Mac OS X
A Brief Technical Introduction

Leon Towns-von Stauber, Occam's Razor

LISA Hit the Ground Running, December 2005

http://www.occam.com/osx/
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opening Remarks</td>
<td>3</td>
</tr>
<tr>
<td>What is Mac OS X?</td>
<td>5</td>
</tr>
<tr>
<td>A New Kind of UNIX</td>
<td>12</td>
</tr>
<tr>
<td>A Different Kind of UNIX</td>
<td>15</td>
</tr>
<tr>
<td>Resources</td>
<td>39</td>
</tr>
</tbody>
</table>
This is a technical introduction to Mac OS X, mainly targeted to experienced UNIX users for whom OS X is at least relatively new.

This presentation covers primarily Mac OS X 10.4.3 (Darwin 8.3), aka Tiger.
This presentation Copyright © 2003–2005 Leon Towns–von Stauber. All rights reserved.

Trademark notices


NeXT®, NeXTstep®, OpenStep®, and NetInfo® are trademarks of NeXT Software. See <http://www.apple.com/legal/nexttmlist.html>.

Other trademarks are the property of their respective owners.
Answers

Ancestry

Operating System Products

The Structure of Mac OS X
It's an elephant

I mean, it's like the elephant in the Chinese/Indian parable of the blind men, perceived as different things depending on the approach.
Inheritor of the Mac OS legacy

Evolved GUI, Carbon (from Mac Toolbox), AppleScript, QuickTime, etc.

The latest version of NeXTstep

Mach, Quartz (from Display PostScript), Cocoa (from OpenStep), NetInfo, apps (Mail, Terminal, TextEdit, Preview, Interface Builder, Project Builder, etc.), bundles, faxing from Print panel, NetBoot, etc.

A new flavor of UNIX

More specifically, a BSD UNIX variant

Full set of command-line utilities, libraries, server software, etc.

All of the above
Operating System Ancestry of Mac OS X
UNIX components primarily based on FreeBSD

Also NetBSD and OpenBSD, as well as NeXTstep's version of BSD

Kernel based on Mach 3.0, MkLinux, and NeXT Mach
**What Is It?**

- **Mac OS X**
  - Apple's flagship operating system

- **Classic**
  - An instance of Mac OS 9 running in a self-contained execution environment within Mac OS X

- **Darwin**
  - The open-source foundation of Mac OS X

- **Mac OS X Server**
  - Mac OS X with additional server and administrative software
The Structure of Mac OS X

What Is It?

Structure

- User Interfaces
  - Platinum
  - Aqua
  - X Window System
  - Command-Line Shell
- Programming Interfaces
  - Macintosh Toolbox
  - J2SE
  - Carbon
  - Cocoa
  - Core Services
  - POSIX
- Kernel Environment
  - BSD Unix
  - Mach
  - I/O Kit Drivers
- Hardware
  - PowerPC Macintosh,
  - x86-Compatible PC
Open Standards

Open Source
Mac OS X is UNIX

On the whole, the similarities far outweigh the differences

Open Standards

Protocols and formats: TCP/IP, LDAP, IPsec, Zeroconf, SMB, NFS, PDF, ...

Hardware: SDRAM, USB, ATA, PCI/AGP, FireWire, HyperTransport, Wi–Fi, Bluetooth, ...
Much of OS X is based on open-source software

- Darwin, FreeBSD, NetBSD, OpenBSD, Mach
- Apache, CUPS, OpenLDAP, Postfix, Cyrus, OpenSSH, MySQL, Samba, BIND
- Bonjour, KAME, OpenSSL, XFree86
- Perl, Python, Tcl, Ruby
- And much more
A Focus on Differences
The Directory Hierarchy
HFS+
Mach
launchd
Directory Services
Administrative Users
Why Is It So Different?
While much of Mac OS X is familiar from other operating systems, there are many important differences that make it unlike any other UNIX system you've used.

Due to the approach of this presentation, and to human nature, we'll be focusing on these differences.
Parts of the OS X directory hierarchy look pretty familiar when viewed from the command line: /bin, /sbin, /dev, /usr, ...

/etc, /var, and /tmp are symlinks to subdirectories of /private

NeXTism related to NetBoot

/Applications, /Library, /System, /Users, /Network, /Developer

By default, non-root filesystems are mounted on subdirectories of /Volumes by diskarbitrati0nd

fstab configuration is possible, but unnecessary

From the Finder (the graphical file manager), things look different

Some directories, called bundles, look like single files in the Finder

Applications, frameworks, plug-ins, mailboxes, ...
A Different UNIX Directory Hierarchy
- The default local filesystem format is HFS+
- Coming from a UNIX background, HFS+ exhibits behaviors that take some getting used to
- Multiple forks per file
  - Data and auxiliary resources can be stored in separate filesystem objects
    - Resource fork used for things like file-specific icons, application multimedia, whatever
  - For the most part, the extra forks are invisible
    - Resource forks are visible with `ls -l filename/..namedfork/rsrc`
HFS+ supports extensive file metadata

Typical UNIX metadata: owner, group, permissions, mod date, etc.

BSD flags: immutable, append-only, etc. (man chflags)

Macintosh file attributes: type, creation date, locked, invisible, etc.

Stored in attribute fork (or in .filename on UFS)

In /Developer/Tools/, SetFile lists available flags, GetFileInfo
filename displays type, creator, and flags

NTFS-compatible ACLs (added in Tiger)
Case-preserving, but case-insensitive

- `ReadMe` is stored with mixed case retained for display, but it can also be accessed as `README`, `Readme`, or `readme`

- `ReadMe` and `README` cannot exist in the same directory

Panther introduced fully case-sensitive HFS+ variant

- Tip: `tcsh` command completion is still case-sensitive unless you set `complete = enhance` in `~/.tcshrc`:

The path separator is a colon (`:`), not a slash (`/`)

- Pathnames are converted on-the-fly by the kernel, so that colons look like slashes

- Carbon apps convert slashes back to colons
Application libraries access filesystem objects by numerical file IDs, not pathnames.

- File IDs are unique per disk volume.
- Lookups are faster than by pathname.
- Kind of like inode numbers; in fact, `ls -i` displays file IDs on HFS+.
- File IDs don't change when files are moved around on a disk volume.

If you know a file's ID, and the the ID of the volume it's on, you can always access it as `./vol/vol_ID/file_ID`.

If you know the ID of the directory containing a file, you can access it as `./vol/vol_ID/dir_ID/filename`. 
Aliases

An alias is a lightweight reference to a file or directory

- Like a symbolic link, but uses both pathname (preferably) and file ID (as backup)

An alias can continue to refer to a file even if it's moved (on the same volume) or renamed

Both aliases and symlinks are useful in different circumstances

- If the actual pathname is all-important, or you need to use it from the CLI, use a symlink

No way to create symlinks from GUI, or aliases from CLI
Hard links

On UFS, a hard link is simply another reference to a file's inode

With no inodes, HFS+ lacks support for hard links

OS X supports hard links for backwards compatibility, but they're implemented in the kernel as symbolic links, faked out to look and act like hard links

Number of links shown for a directory in `ls -l` output counts all items within the directory, including files

HFS+ lacks support for sparse files; void extents are zero-filled

HFS+ supports journaling, for faster recovery after crash

See Wil Sanchez USENIX paper for more on filesystem design decisions
Developed at CMU as experiment in microkernel design

Early versions integrated BSD, which NeXT used

Mac OS X kernel primarily derived from Mach 3.0 used in MkLinux, with NeXT enhancements

Still a monolithic kernel, for performance

Manages memory, processes, and hardware access

Theoretically capable of highly scalable multiprocessing, but Apple has so far released only dual- (and now quad-) processor machines

Better kernel resource locking in Tiger for improved multiprocessing
Mach features an efficient virtual memory implementation

- Backing store is file-based
  - It doesn't use a specially formatted disk partition (e.g., Solaris)
  - Definitive performance comparisons haven't been made, but it's sufficiently fast to not be a problem
  - Of course, you're much better off with enough RAM to avoid paging in the first place

- Allocated as individual files in `/var/vm/`, acc. to the parameters of the `dynamic_pager` command in `/etc/rc`

- VM disk usage grows and shrinks dynamically

- Use `vm_stat` (note the underscore) to keep an eye on memory usage
launchd is a replacement for init, rc scripts, SystemStarter, Mach bootstrap daemons, cron, inetd, login hooks...

Not quite there yet; other methods are deprecated, but still functional in Tiger

Very similar design goals to Solaris 10 SMF

launchd is now PID 1

System daemons or per–user agents are configured by XML property lists

See launchd.plist man page for syntax

Daemon config files are in either /System/Library/LaunchDaemons/
or /Library/LaunchDaemons/

Agent config files are in /System/Library/LaunchAgents/, /
Library/LaunchAgents/, or ~/Library/LaunchAgents/
**launchctl** is the launchd user interface

Start and stop services, load/unload/list configured services, etc.

Example config file, for **syslogd** (translated from XML to NeXT property list format for clarity):

```plaintext
{
    Label = "com.apple.syslogd";
    OnDemand = 0;
    ProgramArguments = ("/usr/sbin/syslogd");
    ServiceDescription = "Apple System Log Daemon";
    ServiceIPC = 0;
}
```
Mac OS X has a deep history with directory services, owing to its NeXT lineage, and is probably the most flexible client and provider of directory services there is.

In OS X, lookups for many kinds of configuration data are made through the Directory Services API.

For legacy UNIX programs, the `getXbyY` system calls are rewritten to proxy lookups through `lookupd`, a daemon that makes use of DS.

The data sources consulted by DS are configured in the Directory Access application.

OS X includes its own directory service, named Open Directory, based on OpenLDAP.
A Different UNIX

Directory Services

Directory Access

<table>
<thead>
<tr>
<th>Enable</th>
<th>Name</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Active Directory</td>
<td>1.5.3</td>
</tr>
<tr>
<td></td>
<td>AppleTalk</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td>Bonjour</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td>BSD Flat File and NIS</td>
<td>1.2.1</td>
</tr>
<tr>
<td>✔️</td>
<td>LDAPv3</td>
<td>1.7.2</td>
</tr>
<tr>
<td></td>
<td>NetInfo</td>
<td>1.7.3</td>
</tr>
<tr>
<td>✔️</td>
<td>SLP</td>
<td>1.2.1</td>
</tr>
<tr>
<td>✔️</td>
<td>SMB/CIFS</td>
<td>1.2</td>
</tr>
</tbody>
</table>

Configure...

Click the lock to prevent further changes.

Directory Access
Host–local information still in NetInfo

- NeXT-legacy access protocol and database format
- Within a NetInfo-formatted database, information is organized in a directory hierarchy, analogous to a filesystem directory hierarchy
  - Root is /, subdirectories include /machines, /users/leonvs, etc.
- Nodes have properties, each one key to a set of values
  - Properties include name, uid, ip_address, passwd, etc.

The Big Surprise

- Most traditional UNIX flat files in /etc (passwd, group, etc.) aren't used by default (except in single-user mode)

CLI tools: dscl, nicl, nidump, niload, nireport, nifind, nigrep
### NetInfo Manager

#### Directory Services

A Different UNIX

#### File System Structure

- `/` (root directory)
- `/aliases`
- `/groups`
- `/machines`
- `/mounts`
- `/networks`
- `/printers`
- `/protocols`
- `/rpcs`
- `/services`
- `/users`
- `/leonvs`

#### Property-Value Table

<table>
<thead>
<tr>
<th>Property</th>
<th>Value(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>leonvs</td>
</tr>
<tr>
<td>home</td>
<td>/Users/leonvs</td>
</tr>
<tr>
<td>authentication_authority</td>
<td>;ShadowHash;</td>
</tr>
<tr>
<td>passwd</td>
<td>*********</td>
</tr>
<tr>
<td>_writers_hint</td>
<td>leonvs</td>
</tr>
<tr>
<td>_shadow_passwd</td>
<td>leonvs</td>
</tr>
<tr>
<td>_writers_picture</td>
<td>leonvs</td>
</tr>
<tr>
<td>realtime</td>
<td>Leon Towns-von Stauber</td>
</tr>
<tr>
<td>uid</td>
<td>501</td>
</tr>
<tr>
<td>shell</td>
<td>/bin/tcsh</td>
</tr>
<tr>
<td>generateduid</td>
<td>1687A883-C5C2-4052-97DE-B17...</td>
</tr>
</tbody>
</table>

Click the lock to make changes.
By default, root logins are disabled on OS X (by an invalid password)

To enable, use NetInfo Manager->Security->Enable Root User, dsenableroot, or simply sudo passwd root

On Mac OS X Server, root password same as initial admin user

Consider changing one or the other, so they're not the same

Administrative work is designed to be accomplished by members of the admin group, who possess special privileges

Some privileges are configurable, and may be removed or reassigned, while others are hard-coded to the admin group

NB: The user account created during installation is in the admin group
File permissions

Directories and files in /Applications/, /Library/, and /Developer/ are owned and writable by group admin, permitting software installation

sudo

Admin users have superuser access to CLI commands, configured in /etc/sudoers

su

Can only su to root if in group admin or wheel

Configurable in /etc/pam.d/su
Authorization Services

Part of the Security framework

Gives admin users superuser privileges for certain GUI activities: running software installers, configuring directory access, changing certain things in System Preferences, etc.

Configured in /etc/authorization

NetInfo

Admin users have full write access to NetInfo domain contents

Hard-coded
Apple Filing Protocol (AFP) server

- Administrators can connect as any user, authenticating with their own password, and they gain special access privileges

- Hard-coded to admin group, but can be configured with properties in Open Directory, under /config/AppleFileServer in local domain

  - attempt_admin_auth: Determines whether authentication is attempted against administrator passwords

  - special_admin_privs: Grants admins read access to all folders

  - permissions_model: Gives admins the ability to change ownership of all files if set to unix_with_classic_admin_permissions

  - admin_gets_sp (Boolean, default 0): Lets admins mount share points instead of volumes
Some important differences: Quartz vs. X11, HFS+ vs. UFS, Objective-C vs. C++, NetInfo vs. LDAP, AFP vs. NFS, file-based VM, etc.

Many design decisions were made in the middle to late 1980s, during the development of NeXTstep.

Many of today's ubiquitous technologies (X11, C++, YP/NIS, LDAP) were not yet well-established.

NeXT was among the first to implement a UNIX GUI, a standard OO dev environment, directory services, etc., and happened to choose differently than the rest of the industry later did.

Some changes were made to support Apple's existing user base.

HFS+, AFP, secure default config.
But why does Apple stick with technologies that require special training?

Because some are just better than the alternatives

Objective-C is a cleaner, more flexible language than C++

HFS+ is arguably more capable than UFS under certain circumstances

Most other UNIX platforms also intend to replace UFS, or have already done so

UFS2 (FreeBSD), XFS (IRIX), ZFS (Solaris), etc.

Quartz performs well and is self-consistent

Because Apple controls these technologies, and can drive their improvement

They are willing to accommodate alternatives (UFS, NFS, X11, C++) or even replace proprietary technologies (NetInfo -> LDAP)
Apple's Mac OS X Site

http://www.apple.com/macosx/

Mac OS X Hints

http://www.macosxhints.com/

O'Reilly Mac DevCenter

http://www.macdevcenter.com/

The Challenges of Integrating the Unix and Mac OS Environments

macos-x-server (Apple)


macosx-admin (Omni Group)

http://www.omnigroup.com/developer/mailinglists/macosx-admin/

macosx-talk (Omni Group)

http://www.omnigroup.com/developer/mailinglists/macosx-talk/

security-announce (Apple)

Mac OS X Tiger for Unix Geeks
  Brian Jepson, Ernest E. Rothman

Mac OS X Tiger in a Nutshell
  Andy Lester, Chris Stone, Chuck Toporek, Jason McIntosh

Running Mac OS X Tiger
  Jason Deraleau, James Duncan Davidson

Essential Mac OS X Panther Server Administration
  Michael Bartosh, Ryan Faas