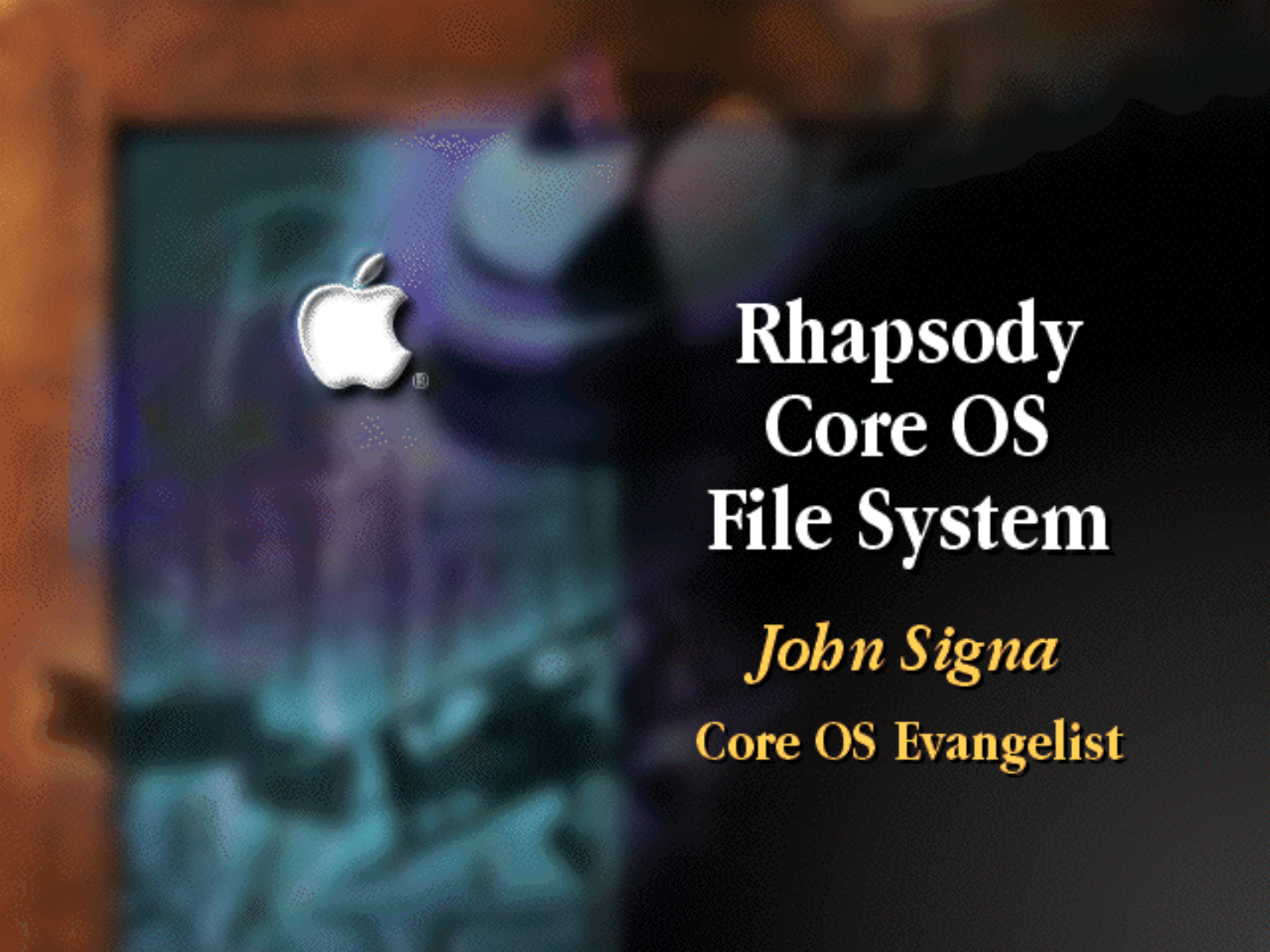
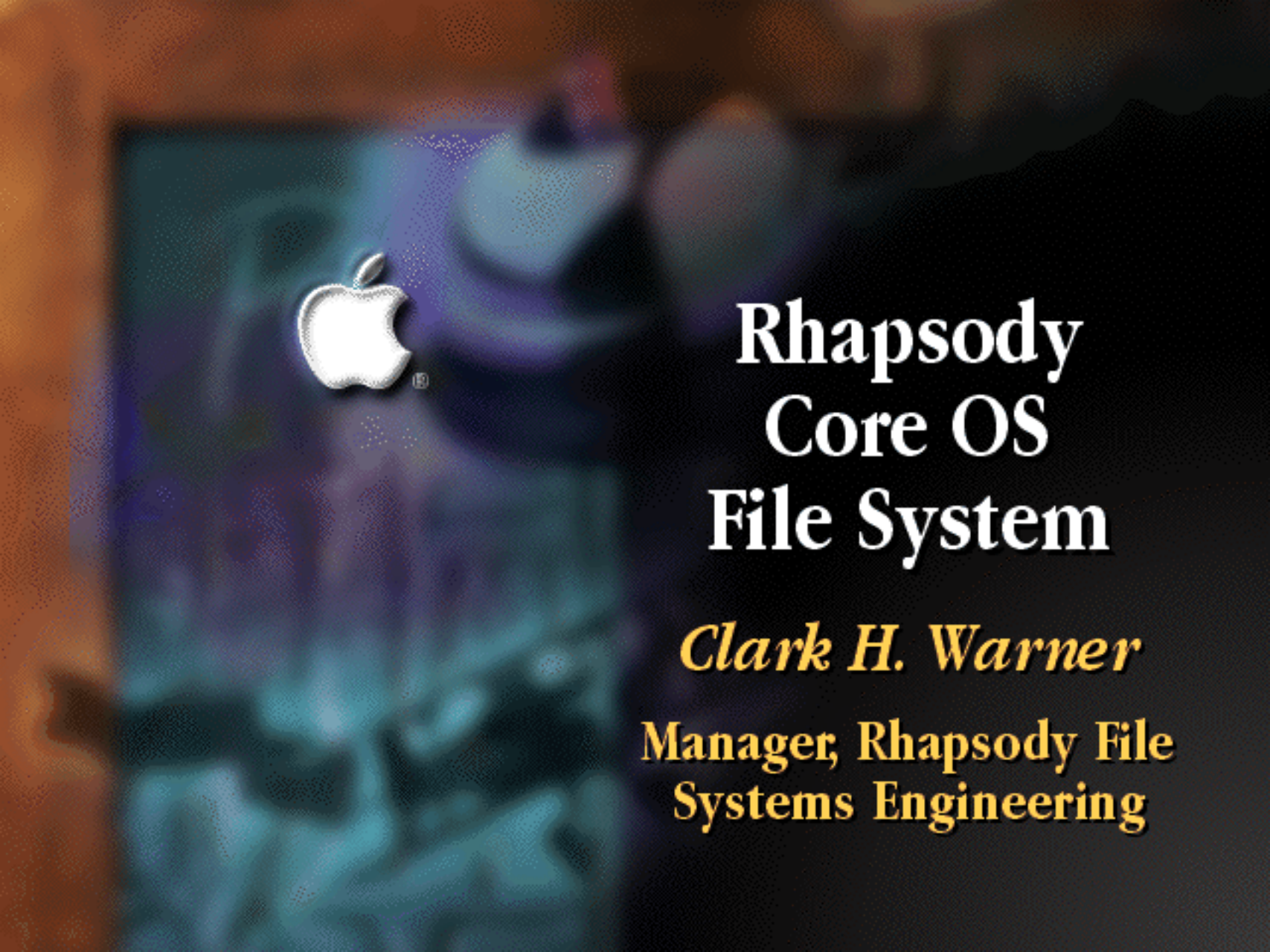

Worldwide
Developers
Conference


Rhapsody
Core OS
File System
John Signa
Core OS Evangelist

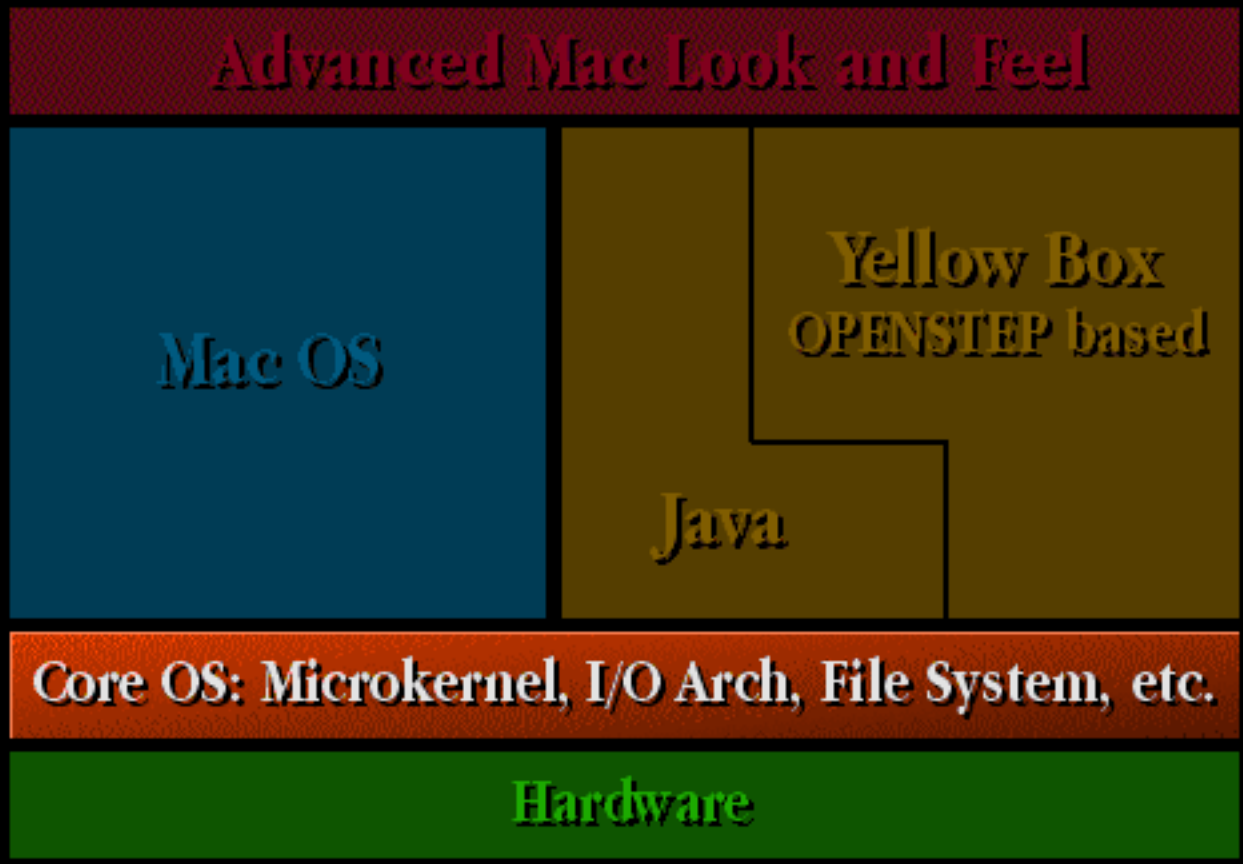


Rhapsody Core OS File System

Clark H. Warner

Manager, Rhapsody File Systems Engineering

Where Is the File System?



Core OS II—File Systems

- General Framework Supports
  - Wide range of data formats and storage media
  - Efficient access to features and benefits of native volume formats
  - Efficient APIs
- File level metadata support for a superior user experience
- Mature foundation
  - Performance
  - Stability

Introduction

Today's topics are…

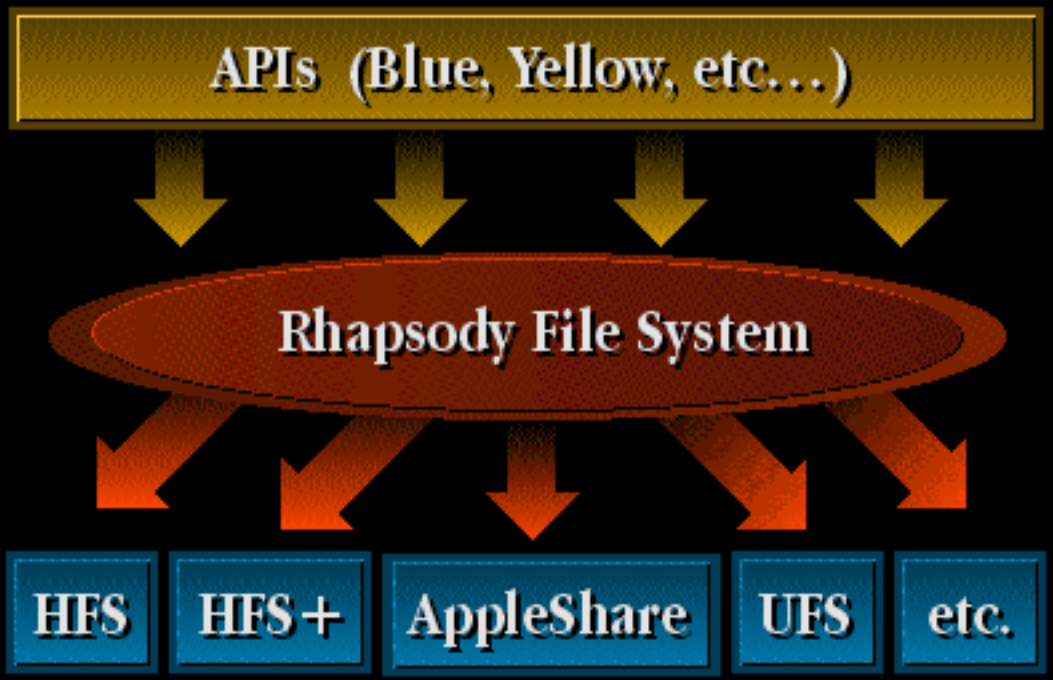
- Architecture
- APIs
  - 3 Classes (Yellow Box, POSIX, Blue Box)
- File system plug-ins and utilities
- Volume formats
- Timing
- Where to go for more

Introduction *(cont.)*

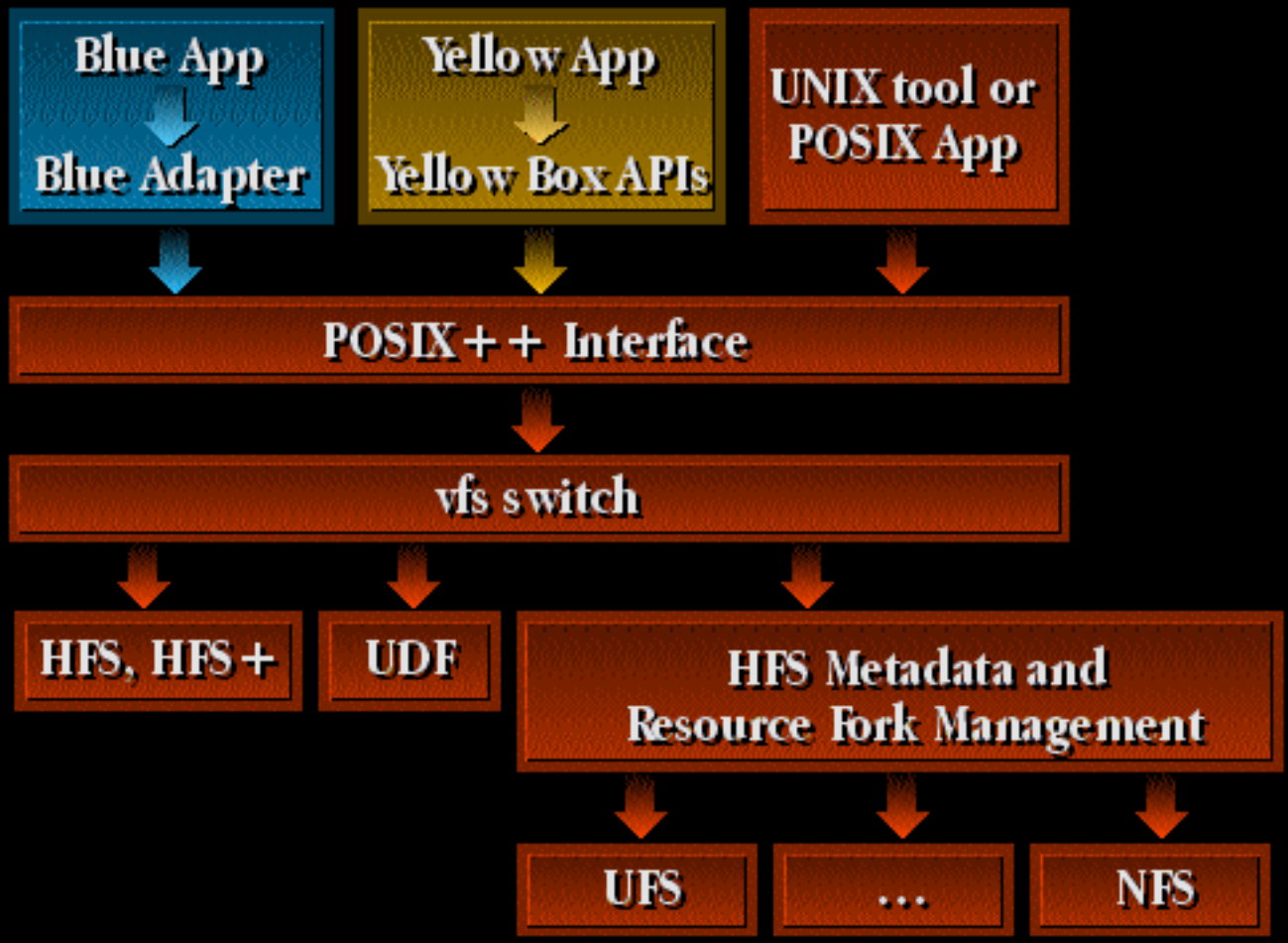
Who cares?

- Architecture—everybody
- APIs
  - Application developers care about Yellow
  - Server app developers care about POSIX
  - Developers of Mac OS apps care about Blue Box APIs
- File system plug-ins—file system developers
- Everything else—everyone (even users)

Architecture



APIs


Blue App
Blue Adapter
Yellow App
Yellow Box APIs
UNIX tool or POSIX App
POSIX++ Interface
vfs switch
HFS, HFS+
UDF
HFS Metadata and Resource Fork Management
UFS
...
NFS


Yellow Box
Chris Kane
Yellow Box Development

Yellow Box

- APIs mostly provided by Foundation, the lowest level Yellow Box framework
  - Some APIs provided by other frameworks
- Sits on top of a platform's file system APIs

Yellow Box *(cont.)*

- Files as parameters are usually named with paths in the API
- Three basic mechanisms for accessing files
  - Reading and writing the contents of objects
  - File descriptors with NSFileHandle
  - File management abstraction based on URLs

Yellow Box *(cont.)*

Reading and writing the contents of objects

- Creating strings and data objects with the contents of files and writing them out to files
- Satisfactory for many applications

Yellow Box *(cont.)*

Reading and writing the contents of objects—examples

- Create a string containing the contents of a file

```
contents = NSString.
  stringWithContentsOfFile(
    @"/Library/README");
```

- Write the bytes of an NSData object to a file

```
dataObject.writeToFile(
  @"/Documents/UI_Proposal");
```

Yellow Box *(cont.)*

File descriptors with NSFileHandle

- Covers Windows™ file handle/socket, Mac OS, and POSIX file descriptor facilities
- Supports usual operations like reading, writing, seeking
- API using file offsets and sizes uses 64-bit quantities
- Some operations can be done asynchronously
  - i.e., “callbacks” via standard notification mechanism

Yellow Box *(cont.)*

File descriptors with NSFileHandle—example

```
handle = NSFileHandle.
  fileHandleWithStandardInput();

... add an observer to the
    notification center here ...

handle.readInBackgroundAndNotify()
```

Yellow Box *(cont.)*

File management abstraction based on URLs

- Based upon the URL abstraction framework in Foundation, and the “file:” scheme handler
- Foundation translates “file:” URLs as necessary to the notation required by specific platforms
- Provides operations like removing files, getting and setting of attributes, retrieving file contents

Yellow Box *(cont.)*

File management abstraction (cont.)

- APIs can mostly be used synchronously or asynchronously
  - Notifications for asynchronous operations
- Handlers and delegation allow for parameterization of the handling of operations
  - For example, an alert panel can be shown on an error

POSIX

- Useful for UNIX legacy and server apps
- Will stay POSIX compliant with high level of compatibility
  - Hard links
  - UNIX permissions
  - Not case sensitive but allows access to files that differ only in case

Blue Box

- Compatibility with Mac OS IM: Files
- Disk Driver supports three modes
  - 1. HFS Partition as a Blue Box—only HFS volume
    - Use existing HFS partitions in the Blue Box
    - Very compatible
  - 2. Yellow file (disk image) as a Blue Box—only HFS volume
    - For convenience and compatibility

Blue Box *(cont.)*

- **Disk Driver modes** *(cont.)*
  - 3. Blue Box and Yellow Box share volumes (UFS/NFS, HFS, CD-ROM etc.)
    - Looks like HFS or HFS Plus volumes to Blue Box via new Core OS APIs
    - Less compatible, but better interoperability
    - Software that expects to see all file system requests through patches will NOT work for volumes in this mode

Blue Box *(cont.)*

- File system patches will work
  - But will not see any activity outside of Blue Box
- FSM is the file system plug in model
  - But no drivers and fewer services

Virtual File System (vfs)

- Internal kernel interface
- All core file system operations pass through
- Expanded to allow use of intrinsic volume format features
  - Resource fork
  - Metadata
- Not an application level service

File System Plug-Ins

- BSD 4.4 vfs interface API with extensions
  - Provides all basic file system operations
    - lookup, open, read, mkdir, etc.
- Mac OS style for Blue Box only

Utility Support

Use for utilities

- Compression, encryption, virus checking
- No patching model in Yellow
- Answer may be stackable file system support

Volume Formats

- Will support a wide range of volume formats
- Volume format capabilities will decide user experience
- Most functional
  - Full user experience enabled by full metadata support
  - Allows booting (on local devices only)
  - Includes persistent 32-bit ID for full alias support

Volume Formats

- Most functional
  - HFS Plus
  - AppleShare
- Additional
  - May not support all features
  - UFS, FAT, VFAT/FAT32, NFS, ISO9660, UDF, HFS

Volume Formats

Benefits of HFS Plus

- Small allocation blocks—even on large volumes
- Compatible with Mac OS
  - DirIDs and FileIDs, forks, HFS metadata
- Large volumes and files
- Direct (255 character) Unicode support
- Extensions for Rhapsody
  - Permissions, hard links, more dates

Capability Timeline

- Developer
  - UFS only (BSD 4.4)
  - Please code to Yellow Box APIs
- Premier
  - HFS Plus, NFS, AFP
- Unified
  - ISO9660, UDF, FAT

Server Requirements

- Volume format appropriate for the market
  - Logical volume management
    - Spanning
    - Striping
    - Mirroring
  - Metadata logging file system


Wrap-Up
John Signa
Core OS Evangelist

How to Find Out More

- Mac OS Compatibility Lab
  - Open all week
- Rhapsody Drivers
  - Session 203, Thursday, 8:30–9:30
- Mac OS Support in Rhapsody's Blue Box
  - Session 204, Wednesday, 1:30–2:50
- Core OS Change
  - Session 101-R2, Thursday, 3:10–4:20

How to Find Out More *(cont.)*

- **Yellow Box API**
  - Session 208, Wednesday, 3:10–4:10
- **Intro to OpenStep's Foundation Framework**
  - Session 207, Wednesday, 4:30-5:30
- **Core OS Feedback Forum**
  - Session 292, Thursday, 11:10–12:10
- **Developer feedback**
  - rhapsody-dev-feedback@apple.com


Q&A


Worldwide
Developers
Conference