

MacOS X Talk/Demonstration

October 24, 2001

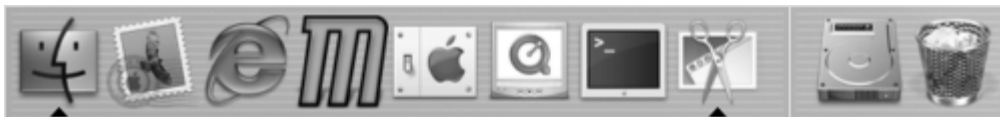
Presented by: Reeves and David Cantrell

MacOS X (pronounced "Mac Oh Ess Ten") is the name of Apple's latest version of the Macintosh operating system. Though it shares a similar name, it shares very little in common with previous releases of MacOS.

MacOS X is a UNIX-like operating system with the look and feel of a Macintosh computer. In 1996, Apple acquired NeXT, Inc. (the acquisition that brought Steve Jobs back to Apple). It was the NeXT codebase (NEXTSTEP/OPENSTEP) that Apple used as the foundation for what eventually became MacOS X.

The following outline aims to point some of the interesting aspects of MacOS X, both from a user and developer perspective.

1. **The graphical user interface:** The graphical user interface of any operating system is the part that is often evaluated in magazine reviews or is touted in discussions about which OS is better. True, the GUI is an important part of the operating system, but not the only part. The MacOS X GUI is composed of several key components.
 - a. *Dock:* The Dock was an idea brought over from NEXTSTEP and OPENSTEP. Dock-like interfaces as also quite popular among UNIX window managers, such as Window Maker and AfterStep. The MacOS X dock is virtually a complete reworking of the classic NEXTSTEP dock, but does have some things in common. To place items on the dock, a user just drags the object to the dock. To remove an item, you drag it off the dock. The dock can be positioned along any edge of the screen and also offers some special effects to make it that much more interesting. The Dock under MacOS X replaces the Apple menu from previous releases of MacOS.



- b. *Finder:* Finder is the name of the GUI "shell" that runs when you log in to MacOS X. Finder is responsible for putting the menu bar at the top of the screen and also placing icons on the desktop. Finder can be stopped and restarted without rebooting, a feature that previous releases lacked (Well, this isn't totally true. You could restart the Finder in previous MacOS releases, but it wasn't an officially supported feature and was rather flaky.). Finder is also responsible for the file manager windows that pop up when you open a folder or drive icon. The views available in the MacOS X Finder are similar to NEXTSTEP (horizontal scrolling view).

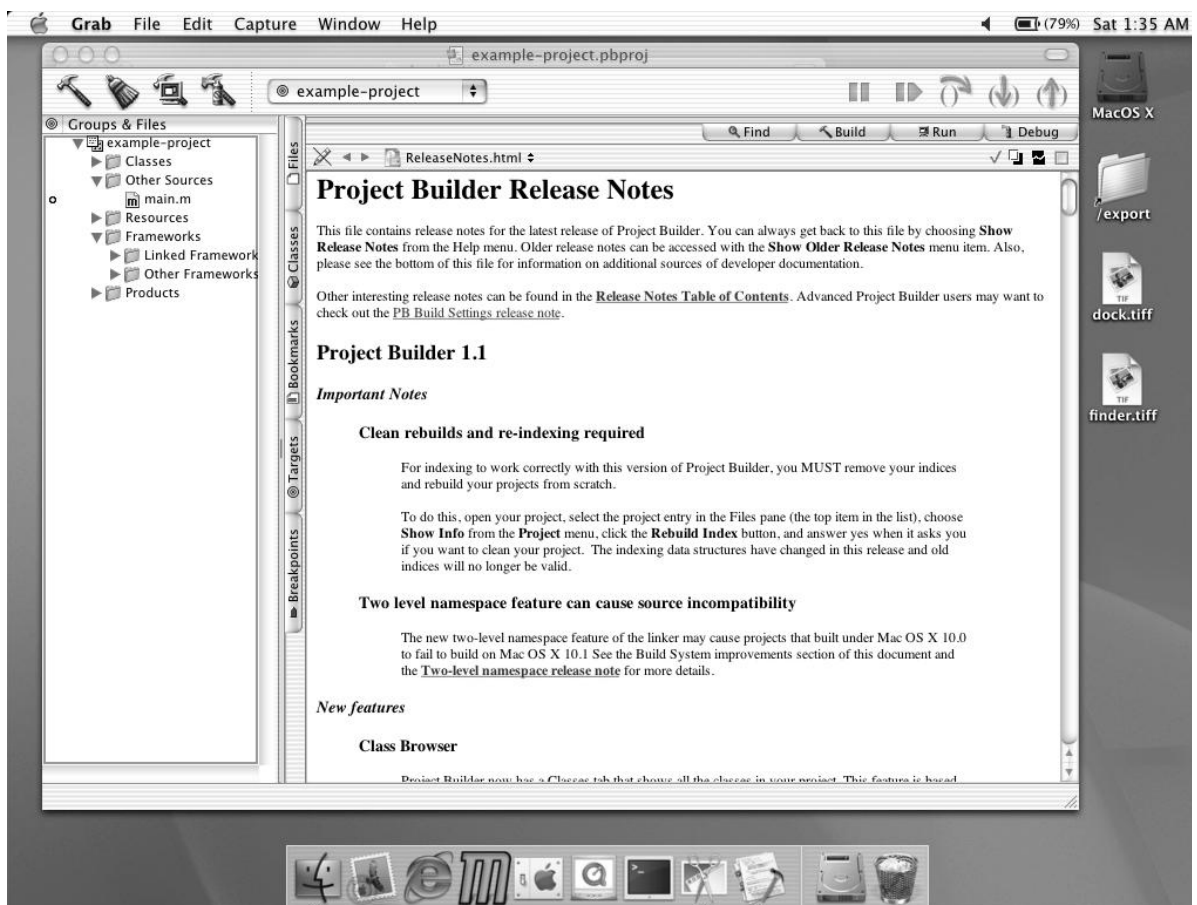


- c. *Login Window*: Since MacOS X is a UNIX-like operating system, it requires one to log in at boot time. Users of xdm and other X11-based login managers may notice a similarity to the MacOS X login screen. You can get a console login from the MacOS X graphical login window. Just type ">console" as the username and click Login.
 - d. *Technologies*: Apple loves to associate fancy names with every software project they currently have going. These names are thrown around and since they usually bear no meaning towards the actual project, it can sometimes get confusing. Here are the common ones you'll hear when discussing MacOS X:
 - i. **DisplayPDF**: The underlying concept behind graphics primitives in MacOS X. All 2D graphics are stored as PDF primitives, which expands on the NEXTSTEP usage of DisplayPostScript technology.
 - ii. **Aqua**: The name of the theme present in a default MacOS X install. This is what people try to clone for KDE and other environments.
 - iii. **Quartz**: Roughly equivalent to the X server on a UNIX system. Quartz draws stuff on the screen.
2. **The UNIX part**: The "UNIX part" of MacOS X is called Darwin. This is the part that's open source and available to anyone. It runs on both PowerPC Macintosh computers as well as Intel-based computers. Darwin consists of the kernel, userland and system tools, and networking code. Basically it's everything except the graphical user interface.
- a. *Technologies*: Several existing and new technologies came together to create Darwin.
 - i. **FreeBSD**: The TCP/IP stack as well as some of the userland tools from FreeBSD 3.2 were

incorporated into the Darwin source tree.

- ii. **NEXTSTEP/OPENSTEP:** The Mach+BSD kernel architecture from NEXTSTEP provided the foundation on which the Darwin kernel was built. Filesystem layout, packaging system, and object concepts were incorporated from NEXTSTEP.
 - iii. **GNU:** Many GNU programs were incorporated into the Darwin source tree, such as gcc, GNU make, and grep. For the most part, userland tools in Darwin and MacOS X are the same as what you would find on a Linux distribution.
 - iv. **NetBSD:** Very portable code. Whenever a FreeBSD tool couldn't easily be ported, the Darwin team went to the NetBSD source tree. This is also a trick used by many Linux distributions.
- b. *Booting:* The system boot procedure currently resembles that of NEXTSTEP. It's almost a pure BSD-style boot system, with init running /etc/rc. The change under OS X is the addition of SystemStarter. The rc scripts handle basic system init stuff, such as loading kernel modules and mounting filesystems. After the critical steps, SystemStarter takes over and brings up services, networking, and the GUI stuff. The SystemStarter works like a SysV init system. Each "service" can be found in /System/Library/StartupItems. Each service is a subdirectory containing resource files and a script that actually does the work. Apple's goal is to eventually move to SystemStarter for all boot time stuff.
- c. *Filesystem Layout:* The filesystem layout may look a bit unfamiliar if you are a Linux user, but it's not too difficult to follow.
- i. **Standard Directories:** You'll find the /bin, /dev, /sbin, and /usr directories. Those pretty much follow a standard layout. The /etc, /tmp, and /var directories are actually symlinks into directories of the same name under /private.
 - ii. **/private:** The /private directory holds files specific to the local machine (configuration files, temporary data, swap data, and logs).
 - iii. **Capital Letter Directories:** All of the directories beginning with a capital letter are fairly self explanatory. /System is for system-specific services and libraries, /Users is home directories, /Library is like /lib, developer tools and documentation is in /Developer, bundled applications are in /Applications. You can add more directories like this, but don't remove the ones created by the installer.
- d. *Things Missing:* If you're familiar with another UNIX-like operating system, you may be surprised to not find some fairly common tools.
- i. **mount/umount:** That's right, no mount or umount command. So how do filesystems get mounted? Just like under MacOS. Everything found is mounted at boot time to the location specified in the volume header.
 - ii. **NFS:** Support for NFS is lacking, and without a mount or umount command, it's even more difficult to add.
 - iii. **virtual terminals:** Missing on many commercial UNIX-like operating systems as well, but still a feature liked by many. But, with the GUI, you can run multiple terminal windows on the same screen, a la X.

- e. *Kernel*: The Darwin kernel, called **xnu**, is a combined source tree starting with NEXTSTEP and merging in FreeBSD and NetBSD stuff, and adding some new things (like the object oriented device driver layer called IOKit). It can be compiled for PowerPC or Intel machines. On MacOS X, you'll find the main kernel file in / and named mach_kernel.
3. **The Programmer part**: MacOS X offers many things specifically for the developer. There are tools for bringing classic MacOS applications to MacOS X, as well as tools for bringing UNIX applications to MacOS X. (much of the information below was gathered from <http://fink.sourceforge.net/doc/porting/>)
- a. *ProjectBuilder IDE vs. GNU development tools*: Seasoned MacOS developers will prefer the ProjectBuilder IDE that comes with the developer CD. It's a nice GUI for editing code and designing UI components. Seasoned UNIX developers will enjoy the standard set of UNIX development tools available (cc, make, bison, yacc, etc.).



- b. *Mach-O binary format*: The binary format used by MacOS X is the Mach-O format. It's not ELF and should not be confused with ELF (even though it offers roughly the same functionality). One cool feature of Mach-O is "fat" binaries, that is, having multiple architectures supported by a single binary.
- c. *Compiler and Preprocessor*: The compiler (/usr/bin/cc) is based on gcc 2.95.2 with Apple modifications. The code is actively being prepared for merging into the mainline gcc source tree, but it's difficult. Among the additions by Apple is the support for AltiVec registers. The preprocessor (cpp) is a custom Apple creation that works nothing like GNU cpp. Most

notably is the precomp problem experienced by people trying to compile open source software on MacOS X. Precomps are precompiled headers. The special cpp can read precompiled headers (binary data files consisting of all the tokens and dependency information) and regular headers. Problem is, it does not always work with regular headers. The `-no-cpp-precomp` flag is usually preferred when compiling software on MacOS X. (precomp information gathered from www.darwininfo.org)

- d. *Libtool*: GNU libtool has problems on MacOS X. There are patches for both the 1.3 and 1.4 versions to correct shared library generation. MacOS X ships with GNU libtool 1.3.5 patched for MacOS X, but it's not completely fixed. The patch can be found at <http://fink.sourceforge.net/doc/porting/libtool.php>.
- e. *Libraries*:
 - i. **.a and .dylib**: Traditional static libraries are offered through the .a files. Dynamic libraries end with the name .dylib and do not function the same as .so's on Linux.
 - ii. **dyld**: The libdl.so equivalent on MacOS X
 - iii. **dlcompat**: To help port software from UNIX to MacOS X, the dlcompat library was created to translate dlopen() calls to the appropriate dyld action.
 - iv. **Versioning and Naming**: The dynamic linker checks major and minor version numbers, unlike Linux. Naming also differs slightly. The version is part of the library name, with .dylib being at the very end of the filename. This makes it a bit easier to specify certain library versions for certain compiles.
 - v. **Modules, Libs, and Bundles**: In Linux, a shared library and loadable module for a program ("plug-in") are the same. Under MacOS X, a shared library is a .dylib file. A loadable module is a bundle ending with .bundle (but sometimes .so). Loadable modules are loaded and unloaded through dyld, which is why the dlcompat interface exists for ported software.
 - vi. **Compiler Flags**: Common symbols are not allowed in shared libraries, so you need to use `-fno-common`. Position independent code is default, so there's no need to specify a PIC flag. To build a loadable module, use the `-bundle`, `-flat_namespace` and `"-undefined suppress"` compiler flags.
- f. *Linker and Assembler*: The assembler is GNU-derived, but the linker is not GNU at all, which presents problems for GNU libtool and other open source software.
- g. *KEXTS*: Kernel extensions are similar to modules in the Linux kernel. They can be added to the kernel without recompiling it. A KEXT is a bundle, so it is a directory containing the actual loadable module and any control files or other resources used by the driver. The driver configuration file is an XML document called Info-macos.xml.
- h. *Packaging*: A MacOS X package is a special directory (Bundle) that ends with .pkg. In this directory are icons, text files displayed by the installer, scripts, and the actual package contents.
 - i. **PackageMaker & /usr/bin/package**: MacOS X currently offers two built-in methods for generating software packages. The one that receives the most attention is the PackageMaker application. It's a graphical, fill-in-the-blanks program that creates the package for you. `/usr/bin/package` is the NEXTSTEP packaging tool, updated for MacOS X. It should be noted that both programs generate

packages compatible with Installer.app, but that have different internal formats.

- ii. **Bill of Materials:** The package manifest is a binary data file called the bill of materials, or bom for short. The bom lists the contents of the package (symlinks, files, directories), the permissions and ownerships for each item, and a 32-bit checksum of the file.
 - iii. **Installer.app:** This program is responsible for adding packages to the system. It reads in the contents of a pkg bundle and walks you through the installation process. A record of the installed package is retained in the /Library/Receipts directory. NOTE: Currently, there is no way to remove, query, or upgrade packages. You can only install and make packages at this time.
 - iv. **How to distribute pkg Bundles:** Since MacOS X packages are directories, it's rather difficult to distribute them online. You can use standard tools, such as tar to make the package a single file, but the end user will have to untar the package before being able to install it. The preferred solution is to use Disk Copy to make a disk image file the size you need and drop the package in there. Disk image files are self mounting on MacOS X.
- i. *Cocoa API:* Cocoa is the Objective-C API native to MacOS X. It originated in OPENSTEP.
 - j. *Carbon API:* Carbon is the API that works on MacOS 8.6 or higher and MacOS X.

4. Other Sources of Information:

- a. <http://www.apple.com/macosx/> - Official Apple web site for MacOS X
- b. <http://www.macosx.org/> - Nice site with writeups on how to do things and reviews on applications. Somewhat wordy and difficult to navigate, but useful.
- c. <http://www.darwininfo.org/> - Independent web site devoted to information about Darwin, the open source operating system on which MacOS X is based. FAQs and HOWTO documents, plus news about Darwin developments.
- d. <http://www.stepwise.com/> - Think Wisely
- e. <http://www.macosxhints.com/> - Nice site with hints and such related to MacOS X
- f. <http://mrcla.com/XonX/> - Installing and using XFree86 4.x on Darwin and MacOS X.

Copyright © 2001 David L. Cantrell Jr., Atlanta, GA, USA

by: David L. Cantrell Jr. (david@burdell.org)

Permission to reprint this information granted provided the above copyright notice remains attached.