions and default answers. Consult the TUG site for a full explanation. Near the top of the display you will be asked to select an installation scheme. Later you will be asked to provide paths to important directories, and then a series of options.

For MacTeX itself, all of the default answers are already correct except possibly paper size. I select paper size = letter; the install package later changes this to a4 depending on the user's print preferences. Accept the remaining defaults and install.

For BasicTeX, a little more work is required, but not much. BasicTeX uses the "simple" iinstallation scheme, so select that. Then in the first choice of a directory path for the full installation, choose

```
/usr/local/texlive/2022basic
```

The only difference here is "2022basic" rather than "2022". Some of the remaining defaults will change slightly, but choices 5) and 6) will refer to 2022 in a spot that should be 2022basic.

After this, some options need to be changed for BasicTeX. Change the paper size option appropriately. Accept the option about restricting write privileges. Then turn off installing formats, documentation, etc.

Developers who have already used the Unix install script will now know enough to build all of MacTeX and can stop reading. We'll give just a little more detail for others.

Here is the actual text which the Unix install script will print.

```
<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 in 2022 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.

For BasicTeX, I won't show the directory choices directly, but will walk through it. The first directory is the location where TeX will be installed. MacTeX installs in a directory named 2022, but BasicTeX installs in a directory named 2022basic so both installations can coexist. Thus change the first option to /usr/local/texlive/2022basic. The script automatically changes several other names. But we must change options 5 and 6 from "~/Library/texlive/2022/texmf-var" to "~/Library/texlive/2022basic/texmf-var"

and from "~/Library/texlive/2022/texmf-config" to "~/Library/texlive/2022basic/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 for BasicTeX, but leave the defaults alone for MacTeX.

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.

When installation concludes, we move the distribution to a root directory with the command

```
sudo sh buildPackage.sh
```

10. 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.

The link is created correctly by TUG and the Unix installer; make sure our software doesn't try to do it again. This paragraph is probably not necessary for 2022, but is kept just in case.

11. 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.

12. DVD

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. 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.

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.

13. 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.

14. Summary

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