Mac OS X: An Introduction for Support Providers

Course Information

Purpose of Course

Mac OS X is the next-generation Macintosh operating system, utilizing a highly robust UNIX core with a brand new simplified user experience. It is the first successful attempt to provide a fully-functional graphical user experience in such an implementation without requiring the user to know or understand UNIX.

This course is designed to provide a theoretical foundation for support providers seeking to provide user support for Mac OS X. It assumes the student has performed this role for Mac OS 9, and seeks to ground the student in Mac OS X using Mac OS 9 terms and concepts.

Author: Robert Dorsett, manager, AppleCare Product Training & Readiness.

Module Length: 2 hours

Audience: Phone support, Apple Solutions Experts, Service Providers.

Prerequisites: Experience supporting Mac OS 9

Course map:

Operating Systems 101
    Mac OS 9 and Cooperative Multitasking
Mac OS X: Pre-emptive Multitasking and Protected Memory.
Mac OS X: Symmetric Multiprocessing
Components of Mac OS X
    The Layered Approach
Darwin
Core Services
Graphics Services
Application Environments
Aqua
Useful Mac OS X Jargon
    Bundles
    Frameworks
    Umbrella Frameworks
Mac OS X Installation
Initialization Options
Installation Options

Change history:

3/19/01: Removed comment about UFS volumes not being selectable by Startup Disk.
3/23/01: Minor grammatical edits.

Operating Systems 101

Apart from the many architectural and design changes "under the hood" in Mac OS X, the two major functional benefits are pre-emptive multitasking and protected memory. This section will explain what these terms mean, contrasting them with Mac OS 9.

Contents of this Section:

Learning Objectives:

At the end of this section, you should be conversant with the concepts of:

- context switching
- cooperative multitasking
- pre-emptive multitasking
- protected memory
- symmetric multiprocessing

Mac OS 9 and Cooperative Multitasking

Mac OS 9 is the latest generation of an operating system architecture that has been evolving since 1982.

When the Macintosh was first released, the operating system was designed to support a single user using a single program on a single computer.

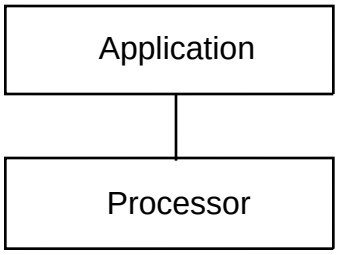
A typical computer program is a sequence of instructions stored in memory. Most instructions operate on data stored in either the microprocessor (registers) or in RAM. Instructions typically are designed to move information from one place to another, to perform logical comparisons, and to jump to subroutines based on the results of such comparisons.

As the microprocessor executes these instructions, it maintains track of where it is via a special register called a "program counter."

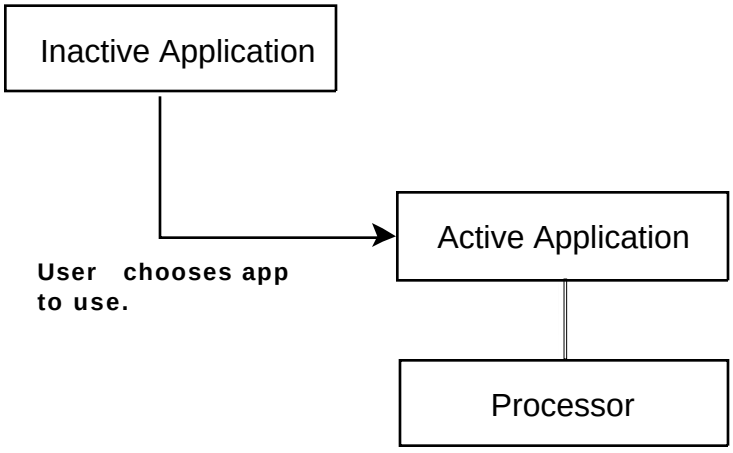
At the most basic level, when such a program is being run, it has total control over the computer. Only the code the program counter is pointing to gets executed.

In 1984, a typical program would be comprised of a "main event loop." This was as sequence of code that would run hundreds of times a second. A command

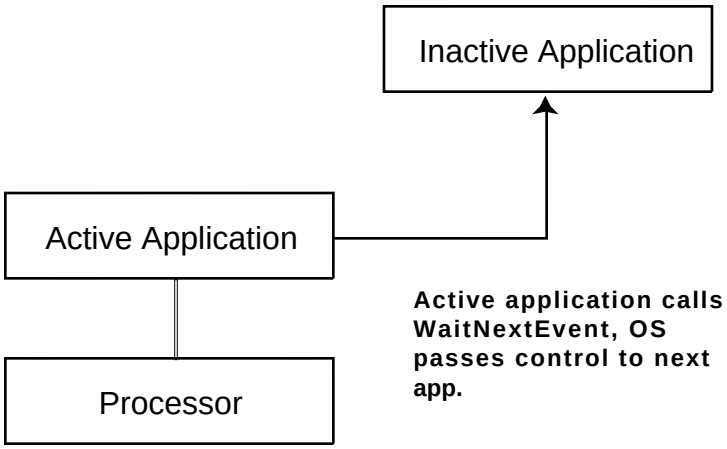
(GetNextEvent) would communicate with the rest of the operating system. This would stop the current program from executing, passing control to the operating system. The operating system would kindly save the microprocessor state (including the program counter) and restore its own state. It would then start performing a number of maintenance activities. When it was finished, it would tell the application what happened in the meantime: what keyboard, mouse, and device activity had occurred. The application would then react to this information appropriately, executing menu commands, opening dialogs, etc.



In the late 1980s, when the Mac started to acquire more and more RAM, a wonderful program came out, called Switcher. This allowed more than one user application to be resident and active in memory at one time. So one could be running MacWrite at one time, then switch over to MacPaint, edit a picture, copy it, switch back to MacWrite, and paste it in. This approach was called **context switching**. The applications weren't running simultaneously—only one would be running at once. What Switcher did was save the state of one application, restore the state of another, and switch the context the user was dealing with.



In the early 1980s, this concept was updated into an extended service called MultiFinder. This did more or less the same thing as Switcher. As MultiFinder evolved, more underlying OS support was provided, until a paradigm that supported **cooperative multitasking** was introduced.



In cooperative multitasking, the operating system was modified such that each time GetNextEvent was called, instead of just updating its internal business, the operating system would also allow other applications (or processes) in the system to update at the same time.

The key for successful implementation of this approach was that all running processes had to allow (cooperate with) the OS to perform this function. If a program stopped calling GetNextEvent, then the microprocessor would continue running that program's codes until the program either called GetNextEvent again, or quit. There was no way for the operating system to interrupt it. When this state of affairs existed, the only control the user would have would be to move the mouse around. User activity such as mouse clicks wouldn't work, since the application was behaving poorly.

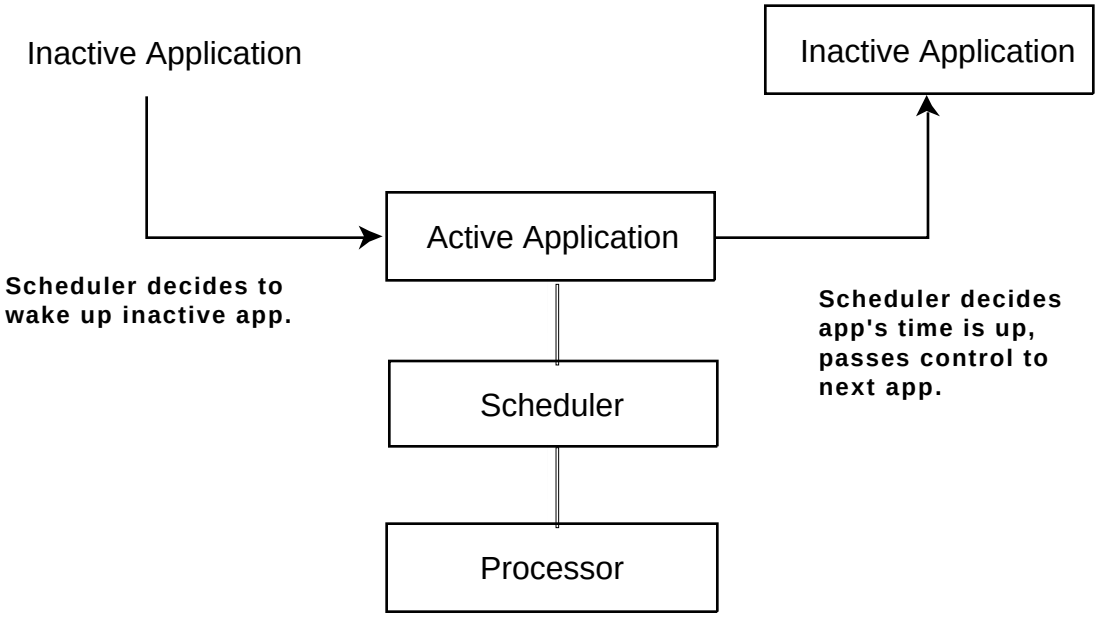
GetNextEvent was later modified and replaced with a series of successors, which allowed programs to specify a self-assessment of their importance to the OS. For example, a game would want absolute top priority, but a background spreadsheet calculation might judge itself to be less important and request a lower priority.

This is pretty much the state of affairs in Mac OS 9.

Mac OS X: Pre-emptive Multitasking and Protected Memory.

Mac OS X does away with all this. It introduces a concept called a "scheduler," which is a layer of interaction between the applications and the microprocessor. Now, an application can be written that may have *no* interface components at all—say, a pure numerical simulation that an astronomer might write, which could be very time-consuming—and leave the user with total control of the operating system and all functionality.

The scheduler is part of a component of the OS called the “kernel,” which loads before all other applications and manages the most elementary behavior of the operating system, such as process, memory, and device management. When you launch an application, the scheduler is put in control, allocating a certain percentage of CPU time to that process. When the scheduler decides that the process’s time is up, it will take control back and allows the next application to start introducing code. Thus, the scheduler can ***pre-empt*** any running process if it wishes to do so.



This introduces a level of indirection between the application and microprocessor. This indirection also allows some other good things to go on. Architecturally, for the scheduler to work, memory activity must be tightly controlled. So as an application’s code is executed, checks are made to ensure the application’s memory activity stays within bounds. If such an application tries to write data in another application’s space, or interfere with the operation of critical system resources, it won’t be successful. This is **protected memory**.

The combination of these two features lends itself to an operating system in which single rogue applications or processes can’t take the entire system hostage. If there’s a bug in an application, the application will simply crash, resulting in its memory immediately being released by the system. Everything else will run fine. Applications no longer have to respect the interests of other applications, so interaction issues can’t arise. They can be as badly behaved as their developers want, but the core operating system will remain stable and the user will remain in control.

Mac OS X: Symmetric Multiprocessing

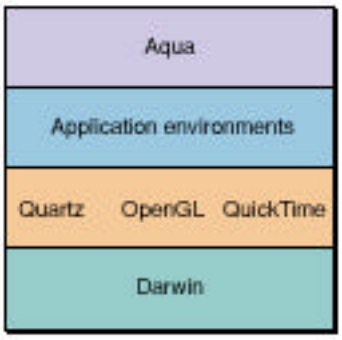
Pre-emptive multitasking and protected memory are facilitated by the UNIX foundation of Mac OS X. The behavior of schedulers and kernels in UNIX have been intensely studied over the last 20 years. UNIX is the operating system of choice in most university computer science programs, and is very well understood.

The design of UNIX kernels has kept pace with the evolution of hardware technology, including support for multiple processors. Mac OS X provides support for two microprocessors via a technology referred to as **symmetric multiprocessing (SMP)**. This is a complex scheme in which the operating system strives to keep both microprocessors (when available) busy in an efficient manner, thus multiplying system and application performance.

Unlike previous versions of the Mac OS, developers do not need to specifically code for multiple processor support, so *any* application will conceivably run much faster with two processors available. Applications which support multithreading will run even faster. This reveals yet another fallacy in the traditional "megahertz wars" between microprocessor and computer vendors: a single metric such as clock speed fails to represent actual performance improvements produced by unique architectural innovations. Increasingly, such measurements fail to represent actual system performance, even within processor families.

Components of Mac OS X

Mac OS X encompasses many different technologies. This section outlines the major ones: Darwin, Quartz, OpenGL, QuickTime , Application Services, the Application Environments, and Aqua.



Contents of this Section:

The layered approach
Darwin
Core Services
Graphics Services
Application Environments
Aqua

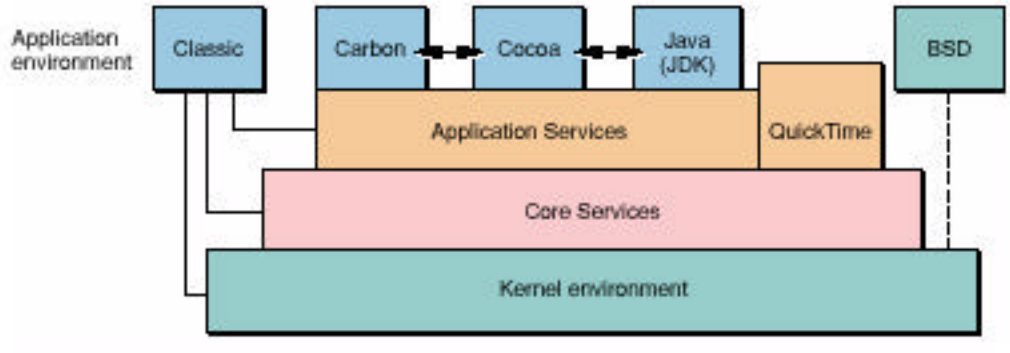
Learning Objectives:

At the end of this section, you should be conversant with the concepts of:

- Darwin
- Mach
- BSD
- Core Services
- Open Source
- QuickTime
- OpenGL
- Quartz
- Application Environments
- Carbon
- Java
- Cocoa
- Classic

The Layered Approach

One way of looking at operating system design is the “layered” approach.



In this approach, the services on each successively higher layer are directly dependent on services provided by the layer beneath them. Thus, Darwin is the bottom (core) layer, encompassing the kernel, the heart of the operating system. Core Services sits on top of that, and so on. An important distinction is that most higher-level services can’t (or shouldn’t) make a call that penetrates the layer underneath them. For instance, an application sitting at the top layer generally won’t penetrate down to Darwin. Such an application would call services in the module underneath it, that one would repeat the process, and so on. The major exception is Classic, which, due to the needs of supporting the Mac OS 9 environment, interacts directly with every level of the OS.

This approach allows for the creation of very complex operating environments which are nonetheless very robust. As long as each layer is thoroughly tested, then each successive layer does not need to worry about the behavior and design of services underneath it. This allows for the creation of abstract, transportable application environments at the topmost levels, and spares the typical software developer from having to have a deep understanding of the underlying operating system. This, in turn, will tend to accelerate application development and application quality.

> **Tip:** You can probably exclude a problem with a lower-level layer (thus obviating the need for reinstallation of system software) if a problem is restricted to a single application using a type of service. In general, if a lower layer breaks, there will be a cascading effect to everything that sits on top of it.

Darwin

Darwin is the heart of Mac OS X. It is comprised of two major technologies: Mach and BSD.

Mach is a UNIX technology developed at Carnegie-Mellon University in the late 1980s and 1990s. It's a robust, "open source" operating system, which is used to provide memory and task implementation services. Darwin uses Mach 3.0. Mach provides memory management, memory protection, process scheduling and interprocess communication services.

BSD is the Berkeley Standard Distribution. Back in the early 1980s, the University of California at Berkeley was a major center of focus of UNIX activity. They adapted AT&T's System III and provided more sophisticated process management and network functionality. They distributed their version of UNIX, called the Berkeley Standard Distribution, for what amounted to the cost of goods. This, in turn, was widely adapted by academia and turned into the source material for further innovations by companies such as Sun Microsystems. Darwin incorporates elements of the BSD 4.4 distribution, which provide file system support, network services, symmetric multiprocessing support, and multi-threading facilities.

BSD also provides support for the "**shell**" environment. Simply put, a shell is an interactive, command-line interface that allows the user to control the operating system and accomplish tasks. UNIX is often confused with its shell interface. However, the best way to view the shell is as one of many possible ways to manipulate the core services. For three decades, the easiest way to do this was via command-line interfaces, such as the Bourne shell (sh), the C-shell (csh), the Korne shell (ksh), tcsh, bash, and many others. However, starting in the 1980s, manufacturers started to experiment with many different ways to provide similar control via graphical interfaces, such as SunView, Motif, X-Windows, and others. None of these graphical approaches has reached the level of integration that Mac OS X provides.

The shell environment is completely optional for most users. It provides maximum control over the operating system. It will appeal mainly to advanced system administrators, computer science students and experienced UNIX users. The only way to get to the shell in Mac OS X is for the user to locate the Terminal application and open it (`/Applications/Utilities`).

Darwin is an **Open Source** environment. Beyond the scope of this training, suffice it to say that Open Source allows developers outside of Apple to modify fundamental components of the system software at the source code level. It is expected that this will drive innovations in the evolution of Mac OS X. More information can be found at http://www.apple.com/opensource.

Core Services

The **Core services** layer provides a number of higher-level utilities of interest to applications, including a number of services analogous to the Operating System Utilities from Mac OS 9. Carbon interfaces (see below) are provided for these utilities, which include services such as the Resource Manager, Time Manager, Alias Manager, Apple Event support, etc.

Core Services is associated with the CoreServices framework (CoreServices.framework). A framework is analogous to a shared library. More on that later.

Core Utilities is another framework of Core Services. This provides a number of services that help facilitate inter-application communication as well as other services.

Open Transport also sits in the Core Services layer.

Graphics Services

Mac OS X supports a rich variety of graphics services.

Quartz is the modern heart of the user experience, superseding QuickDraw and related managers, and directly supporting the Aqua user experience (see below). The emphasis on Quartz is supporting crisp two-dimensional graphics to support the user interface, including on-the-fly rendering and anti-aliasing. Quartz is based on the Adobe Portable Document Format (PDF). PDF is the standard for electronic document distribution worldwide. PDF is a universal file format that preserves all of the fonts, formatting, colors and graphics of any source document — regardless of the application and platform used to create it.

OpenGL is a sophisticated open-source rendering system. Developed as an open source project (OPEN Graphics Library) by Silicon Graphics in the 1990s, OpenGL supports advanced 3D rendering. It is of primary interest to game and multimedia developers. It also has a role in scientific visualization. OpenGL replaces QuickDraw 3D, which is no longer supported.

The third element of graphics suppport, QuickTime, is the same old QuickTime 5 we know and love, with no change in functionality.

Application Environments

One of the key challenges in developing Mac OS X was the need to support older application packages while simultaneously providing a modern platform that new applications could support.

The **Classic** environment supports launching Mac OS 9.1 as a process. Once this process opens, applications can be run. It is important to understand that Classic provides the environment to launch Mac OS 9.1—it isn't directly simulating 9.1. It utilizes a complete, intact, bootable Mac OS 9.1 System Folder located on one of the partitions. The Classic environment supports older, non-updated applications, including 68K applications. Since launching the Classic environment also involves "booting" Mac OS 9.1, user extensions are also available to applications running within the Classic environment.

The **Carbon** environment is a set of Application Programmer Interfaces consistent with Mac OS X's design objectives. Carbon includes versions of about 70% of the APIs present in Mac OS 9.1. Carbon applications can run in Mac OS 9.1 (with CarbonLib present) or in Mac OS X. They will automatically take advantage of all the interface features of Aqua. When you launch a Carbon application, it's indistinguishable from any other Aqua application.

The **Cocoa** environment is an object-oriented framework that provides development tools (Interface Builder and Project Builder, which can also build Carbon and Java apps) and services that allow such applications to interact with one another and take advantage of a variety of advanced libraries. Cocoa applications look the same as Carbon applications to end users.

The **Java** environment supports 100% Java 2 applications and applets. Java may be selected as a development language by the Cocoa developer tools. One of the advantages of Java development is that applications developed in Java are code-compatible with other computers running the same version of the Java Virtual Machine.

Aqua

Aqua is where it all comes together, a new look and feel for the Macintosh user experience. It makes extensive use of animation and rendering to provide a stimulating and pleasing work environment.

The most significant new interface features of Aqua include:

- The look & feel.

- The Dock, which is a user-customizable toolbar-like repository for applications and documents. In this respect, it replaces the Launcher from Mac OS 9 as well as the traditional uses for the Apple menu.

- The extensive and natural support for multiple users.

Control panels are now known as System Preferences, and can be accessed via the Dock or via the Utilities folder.

Training for Aqua is beyond the scope of this document. Please refer to the actual software or to http://www.apple.com/macosx.

Useful Mac OS X Jargon

Contents of this Section:

Bundles
Frameworks
Umbrella Frameworks

Learning Objectives:

At the end of this section, you should be conversant with the concepts of:

- Bundles
- Frameworks
- Umbrella Frameworks

Bundles

Bundles are a new way of packaging applications. In Mac OS 9, an application often resided in several directories, such as shared libraries in the Extensions folder, and the executable application in some arbitrary folder. In addition, an application itself was comprised of two "forks" within a single file: a data fork (which in the case of PowerPC code contained the program's executable code), and a resource fork. The resource fork contained templates for various elements of the user experience, such as the names of menus, menu items, dialogs, window types, the icon the Finder should display, etc.

In Mac OS X, this has changed. A bundle is a collection of these various data elements (executable code, resource information), each of which is a single file. These files are gathered together in a directory and then presented to the user as a single, double-clickable file. In Mac OS X, applications are bundles.

> Tip: Icons have an .icns suffix. You can edit them with Icon Composer or Pixie, two utilities which are distributed on the Developer CD in `/Developer/Applications`.
>
> Tip: A bundle may be opened, such that you can see the individual components of the directory, by option-clicking on its icon.
>
> Tip: Bundles are sometimes called "packages."

Frameworks

A **framework** is a dynamic (loaded only when used) shared library along with its associated resource information. This information is packaged as a bundle. This is analogous to the shared libraries which are placed in the Extensions folder in Mac OS 9. Shared libraries contain code that can be shared by multiple applications. For instance, if a manufacturer produces a suite of productivity software, odds are there's going to be a great deal of similarity within their code. Therefore, from a development and deployment perspective, it makes sense to allow the applications to share common code.

Umbrella Frameworks

Umbrella frameworks are generally system software-level functions. This concept refers to a framework which itself contains two or more subframeworks. Generally this shouldn't come up in support.

Mac OS X Installation

Mac OS X is easy to install, taking a comparable amount of time as Mac OS 9.1. There are a number of concepts which must be understood with respect to installation options as well as to the organization of data on the hard drive.

Contents of this Section:

Learning Objectives:

At the end of this section, you should be able to:

- Explain the differences between UFS and HFS.
- Identify the three package types that can be installed.
- Identify the three new startup key sequences.
- Explain to users why Setup Assistant is a Good Thing.
- Explain what it takes to run Mac OS 9.1 in Classic.
- Explain the limitations of Classic mode.
- Explain the use behind the standard directory names.
- Differentiate between system-level and user-level directories.
- Be able to locate the Mac OS X equivalents of Mac OS 9 directories.
- Be able to explain why there are invisible folders and files in Aqua.
- Be able to read a path name and understand what it refers to.

Initialization Options

A hard drive may be partitioned into several volumes by using Drive Setup, which is now a component of Disk Utility. One of the reasons for doing this is for robustness (e.g., a developer who might want to limit directory damage caused by debugging buggy software), organization, or performance (e.g., users doing video editing may frequently wish to "de-fragment" their drives by erasing them). Partitioning is a solution which is supported by Apple.

On installation of Mac OS X, the user has the option of selecting two file formats: UFS (Unix File System) and Mac OS X Extended (HFS+). Most users will want to use Mac OS X Extended. A partition formatted with UFS will not mount in a Mac OS 9 startup

environment. Nor will it be available as a shared disk via Network File System (a way of sharing data in a workgroup environment).

> Note: Mac OS X must be installed on either a Mac OS X Extended or a UFS volume. The core system cannot be installed on Mac OS X Standard (HFS) volumes, though it can read data on such volumes.
>
> Tip: It is not necessary to erase a volume if it's already formatted as Mac OS X Extended.
>
> Tip: Installing Mac OS X over it will not delete any data.
>
> Caution: Mac OS X has to go on a partition that is within the first 8 GB of the device.

Installation Options

Users have three installation options:

The Base System and Essential System Software packages will always be installed.

The user may opt not to install the BSD subsystem. This is not recommended, as it may break certain types of network services. In particular, parts of the Network Utility, FTP, NFS, and Telnet services will break.

> Tip: If a user reports problems using these services, make sure that BSD wasn't turned off in an attempt to save disk space.

The user may opt not to install the many print drivers.

Startup Keys

At system start time, there are three startup key sequences which will be handy:

Holding down the **option** key will produce a volume list, allowing the user to select which system to start from. This is a hardware-based feature, and may not be available on all systems.

Holding down the **command and v** keys will put the system into "verbose" mode, such that you can watch the status of the system as all of the drivers and packages load.

Holding down the **command and s** keys will put the system into "single-user" mode, and put you right into a shell. This should only be used in extreme troubleshooting situations and only if you're comfortable with shell-level system administration.

Mac OS X Setup Assistant

After installation, the user will be required to go through the Mac OS X Setup Assistant. This will register the user with Apple, as well as configure the initial Internet preferences, an iTools account, and mail services.

> **Important**: Many users will have a natural tendency to open Classic (see below) in order to configure their preferences, simply because they are accustomed to this environment. Several control panels will not be available, due to potential conflict with Mac OS X's functionality. Others may appear to have confusing configuration options. In particular, network services will be overridden by Mac OS X. For example, Mac OS X's Network System Preferences might be set up to use DHCP to gain its IP address, but Classic will show a static address manually assigned. *When troubleshooting issues such as connectivity, it's important to ask the user whether any attempt was made to modify system-level settings from Classic.* All of this is a good reason to extoll the virtues of properly completing the Setup Assistant with your customers—particularly old-time Mac users who are accustomed to setting up their environments "manually," via direct manipulation of control panels.

Mac OS 9 and Classic

Classic is an environment which supports the launching of Mac OS 9.1 as a process. In order to support this feature, Mac OS 9.1 must be installed on the hard drive. The Mac OS X installer does not install Mac OS 9.1. Mac OS 9.1 is, however, bundled as a separate installation CD in the retail distribution.

> **Tip**: Mac OS 9.1 can be installed on the same volume as Mac OS X, or on another volume. You can also install Mac OS 9.1 on the same volume as Mac OS X before or after Mac OS X is installed.

The process of opening the Classic environment is initiated by either using the Classic System Preference, or by opening a Classic application.

Applications running under Classic do not enjoy the benefits of memory protection within the Classic environment. So they can still damage the running Classic environment or other applications if they fail. They cannot damage processes outside of the Classic environment.

Since Classic itself is a process in Mac OS X, Classic applications do enjoy the benefits of pre-emptive multitasking, i.e., if they do hang or crash, the user can still control the system and other non-Classic applications. However, in this case other currently-running Classic applications will prove to be problematic. When this happens, the user can force-quit out of Classic by using the Classic System Preferences.

Keep in mind that since Classic is a hosted environment, some functions may not be available. So if control panels fail to open or options are dimmed, that's why.

If the user should discover that there's an important function that can only be accomplished by launching Mac OS 9.1 (as opposed to Classic), the Startup Disk System Preference can be used to select the Mac OS 9.1 System Folder. If the user then restarts, the system will use Mac OS 9.1. The user can return to Mac OS X by using the Startup Disk Control Panel.

Standard Directory Names

In Mac OS 9, standard directories included Apple Extras, Applications, and the System Folder. Mac OS 9.1 introduced minor variations, such as "Applications (Mac OS 9). The organization of Mac OS X is quite a bit different.

There are five important directory names the user will encounter:

Applications, the folder which contains application software the user can use to be productive. The Applications directory will typically contain a Utilities directory, which contains applications that can be used to monitor or administer system behavior.

System, which contains critical Mac OS X system software. Typically the user will not add or remove items from anything in this folder: it's designed to remain pristine.

Library, where preference information is located. In the UNIX world, though, Libraries tend to contain much more than the sort of information contained in typical Mac OS 9 preferences files. In many cases, Library files contain extensive instructions on how applications should behave. In Mac OS X's case, Libraries are where fonts, desktop pictures, modem scripts, printer drivers and many types of plug-ins also reside.

Users, where user information resides.

Documents, where user documents automatically stored.

Now it gets a bit weird.

This scheme is replicated throughout the system. So at the top level of the hard drive, you'll see folders (Applications, System, Library and Users) which are available to all users on the system.

System and Users are owned by the system, and you'll need "root" access (covered later) to manage them. Typically a user will *not* want to muck with them directly. Things that need to be adjusted (such as creating or deleting users) can be adjusted using the appropriate utilities.

But take a look inside System: you'll see a Library file. This is consistent with the naming philosophy described above—these are preferences, plug-ins, etc., that the system needs to support itself.

Back to the top level: Applications and Library contain data of appeal and interest to all users of the system. Any Administrator-level user or above can delete or add material to these folders.

Continuing this walk-through, go into the Users folder. Note that the active logged in user has the house icon. Go into one of the User folders.

You'll find the same structure as at the top level of the hard disk, minus the System folder. This is to support the notion that users may have their own *private* data they need to have installed. Typically, when a user installs application software, it will be located in the user's *private* Applications folder, Library, etc., unless a conscious decision is made to share it with all other users.

> Tip: It's a good idea to evangelize this type of thinking with users. Many people are accustomed to thinking in terms of single-user/single-system, but as their skill levels evolve, they may wish to share their computer with other people on the network, set up FTP access, etc. Users aren't losing anything by taking this approach. Say, for instance, that a font's installed in a personal Library directory—that font will still be available to applications installed in the top-level Applications folder.
>
> Tip: Note that items on the desktop are maintained in the Desktop folder in the user's home directory.
>
> Note: The default folders within a user's directory are, by default, private, and accessible only to that user. The main exception being the "Public" and "Sites" folders.

Quick Answers: Where do my _____ go?

In the following, `myhome` refers to a user's home directory. E.g., if a user's login name is "Mike", `/Users/myhome/Library` would be `/Users/Mike/Library`.

Item	Mac OS 9	Mac OS X
Fonts	System Folder: Fonts	`/Users/myhome/Library/Fonts` `/Library/Fonts`

Apps	Applications	`/Users/myhome/Applications`
Documents	Documents	`/Users/myhome/Documents` `/Library/Documents`
Desktop Pictures	Anywhere	`/Library/Desktop Pictures`
Control Panels	System Folder: Control Panels	N/A
Extensions	System Folder: Extensions	N/A
ColorSync Profiles	System Folder: ColorSync Profiles	`/Library/ColorSync/Profiles` `/Users/myhome/Library/ColorSync/Profiles` (user-created)
General Preferences	System Folder: Preferences	`/Library/Preferences` `/Users/myhome/Library/Preferences`
Keychains	System Folder: Preferences: Keychains	`/Users/myhome/Library/Keychains`
Modem Scripts	System Folder: Extensions: Modem Scripts	`/Library/Modem Scripts`
Sounds	System Folder: System	`/Library/Audio/Sounds` `/Users/myhome/Audio/Sounds`
Startup Items	System Folder: Startup Items	?
Printer drivers	System Folder: Extensions	`/Library/Printers` `/Users/myhome/Library/Printers`

More Directory Names

But all this is just the tip of the iceberg. Since there's a UNIX environment running under the hood, there are actually quite a few more directories to support standard shell-based tools. These directories are all *invisible* to users operating in the Aqua environment.

If you open up the Terminal application (go to the root-level Applications folder, then to Utilities), you'll find that you can access Darwin's command-line interpreter. The default environment is called "tcsh," a popular variation of the C-Shell. Using the `ls /` command, you'll see these invisible folders and files. These files are there to support a standard shell environment as well as critical system resources.

Discussion of these is beyond the scope of this document, but any good UNIX reference book should be more than sufficient to explain this organization. Suffice it to say that for typical support purposes, a Mac OS X support provider won't have to deal with this level of the operating system. We mainly raise it here since many utilities will reveal the presence of these files.

> CAUTION: These invisible files are supposed to be there. Don't delete them to save disk space. Bad things could happen.

A Word on Paths

The directory is structured as a "tree." The normal way of referring to paths is to start at the top level (/) and then separate folders, working down, by slash (/) signs.

So / is the root directory, the top of the hard drive.

`/Applications` refers to the Applications folder.

`/Applications/Utilities` refers to the Utilities folder, which is located inside the Applications folder.

`/Applications/Utilities/Terminal` refers to the Terminal application in the Utilities folder.

> Note that almost every file and application on the system actually has a suffix indicating what it is. E.g., "terminal.app" to describe the Terminal application. This suffix is hidden from the user, but you will be able to see it in a terminal environment. So you may also see paths written in the form of `/Applications/Utilities/Terminal.app`. Don't let this throw you.

Security

UNIX is a fairly secure operating system, but it's only as secure as how well it's maintained. This section discusses some of the philosophical issues behind UNIX system security and security in general.

Contents of this Section:

Learning Objectives:

At the end of this section, you should be able to:

- Discuss UNIX security in general.
- Discuss the three types of file permissions.
- Discuss the three ways file access can occur.
- Enable or disable root access.

UNIX and Security

UNIX was originally developed to be an *insecure* system. Original builds in the 1970s had no login process, and the rudimentary networking capabilities at the time were based on the "good neighbor" principle. That mode of thinking quickly ran into the real world, so the following ten years saw the introduction of increasing layers of security, including encrypted passwords, login dialogs, file system security, and so forth.

Once UNIX broke out of the academic world, its main susceptibility to attack came in three flavors:

1. "Cultural" attacks. I.e., if someone trying to access a system knew something about the user base, intelligent guesses could be made as to passwords. Such attacks could come from insiders (the user lists traditionally stored in `/etc/passwd` were often world-readable, i.e., everyone on the system could read them) or from external people. This is still a concern with any "closed" system.

2. Misinstalled software. Some software packages were installed improperly (e.g., in one case if a word processor was installed incorrectly by a "root" user, a non-root user could drop into a shell-command mode and execute commands as root). Other software packages were distributed with "debugging" modes turned on, which allowed attackers access to the system. This led to a fad of "worm" attacks, starting with the Morris worm. The UNIX community grew up quickly at this point, and most implementations are now fairly secure. Virus attacks have not generally been

associated with the UNIX world, but note that application environments such as Classic are still vulnerable.

3. Denial of service attacks. These can range from flooding a system with network requests to a user grinding the processor to a halt through creative shell scripts. Depending on the environment, these can still be issues.

Culturally, UNIX systems tended to be locked away in secure machine rooms, so the thinking on security had it that any attack had to come in through a network or terminal. However, with the advent of inexpensive UNIX implementations that could run on personal computers, *physical* security has become an issue. In the case of Mac OS X, for instance, if a user boots up with a Mac OS 9.1 system CD, he'll have access to data in the system. Some manufacturers have striven to produce "secure UNIX" environments with encrypted file systems and the like.

Bottom line: the only truly secure computer is one that is locked in a secure room and not connected to a computer network. We raise these points here mainly to discourage enthusiasts from overstating the robustness of the security environment. There are limits to the security of *any* operating system, but this is not to say that Mac OS X is limited compared to other UNIX implementations or NT.

Multiple User Implementation

Mac OS X supports multiple users. As described above, each user has his own work environment, his own applications. UNIX-style access security applies.

> Tip: When a user chooses to restart Mac OS X after logging in, a login prompt will not be provided by default. This behavior can be changed in the Login System Preferences.

While an in-depth discussion of such security is beyond the scope of this training, we'll highlight the major concepts.

Any file has three types of access permissions: read, write, and execute. These can be intermingled.

If a write bit is set, then users can modify a file or create a file in a directory.

If the read bit is set, then users can read a file or list a directory.

If the execute bit is set, then the file can be opened as an application and the operating system will try to run it.

There are three permissions groupings:

User, which applies to the owner of the file.

Group, which applies to the group the file belongs to. A file may only belong to one group.

Other, which applies to anyone on the system.

These can be intermixed. For instance, by default you can do whatever you want with your file (read and write bits set; execute set if the file is an application).

Groups are collection of users with a common attribute. A file can belong to a group. This assignment is accomplished (via UNIX) by the chgrp command, and in Aqua via the Users System Preference dialog. The permissions (read, write, execute) you assign to this attribute applies to members of the assigned group.

The "Other" category is a universal setting, setting privileges for all users on the system. The permissions apply to anyone on the system.

In terms of hierarchy:

The OWNER of the file can set any file permissions.
Once permissions are set, the owner can use the file.
Owner permissions override all other permissions.
Group permissions override the "Other" permissions.

Any good UNIX reference covers these concepts in detail. We recommend that support providers learn a bit more about the UNIX implementation of permissions; it will help you to understand what the user interface is doing "under the hood."

Root

Root is the ultimate superuser. Remember, the first user is made "administrator" by default. This gives them access to everything on the system except for the System folder. The next step up is to give them access to the System folder. This can be very dangerous, inasmuch as files deleted or modified can affect the system at the most fundamental level. In general, root access should only be used by people very comfortable with UNIX at the system administration level.

You can enable root by using the NetInfo Manager and using the options in the Security hierarchical menu under the Domain menu.

Old Stuff in New Terms

Almost everthing that was present in Mac OS 9 has been carried over to Mac OS X. The items and concepts left behind are either obsolete or have been integrated into the new user experience.

Contents of this Section:

Learning Objectives:

At the end of this section, you should be able to:

- Explain why leaving extensions and control panels with INIT code behind was a good idea.
- Identify the four types of supported fonts.
- Recognize that FKEYs are no longer supported.
- Identify changes in the print architecture, especially that Print Center is at the heart of the new user experience
- Understand that desktop printers no longer exist.
- Understand the functionality of Disk Utility.
- Understand how to set Startup Items.
- Describe how functionality has been deployed in the new interface.
- Leave with a shopping list of items to explore on your own.

INITs in Mac OS X

INITs were a way of having code load and execute at boot time in Systems 4-7 and Mac OS 8-9. They got their name from the fact that they were saved as short code segments in an INIT resource within a file.

A couple of years after the original Macintosh computer was released, third-party developers realized they could patch system-level commands. The original Macintosh shipped with a huge library of services that third-party developers could use. These were referenced via an Application Programmer Interface (API), and were referred to by name. When a developer wished to use one of these commands, he'd simply insert it in the code of the program. When the program was being executed, the microprocessor would stop when it ran into one of these commands and went to an area of memory called a "trap

dispatch table." Based on the formatting of the command, the microprocessor would go to an area within the table and resume execution at an address pointed to by that section of memory.

So some developers figured, why not save the address in the trap dispatch table and put an address to THEIR code instead. Then, when their code finished executing, the third-party code would call the original address in the trap dispatch table.

By choosing the right command (e.g., GetNextEvent, discussed above), a developer could thus ensure that his code was called several times a second. This led to the advent of a huge number of utilities that patched the system, ranging from screen savers to menubar utilities.

The problem, though, is that they WERE patching fundamental aspects of the system—in some cases, there would be dozens of third-party processes hanging off popular traps, and in at least a few cases, a single third-party package might patch dozens of system commands. So a situation arose where INITs of increasing complexity (and appeal to users) were being developed. The problem was that if any of them failed, due to coding errors, the exploitation of undocumented aspects of the operating systems, etc—the whole machine could crash.

This led to the backbone of Mac stability troubleshooting: get the machine back to known configuration (factory default), re-introduce extensions slowly, test for stability, and try to identify what might be causing the problem.

The term "INIT" fell by the wayside in the late 1980s, to be replaced by Control Panels and Extensions. Control Panels and Extensions might or might not have INIT code in them.

While Mac OS X does not support this concept per se, it is an Open Systems architecture, so third-party developers may well figure out a way to patch the kernel. In addition, the kernel itself supports "kernel extensions," but these are designed to support the very lowest levels of the system. They aren't designed to support higher-level processes that would provide access to Core Services or the graphics services.

Fonts

Mac OS X supports TrueType, Type 1, UniCode, and legacy bitmap fonts.

> Tip: Mac OS X will not automatically copy fonts from a pre-existing Mac OS 9 System Folder and make them available to users in Aqua. The user will need to manually copy fonts from the Fonts folder inside the Mac OS 9 System Folder and put them in `/Users/myhome/Library/Fonts`.
>
> Tip: Applications running in Classic will only have access to the Fonts installed in the Classic Mac OS 9 System Folder.

FKEYs

In Mac OS 9, FKEYs were supported that allowed the user to do a screen capture at any point via command-shift-3. This function is no longer available in Mac OS X. Instead, users should use the Grab application, which is located in `/Applications/Utilities`. Once a screen image capture is performed, the document will be opened in the Preview application, from whence the image can be saved and/or printed.

Printing from Mac OS X

Mac OS X has an all-new print architecture. The heart of the user's printing experience is the Print Center. This allows the user to select the current printer and to monitor and manage the progress of print jobs.

PDF is the new standard document format. Applications can save documents to a device-independent PDF format when appropriate.

The device driver architecture also supports a much easier driver development scheme, utilizing a framework called the I/O Kit. This will result in new print drivers being available more quickly for Macintosh users, as well as a more robust printing environment. Drivers for a large number of common printers are pre-installed with Mac OS X (see `/library/printers`).

Mac OS X does not support desktop printers.

Disk First Aid and Drive Setup

Disk First Aid and Drive Setup have been replaced by Disk Utility (`/Applications/Utilities/`), which provides a clean new interface offering all the functionality of its predecessors.

Startup Items

In Mac OS 9, users could select items to start up as the Finder loaded by putting aliases or files into the Startup Items folder.

In Mac OS X, the user may select items to open automatically at startup by going to the Login System Preferences dialog.

Startup Items are located at
`/Users/myhome/Library/Preferences/loginwindow.plist`

This is *not* a user-editable list. Use the Login System Preferences dialog instead.

Mac OS 9 Control Panels and Functionality mapped to Mac OS X

This section correlates features provided by Mac OS 9 to the equivalent functionality in Mac OS X.Note that in Mac OS X, a number of items (such as the need for the Memory control panel) have been rendered obsolete, while the functionality of other items (such as the Launcher) are integrated into the new user interface.

Function	**Mac OS 9**	**Mac OS X**
Desktop pictures	Appearance control panel	Finder: Finder Preferences
Themes	Appearance control panel	N/A
Appearance	Appearance control panel	General System Preferences
Font Preferences	Appearance control panel	N/A
Desktop Patterns	Appearance control panel	N/A
Sound Effects	Appearance control panel	N/A
Collapsible Windows	Appearance control panel	N/A
Scroll Arrow Placement	Appearance control panel	N/A
Network services	AppleTalk control panel, TCP/IP control panel	Network System Preferences
Apple Menu configuration	Apple Menu Options control panel	N/A
ColorSync	ColorSync control panel	ColorSync System Preferences
Control Strip	Control Strip control panel	replaced by Dock
Date & Time	Date & Time control panel	Date & Time System Preferences
Energy Saver	Energy Saver control panel	Energy Saver System Preferences
Extensions Manager	Extensions Manager control panel	N/A
File Exchange	File Exchange control panel	N/A
File Sharing	File Sharing control panel	Sharing System Preferences
File Synchronization	File Synchronization control panel	N/A
General Controls (Launcher, Menu Blinking, Insertion Point Blinking, Document folder preferences, “check disk” function.	General Controls control panel	N/A
Infrared	Infrared control panel	N/A
Global Internet	Internet control panel	Internet System Preferences

application preferences		
Keyboard settings	Keyboard control panel	Keyboard System Preferences
Keychain access	Keychain Access control panel	`/Applications/Utilities/Keychain Access`
Launcher	Launcher control panel	replaced by Dock
Location Manager	Location Manager control panel	Network System Preferences, Apple menu
Memory control	Memory control panel	N/A
Modem	Modem control panel	Network System Preferences
Monitor control	Monitors control panel	Displays System Preferences
Mouse control	Mouse control panel	Mouse System Preferences
Multiple Users	Multiple Users control panel	Users System Preferences
Numbers	Numbers control panel	International System Preferences
QuickTime settings	QuickTime settings control panel	QuickTime System Preferences
Remote Access	Remote Access control panel	Network System Preferences, `/Applications/Internet Connect`
Software Updates	Software Update control panel	Software Update System Preferences
Sound control	Sound control panel	Sound System Preferences
Speech control	Speech control panel	Speech System Preferences
Startup Disk control	Startup Disk control panel	Startup Disk System Preferences
TCP/IP control	TCP/IP control panel	Network System Preferences
Text behavior	Text control panel	International System Preferences
Trackpad control	Trackpad control panel	Mouse System Preferences
Web sharing	Web Sharing control panel	Sharing System Preferences
Chooser AppleTalk selection	Chooser	Network System Preferences
Printer selection	Chooser	Print Center
Connect to IP server	Chooser	Finder, Go menu.
Select AppleShare server	Chooser	Finder, Go menu.
Key Caps	Apple menu	`/Applications/Utilities`
Calculator	Apple menu	`/Applications`
Favorites	Apple menu	Finder, Go menu.
Network Browser	Apple menu	N/A
Scrapbook	Apple menu	N/A
Note Pad	Apple menu	N/A
Stickies	Apple menu	`/Applications`
Erase Disk	Special menu in Finder	Disk Utility
Finder display options	Finder	Finder

New Stuff to Check Out

The only way to become truly proficient supporting an operating system is to use it. Be sure to become familiar with the following new & improved features:

User Interface

1. The Dock
2. The Apple menu
3. System Preferences (Apple menu)
4. Dock preferences (Apple menu)
5. Force Quit (Apple menu)
6. Customize Toolbar (Finder/View menu)

System Preferences

7. Classic System Preferences
8. Dock System Preferences
9. International System Preferences
10. Internet System Preferences
11. Login System Preferences
12. Network System Preferences
13. Screen Saver System Preferences
14. Sharing System Preferences
15. Users System Preferences

`/Applications`

16. Address Book
17. Chess
18. Clock
19. Dock Extras
20. Image Capture
21. Preview

`/Applications/Utilities`

22. Applet Launcher
23. Console
24. CPU Monitor
25. DigitalColor Meter
26. Directory Setup
27. Disk Utility
28. Display Calibrator
29. Grab
30. NetInfo Manager
31. Network Utility
32. Print Center

33. ProcessViewer
34. Terminal

Review Questions

Operating Systems 101

1. What kind of limitations do cooperative multitasking put on the user experience?

2. What advantages do protected memory offer the user experience?

3. What advantages do pre-emptive multitasking offer the user experience?

4. Will Mac OS X take advantage of multiple processors? How?

Components of Mac OS X

5. What are the two major components of Darwin?

6. Is declining to install BSD a good idea? Why?

7. How is Quartz different from QuickTime and OpenGL?

8. What is Quartz based on?

9. What kind of applications run directly in Mac OS X (as opposed to Classic)?

10. Can you install extensions and control panels in Classic and make them available to applications running in Classic?

11. Does the Mac OS 9 System Folder used by Classic have to be on the same volume as Mac OS X?

Useful Mac OS X Jargon

12. How do you peek inside a bundle?

13. How is a bundle different from a package?

14. What is a Framework analogous to in Mac OS 9?

Mac OS X Installation

15. What directory format should a user select?

16. Why is the Mac OS X Setup Assistant a Good Thing?

17. What keyboard sequence might a user use to watch system components load at system startup?

18. What goes into a Library folder?

19. What's the difference between a user's Applications folder and the system's Applications folder?

20. Can different users see each others' Applications folders by default?

21. Where do user fonts go?

Security

22. What is a truly secure computer?

23. How many groups can a file belong to?

24. What's the main difference between root and an Administrative user?

25. Where do you go to enable root access?

26. How do you ensure a login prompt is displayed after a simple restart?

Old Stuff in New Terms

27. What do you tell a user who wants to turn on desktop printing?

28. How would you troubleshoot a situation where command-shift-3 doesn't perform a screen capture?

29. How do you set startup items?

30. How do you troubleshoot extension conflicts?

31. Where do you go to enable AppleTalk?

Review Answers

Operating Systems 101

1. What kind of limitations do cooperative multitasking put on the user experience?

Poorly behaved applications can take over the system, resulting in inconsistent or poor performance.

2. What advantages do protected memory offer the user experience?

A process can crash without affecting other processes or the operating system.

3. What advantages do pre-emptive multitasking offer the user experience?

The user stays in control.

4. Will Mac OS X take advantage of multiple processors? How?

Mac OS X will take advantage of two processors, when available. It does so via symmetric multiprocessing support (SMP). Individual applications can boost their multi-processor component by utilizing multithreading.

Components of Mac OS X

5. What are the two major components of Darwin?

Mach and BSD.

6. Is declining to install BSD a good idea? Why?

No. You lose a number of network services.

7. How is Quartz different from QuickTime and OpenGL?

Quartz focuses on 2D imaging.

8. What is Quartz based on?

Adobe Portable Document Format.

9. What kind of applications run directly in Mac OS X (as opposed to Classic)?

Cocoa, Carbon or Java.

10. Can you install extensions and control panels in Classic and make them available to applications running in Classic?

Yes.

11. Does the Mac OS 9 System Folder used by Classic have to be on the same volume as Mac OS X?

No.

Useful Mac OS X Jargon

12. How do you peek inside a bundle?

Hold down the Option key and click on the bundle.

13. How is a bundle different from a package?

Same thing.

14. What is a Framework analogous to in Mac OS 9?

Similar to a PowerPC shared library.

Mac OS X Installation

15. What directory format should a user select?

Mac OS Extended (HFS+).

16. Why is the Mac OS X Setup Assistant a Good Thing?

It helps ensure that networking services are correctly set up.

17. What keyboard sequence might a user use to watch system components load at system startup?

Command-V.

18. What goes into a Library folder?

Data and preferences utilized by applications.

19. What's the difference between a user's Applications folder and the system's Applications folder?

A User's applications are installed and used by that single user. The system's applications are available to all users.

20. Can different users see each others' Applications folders by default?

No.

21. Where do user fonts go?

/Users/myhome/Library/Fonts

Security

22. What is a truly secure computer?

Locked in a room and disconnected from a network.

23. How many groups can a file belong to?

One.

24. What's the main difference between root and an Administrative user?

Root can introduce and remove files from the System folder.

25. Where do you go to enable root access?

Netinfo Manager.

26. How do you ensure a login prompt is displayed after a simple restart?

It's an option in the Login System Preferences.

Old Stuff in New Terms

27. What do you tell a user who wants to turn on desktop printing?

Mac OS X does not support desktop printing. Check out the Print Center.

28. How would you troubleshoot a situation where command-shift-3 doesn't perform a screen capture?

Use the Grab utility instead.

29. How do you set startup items?

It's an option in the Login System Preferences.

30. How do you troubleshoot extension conflicts?

You don't—there isn't an equivalent of Mac OS 9 extensions.

31. Where do you go to enable AppleTalk?

Network System Preferences.

Further Reading

UNIX System Architecture

McKusick, et. al. The Design and Implementation of the 4.4 BSD Operating System. Boston: Addison-Wesley, 1996

Bach, M.J. The Design of the UNIX Operating System. Englewood Cliffs: Prentice Hall, 1986.

Wood, P. H. et. al. UNIX System Security. Indianapolis: Hayden Books, 1988

Silberschatz et al, Operating Systems Concepts, Fourth Edition.

Darwin Information

http://www.darwinfo.com
Unofficial site hosting a number of patches, ports, and an exhaustive FAQ.

http://www.opensource.apple.com/
Official home page for Darwin information.

Mac OS X General Info Sources

http://www.apple.com/macosx
The Mac OS X home page.

Mac OS X System Architecture

http://developer.apple.com/techpubs/macosx/SystemOverview/SystemOverview.pdf
An awesome high-level overview of how Mac OS X works.

http://developer.apple.com/techpubs/macosx/Kernel/KernelEnvironment/KernelEnvironment.pdf
A low-level overview of how the innards of Mac OS X works.

http://www.mit.edu/people/wsanchez/papers/USENIX_2000/
Discussion of the challenges involved in grafting UNIX with Mac OS.

Mac OS X Developer Information

http://developer.apple.com/macosx/
General information.

http://developer.apple.com/techpubs/macosx/macosx.html
Technical publications.

General UNIX Texts

Abrahams et al, UNIX for the Impatient, Second Edition

Sobell, M. G., A Practical Guide to the UNIX System, Third edition.

