

# Building BOINC Client and Manager on Macintosh OSX

Written by Charlie Fenton and Robert Chalmers  
Last updated 1 October 2019

This document applies to BOINC version 7.15.0 and later. It has instructions for building the BOINC Client and Manager for Macintosh OSX. Information for building science project applications to run under BOINC on Macintosh OSX can be found [here](#).

**Note:** the information in this document changes from time to time for different versions of BOINC. For any version of BOINC source files, the corresponding version of this document can be found in the source tree at:

`boinc/mac_build/HowToBuildBOINC_XCode.rtf`

Contents of this document:

- Important requirements for building BOINC software for the Mac.
- Cross-Platform Development.
- Building BOINC Manager with embedded BOINC Client.
- Building BOINC Manager Installer.
- Code Signing the BOINC Manager Installer and Uninstaller
- Debugging and BOINC security.
- Debugging into wxWidgets.
- Installing and setting up Xcode.

**Note:** `setupForBoinc.sh` (described late in this document) runs `buildWxMac.sh` to build the wxWidgets library used by BOINC Manager. If you built the wxWidgets library with an earlier version of `buildWxMac.sh`, then you must rebuild it with the `buildWxMac.sh` included in the newer source tree. Otherwise, the BOINC Manager build will fail with linker (`ld`) errors.

## Important requirements for building BOINC software for the Mac

As of version 6.13.0, BOINC does not support Macintosh PowerPC processors. As of 7.15.0, BOINC is built entirely for 64-bit Intel, including the BOINC libraries.

You need to take certain steps to ensure that you use only APIs that are available in all the OS versions BOINC supports for each architecture. The best way to accomplish this is to use a single development system running OS 10.8.x or later and cross-compile for the various platforms. The remainder of this document describes that process.

**The above requirements apply not only to BOINC itself, but also to the WxWidgets, c-ares, cURL, openssl, freetype, ftgl and SQLite3 libraries, as well as all project applications.**

Be sure to follow the directions in this document to ensure that these requirements are

met.

Starting with version 7.15.0, the BOINC screensaver supports only Mac OS 10.7.0 and later.

## **Cross-Platform Development**

Apple provides the tools necessary to build BOINC on any Mac running OS 10.8.x or later.

You get these tools, including the GCC or Apple LLVM compiler and system library header files, by installing the Xcode Tools package.

**Building BOINC now requires Xcode Tools version 6.0 or later.**

You can download Xcode from Apple's App Store (it is large: over 4 GB). If you are a member of Apple's Mac Developer Program, you can also download it from Apple's web site: <http://developer.apple.com>.

Source files are now archived using git. For details, see:

<http://boinc.berkeley.edu/trac/wiki/SourceCodeGit>

## **Building BOINC Manager with embedded Core Client**

Note: building BOINC Manager 7.3.0 and later requires the OS 10.8 SDK or later.

BOINC depends on seven third-party libraries: wxWidgets-3.1.0, c-ares-1.13.0, curl-7.58.0, openssl-1.1.0g, freetype-2.9, ftgl-2.1.3-rc5 and sqlite-3.22.0. You can obtain the source files from the following URLs. Clicking on the first URL of each pair will download the tar file. The second URL will open the third party's home web page. On Mac OS X the tar file will usually be downloaded into the Downloads folder. You will need to expand the tar files by double-clicking on them, which will create a folder and place the appropriate files into that folder. You will need to move these folders later.

wxWidgets-3.1.0 (needed only if you are building the BOINC Manager):

<https://github.com/wxWidgets/wxWidgets/releases/download/v3.1.0/wxWidgets-3.1.0.tar.bz2>  
<http://www.wxwidgets.org>

sqlite-3.22.0 (needed only if you are building the BOINC Manager):

<https://www.sqlite.org/2018/sqlite-autoconf-3220000.tar.gz>  
<http://www.sqlite.org/>

curl-7.58.0:

<http://curl.haxx.se/download/curl-7.58.0.tar.gz>

`http://curl.haxx.se`

**c-ares-1.13.0 (used by curl):**

`https://c-ares.haxx.se/download/c-ares-1.13.0.tar.gz`

`http://daniel.haxx.se/projects/c-ares/`

**openssl-1.1.0g**

`https://www.openssl.org/source/openssl-1.1.0g.tar.gz`

`http://www.openssl.org/`

**freetype-2.9 (needed only if you are building the BOINC default screensaver or a project screensaver):**

<https://sourceforge.net/projects/freetype/files/freetype2/2.9/freetype-2.9.tar.bz2>

<http://www.freetype.org/>

**ftgl-2.1.3~rc5 (needed only if you are building the BOINC default screensaver or a project screensaver):**

`http://sourceforge.net/projects/ftgl/files/FTGL%20Source/2.1.3%7Erc5/ftgl-2.1.3-rc5.tar.gz`

<http://sourceforge.net/projects/ftgl>

XCode will automatically check compatibility back to OS 10.7 if the following are defined during compilation:

```
MAC_OS_X_VERSION_MAX_ALLOWED=1070
```

```
MAC_OS_X_VERSION_MIN_REQUIRED=1070
```

These are not done automatically by either the Xcode projects which come with wxWidgets-3.1.0, nor the AutoMake scripts supplied with wxWidgets-3.1.0, c-ares-1.13.0, curl-7.58.0, openssl-1.1.0g, freetype-2.9, ftgl-2.1.3~rc5 and sqlite-3.22.0. So be sure to use our special scripts to build these packages.

[1] Make sure you are logged into the Mac using an account with administrator privileges. Create a parent directory within which to work. In this description; we will call it BOINC\_dev, but you can name it anything you wish.

[2] Move the following 7 directories from the Downloads folder into the BOINC\_dev folder (omit any you don't need):

`c-ares-1.13.0`

`curl-7.58.0`

`openssl-1.1.0g`

`wxWidgets-3.1.0`

`freetype-2.9`

`ftgl-2.1.3~rc5`

sqlite-3.22.0

Important: do not change the names of any of these 7 directories. Remember that the names are case sensitive.

[3] If you have not yet done so, install Xcode and launch it once to accept the license agreement and complete the installation.

[4] Get the BOINC source tree from the repository, and put it in the same BOINC\_dev folder. To do this, type the following in Terminal (if you have problems, you may need to enter `sudo` and a space before the `git` command):

```
cd {path}/BOINC_dev/  
git clone https://github.com/BOINC/boinc.git boinc
```

(You may change the name of the `boinc` directory to anything you wish.)

The command above retrieves the source code from the HEAD / MASTER (TRUNK) or development branch of the git repository. For more information on getting the BOINC source code, see:

<http://boinc.berkeley.edu/trac/wiki/SourceCodeGit>

[5] Run the script to build the c-ares, curl, openssl, wxWidgets, freetype, ftgl and sqlite3 libraries as follows:

```
cd {path}/BOINC_dev/boinc/mac_build/  
source setupForBoinc.sh -clean
```

If you don't wish to force a full rebuild of everything, omit the `-clean` argument.

**Note 1:** Be sure to run the script using the `source` command. Do not double-click on the scripts or use the `sh` command to run them.

**Note 2:** This script tries to build all seven third-party libraries: wxWidgets-3.1.0, c-ares-1.13.0, curl-7.58.0, openssl-1.1.0g, freetype-2.9, ftgl-2.1.3~rc5 and sqlite-3.22.0. When the script finishes, it will display a warning about any libraries it was unable to build (for example, if you have not downloaded them.) To make it easier to find the error messages, clear the Terminal display and run the script again without `-clean`.

**Note 3:** `setupForBoinc.sh` runs `buildWxMac.sh` to build the wxWidgets library used by BOINC Manager. If you built the wxWidgets library with an earlier version of `buildWxMac.sh`, then you must rebuild it with the `buildWxMac.sh` included in the newer source tree. Otherwise, the BOINC Manager build will fail with linker (`ld`) errors.

**Note 4:** The `{path}` must not contain any space characters.

**Hint:** You don't need to type the path to a file or folder into Terminal; just drag the file or folder icon from a Finder window onto the Terminal window.

**Note 5:** To be compatible with OS 10.7, the screensaver must be built with Garbage

Collection (GC) supported (and without Automatic Reference Counting) , but Xcode versions later than 5.0.2 do not allow building with GC. To allow building with newer versions of Xcode while keeping backward compatibility to OS 10.7, the GIT repository includes the screensaver executable built with GC, while the Xcode project builds the screensaver with ARC (for newer versions of OS X.) The `release_boinc.sh` script (described later in this document) adds both the GC and ARC builds of the screensaver to the installer; the installer code selects the correct screensaver for the target version of OS X at install time.

[6] Build BOINC as follows:

BOINC itself is built using the **boinc.xcodeproj** file. You can either build directly in Xcode (more information below) or run the **BuildMacBOINC.sh** script:

```
cd {path}/BOINC_dev/boinc/mac_build/  
source BuildMacBOINC.sh
```

The complete syntax for this script is

```
source BuildMacBOINC.sh [-dev] [-noclean] [-all] [-lib] [-client]  
[-libc] [-c++11] [-help]
```

The options for BuildMacBOINC.sh are:

- dev            build the development (debug) build.  
                 default is deployment (release) build.
- noclean       don't do a "clean" of each target before building.  
                 default is to clean all first.

The following arguments determine which targets to build

- all            build all targets (i.e. target "Build\_All" -- this is the default)
- lib            build the six libraries: libboinc\_api.a, libboinc\_graphics\_api.a,  
libboinc.a, libboinc\_opengl.a, libboinc\_zip.a, jpeglib.a
- client        build two targets: boinc client and command-line utility boinccmd  
                 (also builds libboinc.a, since boinc\_cmd requires it.)
- libc          build using libc++ instead of libstdc++ (requires OS 10.7 or later)
- c++11        build using c++11 language dialect instead of default

Both -lib and -client may be specified to build eight targets (no BOINC Manager or screensaver.)

**Note 1:** **boinc.xcodeproj** in the BOINC\_dev/boinc/mac\_build/ directory builds BOINC. It can be used either with the `BuildMacBOINC.sh` script or as a stand-alone project.

The *Development* build configuration builds only the native architecture and is used for debugging. The *Deployment* build configuration builds a universal binary and is suitable for release builds. If there are any other build configurations, they should not be used as they are obsolete.

**Note 2:** To perform a release build under Xcode 6 or later when not using the `BuildMacBOINC.sh` script, select "Build for Profiling" from Xcode's Product menu. To save disk space, do **not** select "Archive."

**Note 3:** Using the `BuildMacBOINC.sh` script is generally easier than building directly in Xcode. The script will place the built products in the directory `BOINC_dev/boinc/mac_build/build/Deployment/` or `BOINC_dev/boinc/mac_build/build/Development/` where they are easy to find. Building directly in Xcode places the built products in a somewhat obscure location. To determine this location, control-click on *Products* in Xcode's Project Navigator and select "Show in Finder."

**Note 3a:** If built using XCode, the actual folder containing the products are shown in the right hand info navigation bar. However, If you built correctly using the Profiling option – and you should have, then you don't need to worry about it. Just proceed to install the binaries using the install script. Do not try and use XCode to install the app. See the notes below on Installing the binaries.

**Note 4:** As of version 7.15.0, BOINC is always built using `libc++`. Project applications built for `libstdc++` with newer versions of Xcode will not link properly with BOINC libraries built for `libc++`.

**Hint:** You can install multiple versions of Xcode on the same Mac, either by putting them in different directories or by renaming `Xcode.app` of different versions. You can then specify which version the `BuildMacBOINC.sh` script should use by setting the `DEVELOPER_DIR` environment variable using the `env` command. For example, if you have installed Xcode 6.2 in the Applications directory and renamed `Xcode.app` to `Xcode_6_2.app`, you can invoke the script with:

```
env
DEVELOPER_DIR=/Applications/Xcode_6_2.app/Contents/Developer
sh BuildMacBOINC.sh
```

The BOINC Xcode project has built-in scripts which create a text file with the path to the built products at either

`BOINC_dev/boinc/mac_build/Build_Deployment_Dir`

or

`BOINC_dev/boinc/mac_build/Build_Development_Dir`.

These files are used by the `release_boinc.sh` script, but you can also use them to access the built products directly as follows; open the file with TextEdit and copy the

path, then enter command-shift-G in the Finder and paste the path into the Finder's dialog.

The standard release of BOINC version 7.15.0 and later builds only for Macintosh computers with 64-bit Intel processors. The executables and libraries are built only for the x86\_64 architecture.

## Building BOINC Manager Installer

In order to execute BOINC Manager, you must install it using BOINC Manager Installer. Otherwise, you will encounter an error prompting for proper installation.

To build the Installer for the BOINC Manager, you must be logged in as an administrator. If you are building BOINC version number x.y.z, type the following in Terminal, then enter your administrator password when prompted by the script:

NOTE: The version numbers are contained in a little file in the boinc root folder named version.h.

NOTE: The word 'source' refers to the binary of that name. It is a command.

- - `cd {path}/BOINC_dev/boinc/`
- - `source`  
`{path}/BOINC_dev/boinc/mac_installer/release_boinc.sh x`  
`y z`

For Example: On Robert's particular Mac the dev directory is at

`/Users/robert/Documents/GitHub/ShanghaiTimes/boinc`

So in the boinc directory I type,

`source`

`/Users/robert/Documents/GitHub/ShanghaiTimes/boinc/mac_installer/release_boinc.sh 7 15 1`

If everything is OK then you will be asked to type in your machine password and it will proceed with no errors.

Substitute the 3 parts of the BOINC version number for x y and z in the above. For example, to build the installer for BOINC version 7.9.0, the command would be  
`source {path}/BOINC_dev/boinc/mac_installer/release_boinc.sh`  
`7 9 0`

This will create a directory "BOINC\_Installer/New\_Release\_7\_9\_0" in the BOINC\_dev directory, and the installer will be located in  
'{path}/BOINC\_dev/BOINC\_Installer/New\_Release\_7\_9\_0/boinc\_7.9.0\_macOSX\_x86\_64'.

The installer script uses the deployment (release) build of BOINC; it won't work with a

development (debug) build.

Which is why you must build with the Profile build if you use XCode.

You can find the current version number in the file

```
{path}/BOINC_dev/boinc/version.h
```

## Code Signing the BOINC Manager Installer and Uninstaller

Mac OS 10.8 introduces a security feature called Gatekeeper, whose default settings won't allow a user to run applications or installers downloaded from the Internet unless they are signed by a registered Apple Developer. The `release_boinc.sh` script looks for a file `~/BOINCCodeSignIdentities.txt` containing the name of a valid code signing identity stored in the user's Keychain. If this is found, the script will automatically sign the BOINC installer and BOINC uninstaller using that identity. For example, if your user name is John Smith, the first line of `~/BOINCCodeSignIdentities.txt` would be something like the following:

```
Developer ID Application: John Smith
```

If you wish to also code sign the installer package, add a second line to `~/BOINCCodeSignIdentities.txt` with the installer code signing identity. This would be something like the following:

```
Developer ID Installer: John Smith
```

If there is no `~/BOINCCodeSignIdentities.txt` file, then the script will not sign the installer or uninstaller. Code signing is not necessary if you won't be transferring the built software over the Internet. For more information on code signing identities see the documentation for the `codesign` utility, Apple's Code Signing Guide and Tech Note 2206.

MOJAVE and Subsequent Mac Builds Note:

It is strongly recommended that you read the [Apple Documentation on Notarization](#) carefully.

Before you can use the `altool` command, you must first set up an app-specific password, as explained in Apple's documentation at the [Apple Documentation on Notarization](#) website and at [Using app-specific passwords](#) in their support documentation.

General Notes Referring to this Process. These notes now need you to have processed your Notarization requirements.

As of OS 10.14 Mojave, Apple has introduced a new level of security which Apple calls "notarization". Under OS 10.14, the only difference is that Gatekeeper adds the sentence "Apple checked it for malicious software and found none." However, Apple has warned: "In an upcoming release of macOS, Gatekeeper will require Developer ID–signed software to be notarized by



Apple."

To notarize the installer and uninstaller:

- Use the command line tools in Xcode 10 or later
- Provide valid application & installer code signing identities as above
- In Terminal":

```
$ xcrun altool --notarize-app -t osx -f {path to ...macOSX_x86_64.zip}
```

```
--primary-bundle-id edu.berkeley.boinc.Installer -u {userID} -p {password}
```

- After a few minutes, check whether the notarize-app request succeeded:

```
$ xcrun altool --notarization-info {UUID from last step} -u {userID} -p {password}
```

- If the notarize-app request succeeded, attach tickets to top level applications:

```
$ xcrun stapler staple {path to "...macOSX_x86_64/BOINC Installer.app"}
```

```
$ xcrun stapler staple {path to "...macOSX_x86_64/extras/Uninstall BOINC.app"}
```

- delete or rename the original ...macOSX\_x86\_64.zip}
- Run this ditto command again to create a new ...macOSX\_x86\_64.zip containing the updated (notarized) BOINC Installer.app and Uninstall BOINC.app:

```
$ ditto -ck --sequesterRsrc --keepParent boinc_$1.$2.$3_macOSX_$arch
```

```
boinc_$1.$2.$3_macOSX_$arch.zip
```

- Note: if you are running stapler under OS 10.13 and get an error 68, the local CRL cache may have become corrupted. You can resolve this by either running stapler under MacOS 10.14 Mojave or by running this command under OS 10.13:

```
$ sudo killall -9 trustd; sudo rm /Library/Keychains/crls/valid.sqlite3
```

- for more information:

```
$ xcrun altool --help
```

```
$ man stapler
```

## Debugging and BOINC security

Version 5.5.4 of BOINC Manager for the Macintosh introduced new, stricter security measures. For details, please see the file

`BOINC_dev/boinc/mac_installer/Readme.rtf` and

<http://boinc.berkeley.edu/sandbox.php> and

<http://boinc.berkeley.edu/trac/wiki/SandboxUser>

The LLDB debugger can't attach to applications which are running as a different user or group so it ignores the `S_ISUID` and `S_ISGID` permission bits when launching an application. To work around this, the BOINC *Development* build does not use the special `boinc_master` or `boinc_project` users or groups, and so can be run under the debugger from Xcode. This also streamlines the development cycle by avoiding the need to run the installer for every change. (To generate the development build under Xcode, choose "Build" from the product menu, or enter command-B on the keyboard.)

To restore the standard ownerships and permissions, run the installer or run the `Mac_SA_Secure.sh` shell script in Terminal (the comments in this script have instructions for running it.)

For information on interpreting crash dumps and backtraces, see [Mac Backtrace](#).

## Debugging into wxWidgets

The BOINC Xcode project links the BOINC Manager with the non-debugging (Deployment) build of wxWidgets for the Deployment build configuration of the Manager, and with the debugging (Development) build of wxWidgets for the Development build configuration of the Manager. You should use the Development build of the Manager and wxWidgets for debugging; this allows you to step into internal wxWidgets code. With the Development build, you can even put breakpoints in wxWidgets; this is most easily done after stepping into a function in wxWidgets source file containing the code where you wish to put the breakpoint.

## Installing and setting up Xcode

If Xcode is obtained from the Apple Store then it will be installed automatically into the Applications folder. Double-click on the installed Xcode icon to run Xcode. Xcode will display a dialog allowing you to finish the installation; you must do this before running BOINC's build scripts. (Some versions of Xcode may not display this dialog until you open a file with Xcode.)

NOTE to building with XCode.

The general instructions in the mac\_build folder in the file `HowToBuildBOINC_XCode.pdf` should also note that if you want to build using XCode in its GUI implementation - not command line - alone, you need to put all the downloaded libraries in the folder directly above the build folder. For example:

All of these external programs should be unpacked into their folders here:

```
wxWidgets-3.1.0 s-3.1.0.tar.bz2
sqlite-3.22.0
curl-7.58.0:
c-ares-1.13.0 (used by curl):
openssl-1.1.0g
freetype-2.9
ftgl-2.1.3~rc5
```

They need to go into the top GitHub directory on your system and be unpacked there. Otherwise XCode as the project is currently set up, will not find the libraries. They need to be unzipped there of course.

`/Users/<your name>/GitHub/`

So that when you build in XCode, they can be found in the existing setup, which looks for them in - for example -

```
USER_HEADER_SEARCH_PATHS
```

```
= ../../curl-7.58.0/include ../../openssl-1.1.0g/include ../lib/**
```

```
/Users/<user>/Documents/GitHub  
  /boinc  
  /c-ares-1.13.0  
  /curl-7.58.0
```

... and so on for the other required programs.

Below the boinc folder there is of course the 'boinc/mac\_build' folder.

When you have everything in place, and the appropriate scripts run to build the additional required libraries and binaries, you can build with XCode. Building the BOINC\_client should result in the following in the XCode debug window if all is well.

The result is

```
01-Apr-2019 09:21:01 [---] OS: Mac OS X 10.14.4 (Darwin 18.5.0)  
01-Apr-2019 09:21:01 [---] Memory: 16.00 GB physical, 67.34 GB virtual  
01-Apr-2019 09:21:01 [---] Disk: 931.19 GB total, 64.59 GB free  
01-Apr-2019 09:21:01 [---] Local time is UTC +1 hours  
01-Apr-2019 09:21:01 [---] Last benchmark was 17987 days 08:21:01 ago  
01-Apr-2019 09:21:01 [---] No general preferences found - using defaults  
01-Apr-2019 09:21:01 [---] Preferences:  
01-Apr-2019 09:21:01 [---]     max memory usage when active: 8192.00 MB  
01-Apr-2019 09:21:01 [---]     max memory usage when idle: 14745.60 MB  
01-Apr-2019 09:21:01 [---]     max disk usage: 64.50 GB  
01-Apr-2019 09:21:01 [---]     don't use GPU while active  
01-Apr-2019 09:21:01 [---]     suspend work if non-BOINC CPU load exceeds 25%  
01-Apr-2019 09:21:01 [---]     (to change preferences, visit a project web site or select  
Preferences in the Manager)  
01-Apr-2019 09:21:01 [---] Setting up project and slot directories  
01-Apr-2019 09:21:01 [---] Checking active tasks  
01-Apr-2019 09:21:01 [---] Setting up GUI RPC socket  
01-Apr-2019 09:21:01 [---] Checking presence of 0 project files  
01-Apr-2019 09:21:01 [---] This computer is not attached to any projects  
01-Apr-2019 09:21:01 Initialization completed  
01-Apr-2019 09:21:01 [---] Suspending GPU computation - computer is in use
```