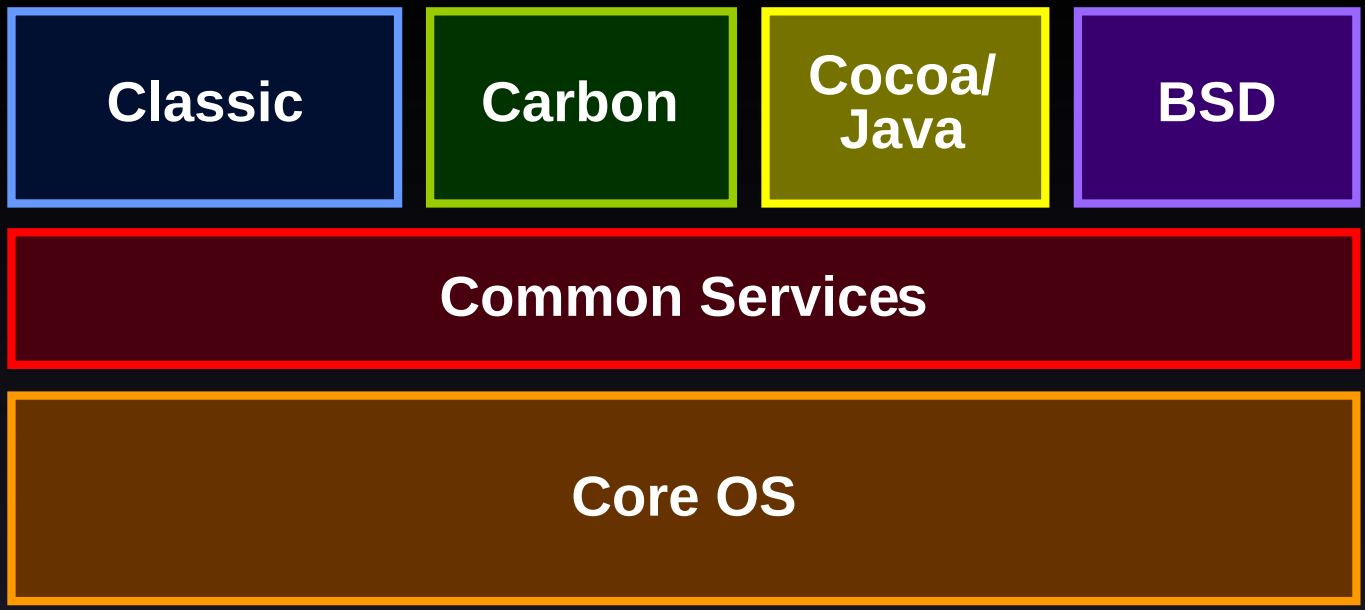
Welcome

To Advance through Presentation
Use Page Up and Page Down Keys

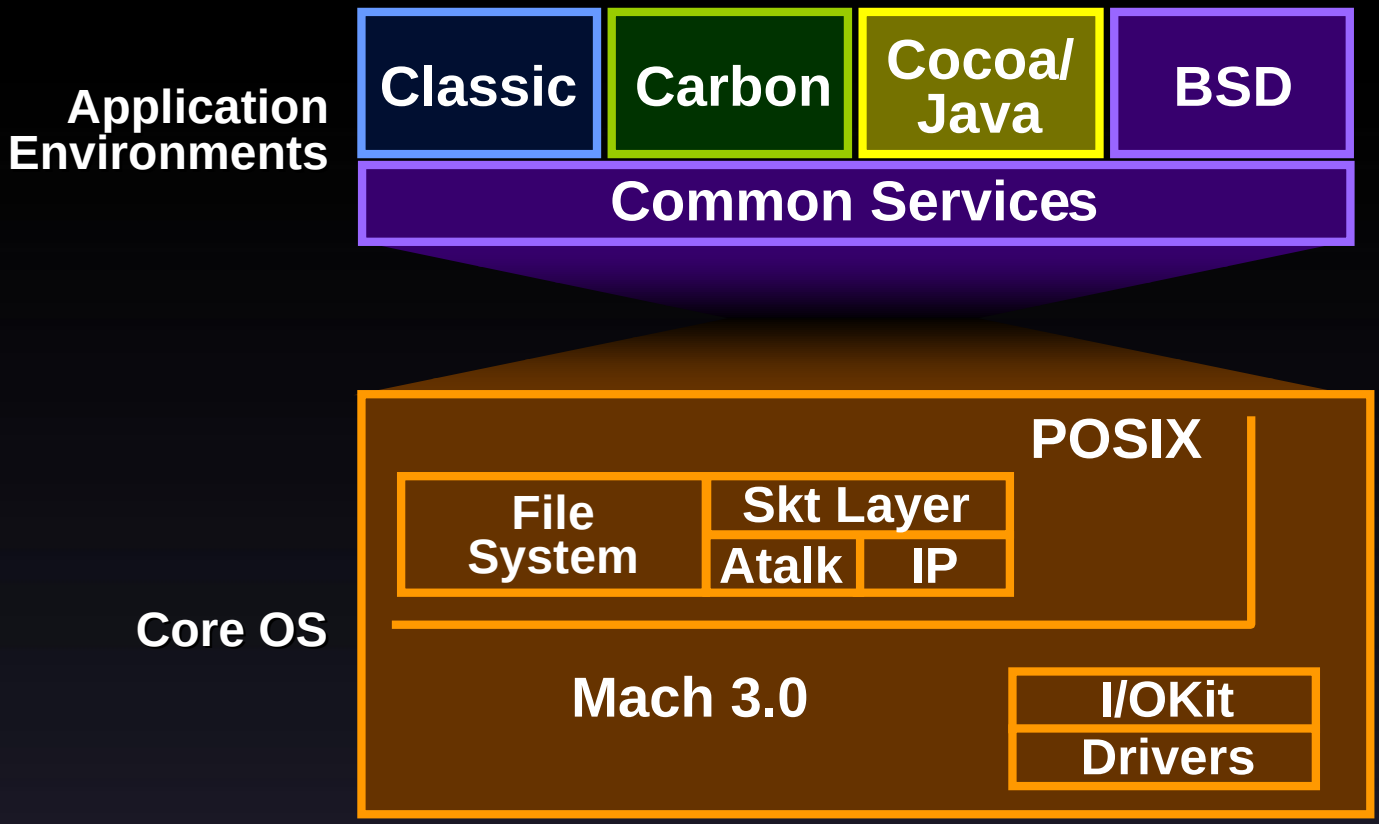
I/O Drivers—Mac OS X

Dean Reece
I/O Kit Team Manager

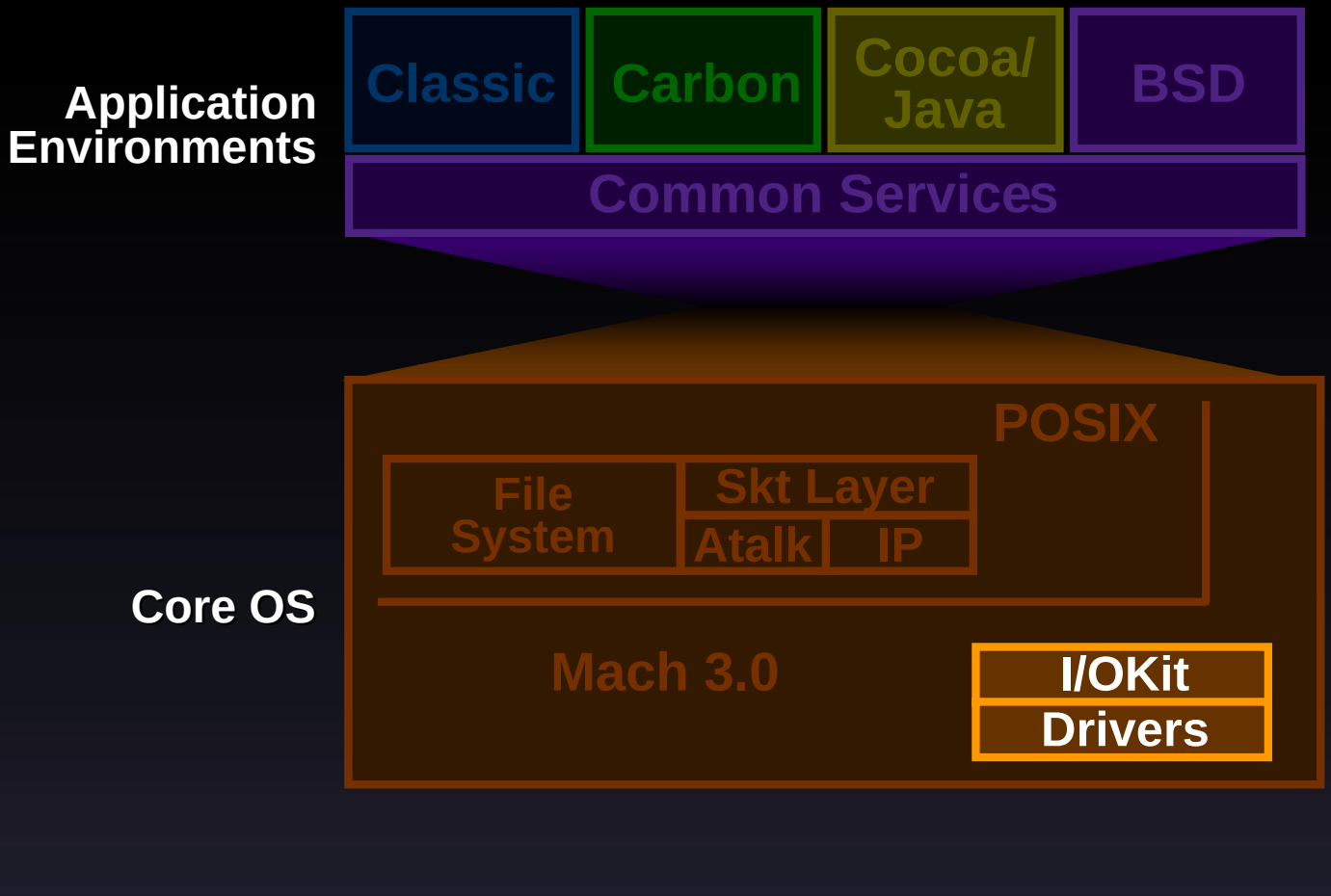
Mac OS X Architecture



Core OS and Mac OS X



Core OS and Mac OS X



What Is I/O Kit?

Everything needed to
develop and use Kernel Extensions

- A set of SDKs
- A set of libraries
- Common services

I/O Kit Design Goals

I/O Kit is intended to make it as easy as possible to develop Mac OS X drivers

- Protected Address Space
- Multiple Threads
- Preemptive Scheduling
- SMP Efficient

I/O Kit Lineage

- Derived from NeXT's DriverKit
- DriverKit treated as a prototype
- Added IORegistry and IOCatalog
  - Plug-n-Play/Hot Swap
  - Power Management
  - Better App-to-Driver interaction
- Added Several New Technologies
- Added support for some NDRVs

Objective C to C++

- Full C++ difficult in kernel environment
- Embedded C++ offers the right tradeoffs
    - Strict subset of C++
    - Single inheritance only
    - No templates, static objects, RTTI…
- Embedded C++ web site:

http://www.caravan.net/ec2plus/index.html

Kernel Extensions

- Package new functionality for Mac OS X kernel
  - Filesystem Plugins
  - Network Implants
  - I/O Kit Drivers and Families
- Familiar user experience
- Live in the System folder
- Optional “double click” action

Kernel Extensions

- Type and Creator are "*kext*"
- Optional Icon
- "Driver" is contained in App package
  - Property List—serves as Table of Contents
  - Relocs—Relocatable code fragment(s)
  - Optional UI Elements—about box, help, . . .

Kernel Extension Manager

- (Un)Loads Kernel Extensions as needed
- Apps can register new extensions at runtime
  - Can have extensions associated with single apps
  - No need to reboot after installing new software
- Default app for *kext* type files
  - Can handle “double click” action
  - Can Auto-update extensions
  - Can show user which extensions are being used

I/O Kit Architecture

- Object-oriented
- Two primary classes
  - IOObject
  - IOService
- All drivers inherit from IOService
- IORegistry tracks IOService objects

I/O Kit Infrastructure

- Always available in kernel
- Services needed by all drivers
  - Kernel Extension Manager
  - IORegistry and User-Client access
  - Memory Management
  - Interrupt Management and Timer Support
  - Thread Management and Synchronization
  - Common classes

I/O Kit Families

- Families build on the I/O Kit Infrastructure
- Each interface or protocol gets a family

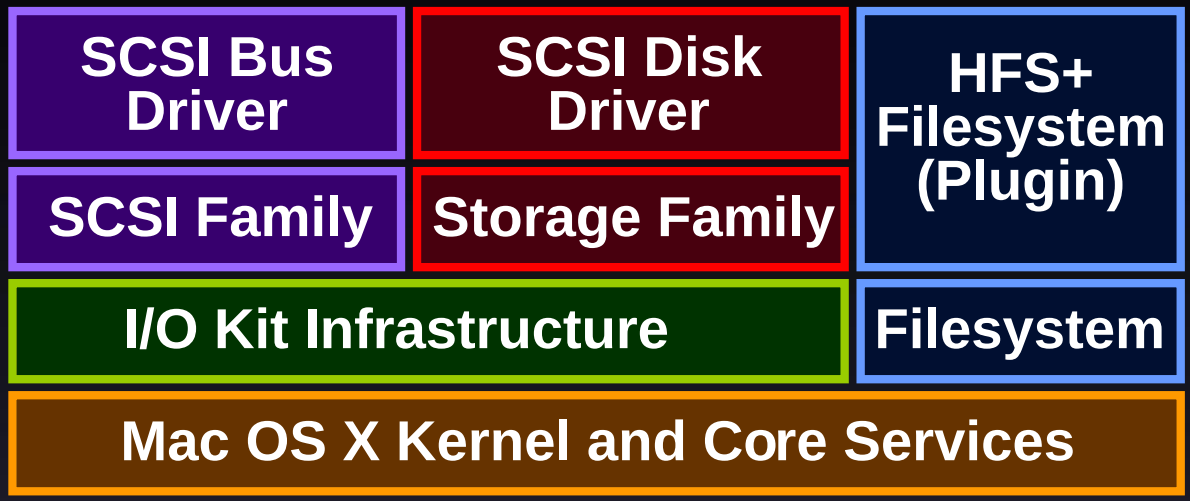
PCI	**Network**	**ATA/PI**	**Video**
USB	**FireWire**	**SCSI**	**Graphics**
ADB	**Serial**	**Mass Storage**	**Human Interface**

I/O Kit Families

- A Family includes everything needed to develop for a single interface or protocol:
  Classes, Headers, Consts, Structs, Enums, Libraries...
- Families built into kernel now, will be loadable Kernel Extensions later

I/O Kit Families

- I/O Kit builds on Mac OS X Core Services
- Families build on I/O Kit Infrastructure
- Drivers build on I/O Kit Families



I/O Kit Direct Drivers

Godfrey van der Linden
I/O Kit Team Tech Lead

Why Direct Drivers

- Efficient, robust direct drivers are difficult to write
- I /O Kit is designed to make it easier without sacrificing flexibility and power
- Direct Drivers support is only small part of I/O Kit

Easiest Solution

- Mac OS 8 Drivers
- We have done a lot of research into supporting old drivers
- Unfortunately no efficient general solution was found
- Mac OS 8 and Mac OS X have fundamentally different operating systems

Traditional Unix Drivers

- Traditional upper/lower drivers
- Complex operating system interactions to prevent clobbering data
- Almost unimplementable on multi-processors

I/O Kit Drivers

- Object-oriented
- Single-threaded
- Easy VM model
- And a real helpful extra

Object Oriented

- Models most common types of direct drivers, in other words a HAL
- The family classes interact with the OS
- Object-oriented solutions are flexible and easy to extend

Single Threaded

- Threading makes multiprocessor support a ‘gimme’
- Each piece of hardware has its own thread
- Usually no primary interrupts

Simple VM Model

- Mac OS X Kernel deals with the VM issues
- Uses simple prepare/cleanup I/O style
- Some complexity if writing a family

I/O Kit Open Source

- We expect to be opening the I/O Kit source
- No more worries about code disappearing into a poorly documented library
- The I/O Team will also be responsive to fixes and suggestions

Roadmap

Mac OS X File System	Hall A1 **Fri., 2:30pm**
Mac OS X Networking Overview	Hall A2 **Fri., 9:00am**
Mac OS X—Core OS Feedback Forum	Hall B **Fri., 4:00pm**

Q&A

Think different.™

Welcome

To Advance through Presentation
Use Page Up and Page Down Keys