XFree86 on Darwin and Mac OS X

Torrey T. Lyons

11 December 2000

1. Introduction

XFree86, a free implementation of X Windows, has been ported to Darwin and Mac OS X. This document is a collection of information for anyone experimenting with running X Windows on Apple's next generation operating system. Please send any comments on this page to <torrey@mrcla.com>. If you are also working on modifications to the Darwin X server, please let me know.

2. Hardware Support and Configuration

The Darwin port of XFree86 uses the IOKit for device driver loading and discovery, rather than the XFree86 module loader. Because of this, the XFree86 configuration file is not used on Darwin systems. The descriptions found elsewhere of devices supported by XFree86 for other operating systems also do not apply.

XFree86 for Darwin will work with any video card that you have an IOKit driver for. If you are running Mac OS X, this typically means any card that has been shipped by Apple. If a card works for you with the Mac OS X GUI, it will work on the same machine with XFree86. If you have a pure Darwin installation, your hardware support may be somewhat limited because Apple does not include all its hardware drivers with Darwin. IOKit drivers are typically found in /System/Library/Extensions/ or /Library/Extensions/.

Multi-button and scroll wheel mouse support works well with all USB mice that I have tested. I have done no testing with serial mice.

3. Getting Started

Following are instructions for building and running your own copy of XFree86. The procedure for getting your X server up and running on Darwin and Mac OS X are very similar, but it is actually somewhat more difficult on Mac OS X. Where there are differences I have noted them below. Both Mac OS X Developers' Preview 4 and Mac OS X Public Beta are covered. Note that to build XFree86 for yourself on Mac OS X Beta, you will need to have the Developers' Tools. If you did not receive these as a member of ADC, they are available for download from the Apple Developer Connection.

If you want to save some time, you can use the precompiled binaries provided by the XFree86 server at `<URL:ftp://ftp.xfree86.org/pub/XFree86/4.0.2/binaries/Darwin/>`. Follow the instructions in the Install document to install it.

This will create two new directory trees, `/usr/X11R6` and `/private/etc/X11`. If you have the developers' tools, you may have to run ranlib on all the libraries in `/usr/X11R6/lib/*` before trying to build other X applications.

If you get the precompiled binaries you can skip ahead to the section on *running X windows* (section 4., page 2)

On the other hand, if you want to build things yourself from scratch, follow the directions below.

If you want to recompile the X server on Mac OS X, you will have to make the header files as described *below* (section 3.2, page 2).

3.1 Get the Code

The first thing you need to do is to get the source code from the XFree86 project.

- Setup CVS: Both Darwin and Mac OS X include CVS (Concurrent Versions System), which is an easy way to download and keep up to date with open source code released by XFree86 and others. You do have to setup CVS properly, however, to tell it where to look. Follow the instructions here on how to do this either via ssh or via the cvs pserver.
- Now make a directory in some convenient place which will serve as your top level directory for you open source development efforts. I am going to call this directory `sandbox/` in the following discussions and you could put this directory, for example, in `~/sandbox/`.
- From the command line type:

  ```
  cd sandbox
  cvs checkout xc
  ```

  Wait for all the files to complete downloading.

3.2 Make Header Files for Mac OS X

These next steps are not necessary if you are building XFree86 on Darwin. Mac OS X Beta or DP4 does not include many of the 'in-progress' header files that are needed to successfully build XFree86. The general philosophy is that at any given time, Mac OS X will include only headers files that are 'ready for prime time' while Darwin will include a snapshot of Apple's current efforts. So if you have access to a Darwin system, I would suggest using this instead of Mac OS X to build XFree86. If you don't, read the appropriate page below for an outline of what you need to do:

- *Mac OS X Public Beta* (section 6.1, page 4)
- *Mac OS X Developers' Preview 4* (section 6.2, page 5)

3.3 Build XFree86

Once you have everything ready it is easy to build and install XFree86. From the command line:

```
cd sandbox/xc
make World >& world.log

(wait for several hours)

su root
make install >& install.log
make install.man >& man.log
```

4. Run X Windows

You will want to add the X windows executables to your path. Your path is the list of directories to be searched when you type a command. You will also want to add the man pages from XFree86 to the list of pages to be searched when you are looking for documentation. The way to do this depends on the shell you are using. With tcsh or csh add the following two lines to a file in your home directory called `.cshrc` (create this file if you don't have one yet):

```
setenv MANPATH "${MANPATH}:/usr/X11R6/man:/usr/local/man"
setenv PATH "${PATH}:/usr/X11R6/bin"
```

Now type "`source ~/.cshrc;rehash`" to get these changes to take effect. On future logins, this file will be executed automatically.

You are now ready to go. On Mac OS X, you have to do a few extra steps each time you want to run X windows. You can't run XFree86 with CoreGraphics running since they both want to control the screen. So for Mac OS X you first need to logout. Then for Mac OS X DP4, from the login window type "console" as the user name and push login. For Mac OS X Beta, type ">console" as the user name. This should shutdown CoreGraphics and bring you up with a text console. Login again.

Now from the text console you can start X Windows in either Darwin or Mac OS X by typing "startx". After a brief wait at least one terminal window should pop up for you. So far it probably isn't very impressive as you might not even have a window manager running to allow you to move windows around.

When you are ready to quit X windows type "exit" in the main terminal window or quit with the window manager if you have one running. Unfortunately the X server won't shutdown correctly and you will get an apparently frozen screen with only a spinning beachball cursor on it. Nothing you type shows up on the screen, but in fact your keystrokes are being received by the console. Type "logout" to return to normalcy after a brief delay. With Darwin this should put you back at the text console login prompt. With Mac OS X, CoreGraphics will restart and you should be given a login window.

Customize X Windows

X windows is very customizable and you will certainly want to change some things. There is a lot you can do to control how your windows look, how the windows are moved, resized, etc. You will likely want to get a better window manager than twm, which is included with XFree86. The `.xinitrc` file in your home directory controls what programs are run when you start X windows. You can find a sample `.xinitrc` file in `/private/etc/X11/xinit/xinitrc`.

There are several window managers that have been ported to Darwin. The best collection of window managers and other X windows applications is at Darwinfo's ports page. Another good place to look for Darwin ports in general is at Stepwise's Softrak site.

Good luck!

5. Outstanding Issues with the X server

There is a still a lot to be done with the X server. As always please let me know if you are interested in working on any of these issues.

Things that are broken:

- The bell is broken.
- Server shutdown problem: After quitting the X server the colored beachball cursor reappears, but the screen never returns to the console display. The console is accepting input however, so typing "logout" will return you to the login prompt.
- Only one display is currently supported.
- Screen saver not supported.
- Key repeat rate can not be adjusted.

Things I am working on:

- Allowing rootless use on Mac OS X
- Correct wake from sleep behavior

6. Appendix

6.1 Installing IOKit Header files on Mac OS X Beta

Here is what you need to do to be able to build XFree86 (or other clients of IOGraphics services) on Mac OS X Public Beta.

1. You need to install some more build tools that aren't included in Mac OS X Beta, but are included in Darwin. These are needed for the next step. Go to your development directory and type:

   ```
   cvs checkout bootstrap_cmds
   cd bootstrap_cmds
   make
   ```

   Now su to root and do a ''make install'' from the bootstrap_cmds directory.

2. The IOKit framework included with Mac OS X Beta is a little on the wimpy side, even after installing the Developer Tools. You'll need to rebuild it, but to do so safely we'll take advantage of the cool directory structure of Mac OS X and put the new version in /Library/Frameworks/. This will override the version in /System/Library/Frameworks/. Or at least, that's where you should put it using the new Mac OS X Beta file structure, but in fact the compiler doesn't know this and still looks in /Local/Library/Frameworks. (I have already filed a developer's bug report on this.) To start we make a copy of the original. Login as root and do the following:

   ```
   cd /Library
   mkdir -p Frameworks
   cp -R /System/Library/Frameworks/IOKit.framework Frameworks/IOKit.framework
   cd /
   mkdir -p Local
   ln -s /Library /Local/Library
   ```

3. Now you need to get a full version of the IOKit framework from the Darwin repository. This comes as part of the kernel. So, in your development directory again:

   ```
   cvs checkout xnu
   cd xnu
   source SETUP/setup.csh
   make exporthdrs
   make install
   ```

   The "make install" will start by assembling all the headers you need, but then will proceed to build a new kernel. You can Control-C out of the build once it starts compiling things if you want. Otherwise, just be patient.

4. You also need to get the IOKitUser project, which contains the user interface to the IOKit, and merge it into the IOKit framework. From your development directory type:

   ```
   cvs checkout IOKitUser
   ```

   Then you need to change the Makefile so it builds on top of the new IOKit framework you are building. Change to the IOKitUser project directory and edit the Makefile. Change the following line:

```
NEXTSTEP_BUILD_OUTPUT_DIR = /$(USER)/build/$(NAME)
```

to:

```
NEXTSTEP_BUILD_OUTPUT_DIR = <your xnudir>/BUILD/dst/System/Library/Frameworks
```

Now, run ''make'' while in the IOKitUser directory. It will likely not finish building since the IOKitUser project relies on some stuff that Apple hasn't really made public yet, but it should finish populating the new IOKit framework with headers and start compiling. (Strangely enough, the headers installed in your new IOKit framework don't contain the real header files. They are just single line files with an #import of the file in your IOKitUser project. This works fine as long as you don't ever delete or move your IOKitUser project. I imagine that using ''make install'' would fix this up, but you can't do this as long as the project doesn't build correctly. So for now you can either move the header files by hand, or make sure you leave your IOKitUser project in place.)

5. Now put the new IOKit headers into our local copy. Change to the xnu project directory and do the following as root:

```
cd BUILD/dst/System/Library/Frameworks/
cp -R IOKit.framework/Versions/A/Headers/* \
  /Library/Frameworks/IOKit.framework/Versions/A/Headers
```

6. The System framework needs touching up a bit too. You need to copy the libkern header files from the new System.framework. As root, go to the xnu project directory and type: (Of course you could make a local copy of the System framework in /Library/Frameworks/, but the additions to the System framework are small.)

```
cd BUILD/dst/System/Library/Frameworks/System.framework/Versions/B/Headers
cp -R libkern /System/Library/Frameworks/System.framework/Headers/libkern
```

6.2 Installing IOKit Header files on Mac OS X DP4

Here is an outline of what you need to do to be able to build XFree86 (or other clients of IOGraphics services) on Mac OS X Developers' Preview 4.

1. Checkout, build, and install bootstrap_cmds from the CVS repository.
2. Checkout xnu from the CVS repository. Go to sandbox/xnu and do the following:

```
source SETUP/setup.csh
make exporthdrs
```

3. su to root and do the following:

```
mkdir -p /Local/Library/Frameworks
cp -R /System/Library/Frameworks/IOKit.framework /Local/Library/Frameworks
```

This will create a new local version of the IOKit framework that will override the System version without changing the original.

4. You now want to merge the IOKit headers from the xnu project into your local IOKit framework. I did this fairly tediously by hand, but I would suggest something like:

```
cp -R sandbox/xnu/BUILD/dst/System/Library/Frameworks/IOKit.framework/Versions/A/Headers \
/Local/Library/Frameworks/IOKit.framework/Versions/A
```

The only problem with this is that there are a few stub header files included that are just a single line with an #import to the appropriate place in the xnu source on your disk. This is fine as long as you don't move your xnu project. If you do, you can copy the original file to replace the stub. There is probably an easier way to automatically build a clean version of the IOKit framework, but I didn't find it. (Let me know if you do.)

5. There are just a few more files you need in the IOKit framework. Checkout the IO project from CVS. Get the necessary header files from there and put them in your new local IOKit framework. You'll get errors when you try to build XFree86 from which you can tell which files are needed. I also needed to make the following patch to IOLLEvent.h. Your mileage may vary.

```
47c47
< #include <Kernel/libkern/OSTypes.h>
---
> #include <libkern/OSTypes.h>
```

6. You'll also need a few extra CoreFoundation header files. Checkout CoreFoundation. You can make a new local version of the CoreFoundation framework as above, or just add to the existing one. The files you need are CFMachPort.h and CFMessagePort.h in the RunLoop.subproj and CFStorage.h in Collections.subproj.

CONTENTS

$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/Darwin.sgml,v 1.4 2000/12/12 18:54:29 dawes Exp