



Macoun'10

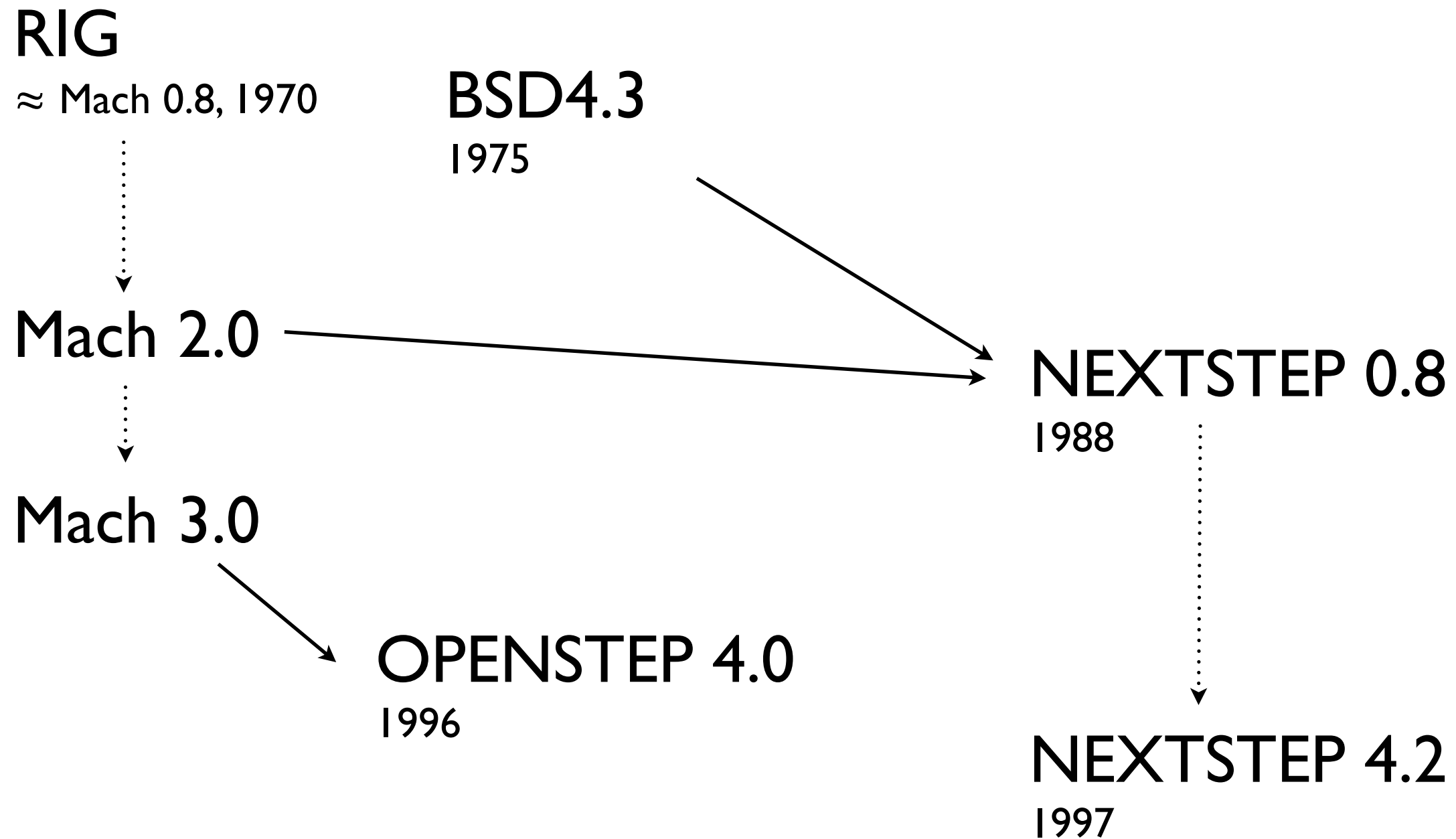
Darwin Kern selbstgebaut

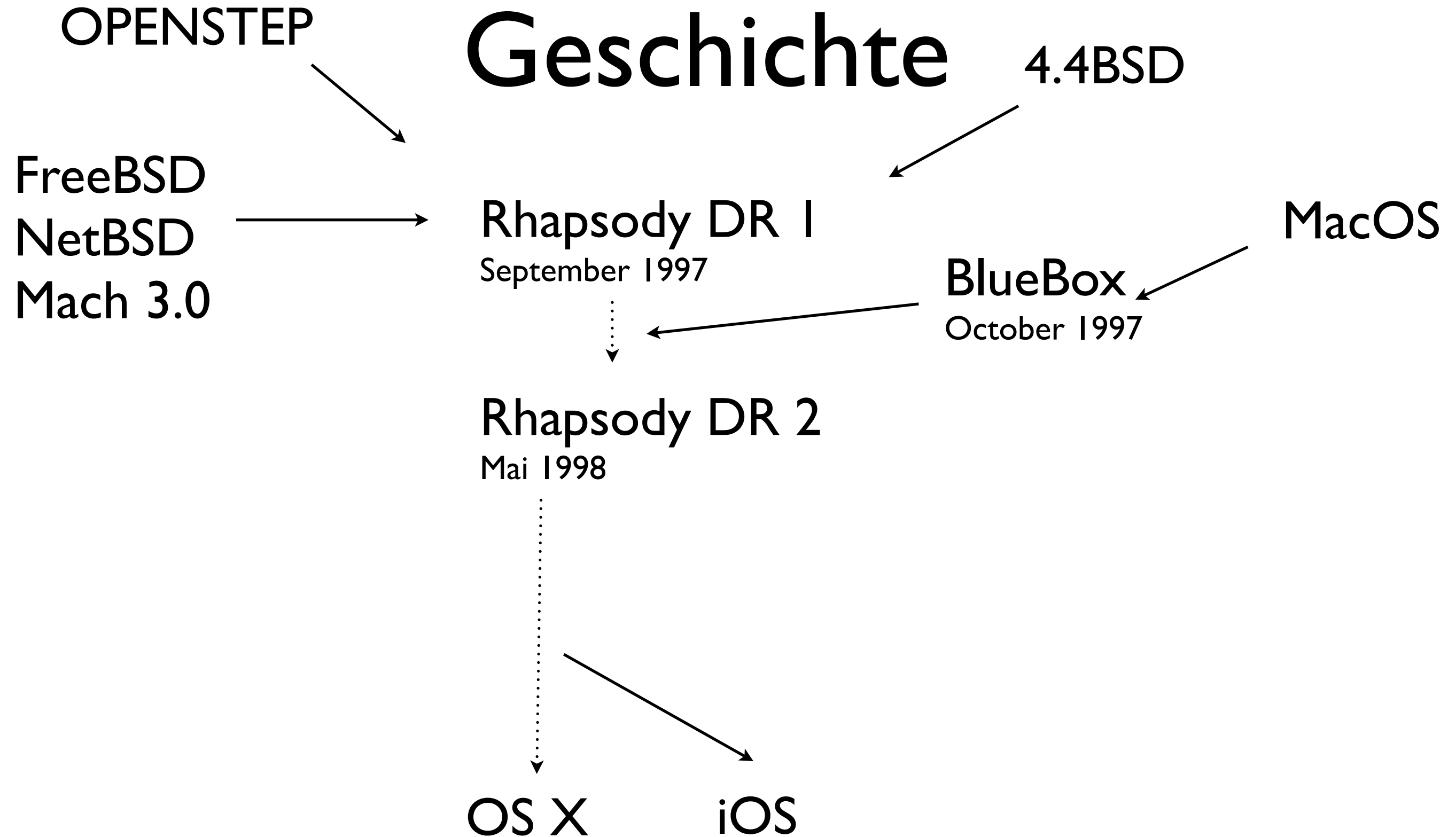
Alexander v. Below

Agenda

- Was ist der OS X Kern?
- Wie kann man ihn erweitern?
- Wie kann man ihn selbst bauen?

Geschichte





Was macht ein Kern

- Muss ich das hier wirklich erklären?
- Memory Management
- Scheduling
- Hardware

Stichworte...

Mach

- Der Mac OS X Kern ist nicht “Mach”
- Der OS X Kern ist xnu („X is not UNIX“)
- xnu beinhaltet Mach, BSD und das I/O-Kit.

Microkernel

- Der OS X Kern ist kein Microkernel
- Aus der Mach Codebasis kann man einen Microkernel machen
- xnu ist ein monolithischer Kern
- BSD und die meisten Treiber laufen im Kernel Mode

FreeBSD

- Der Mac OS X Kern basiert nicht auf dem FreeBSD Kernel
- xnu enthält etwas FreeBSD Code

C++

- xnu ist nicht in C++ geschrieben
- Das I/O-Kit ist in Embedded C++ geschrieben
- Mach und BSD sind in C geschrieben

UNIX

- Mac OS X ist “UNIX”.
- Erst seit 10.5 Leopard.
- Es hat den POSIX Conformance Test bestanden
- Es darf das “UNIX®” Trademark verwenden
- Es enthält keinen AT&T UNIX Code.

Open Source

- Der Mac OS X Kern und die meisten UNIX Teile sind Open Source.
- Es gibt kein Live Repository
- Einiger Code fehlt
- Aber man kann daraus ein funktionierende System bauen

Kernelerweiterungen

- Viele Dinge in OS X können vom Userspace aus erledigt werden
 - Drucker, Scanner und Kameratreiber
 - Viele Kernel Dienste (auch) Geräteklassen vom Userspace erreichbar
 - Userspace IOKit.framework
- Es lohnt sich, nachzusehen

Gefahren

- Kernelcode hat keine Schutzmechanismen
- Ein Fehler reißt das System herunter
- Kernel Debugging ist komplizierter
- Code läuft nicht schneller im Kern

Kernel Extensions sind für...

- Low-level Treiber
 - Die Zugang auf Hardwareregister brauchen
 - Die Interrupts verarbeiten sollen
 - Die ihrerseits Clients im Kern haben
- Netzwerkstack Filters
- Kernel Authorization Plug-Ins
- Dateisysteme (oder MacFUSE ...)

kext Werkzeuge

Entwicklung	Deployment	System
kextutil	kextload	kextd
kextlibs	kextunload	kextcache
kextstat		
kextfind		

Kext Debugging

- kextutil
- kext_logging

Kext Debugging (echt)

- Zwei Maschinen Setup
- Ethernet gdb
- Firewire kdp oder gdb
- Serial ddb

Don't do this at home...

- Kexts run within the kernel's 32- or 64-bit virtual address space
- No sandboxing or address space protection
- Errors bring down the whole OS (kernel panic)
- All code running within the kernel is inherently trusted
- If you can implement your feature in user space, do so

xnu Selberbauen












- NICOLAS
- Eigene, angepasste Kerne
- Open Source für Sicherheitsrelevante Anwendungen
- Spass!

Opensource

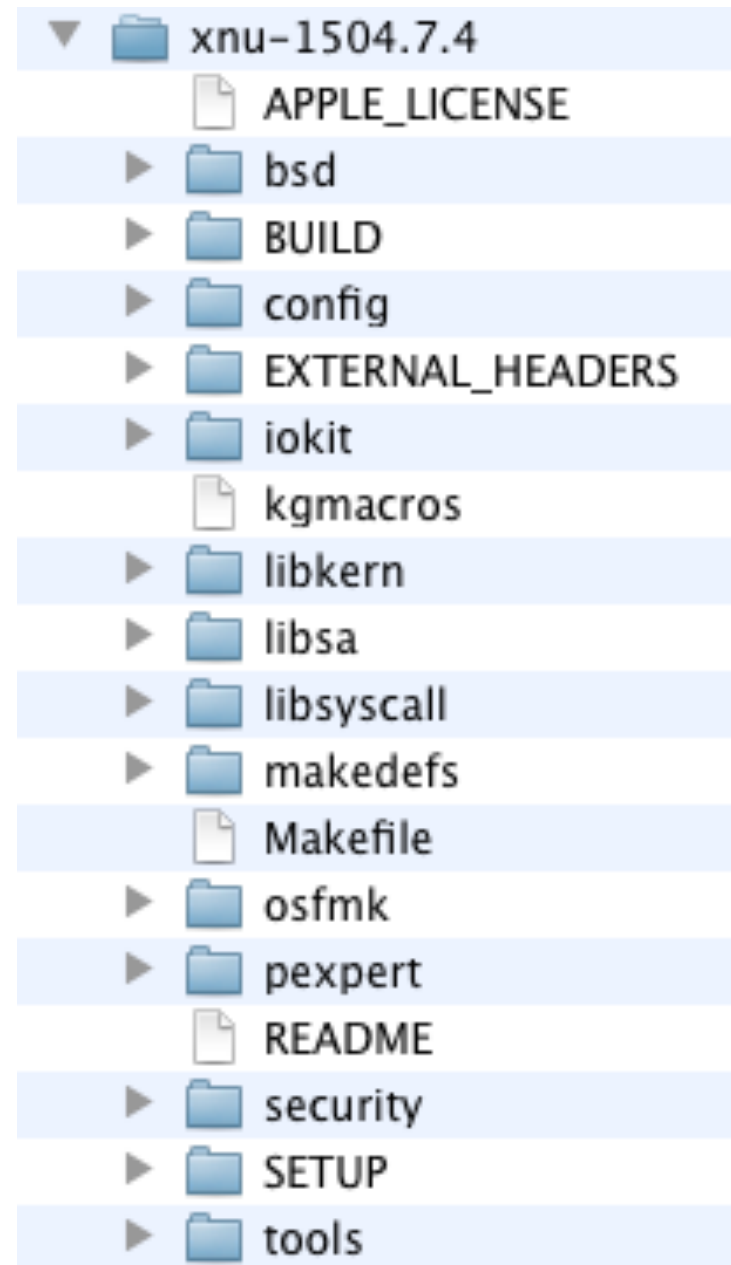
- <http://opensource.apple.com>



Mac OS X 10.6.4 Source

● Project	Licenses	Downloads	Related Sites
Apple16X50Serial-24	APSL		Open Source Development Documentation and resources for Bonjour, Java, UNIX, and WebKit.  Mac OS Forge  Open development of select Mac OS X projects
AppleAPIC-13	APSL		
● AppleDisplays-1170.0.40	APSL		
AppleFileSystemDriver-13	APSL		
AppleHWSensor-193	APSL		
AppleIntelPIIXATA-251.0.1	APSL		
AppleKeyswitch-105.3.4	APSL		
AppleRAID-4.0.6	APSL		
AppleRTL8139Ethernet-153	APSL		

xnu Source

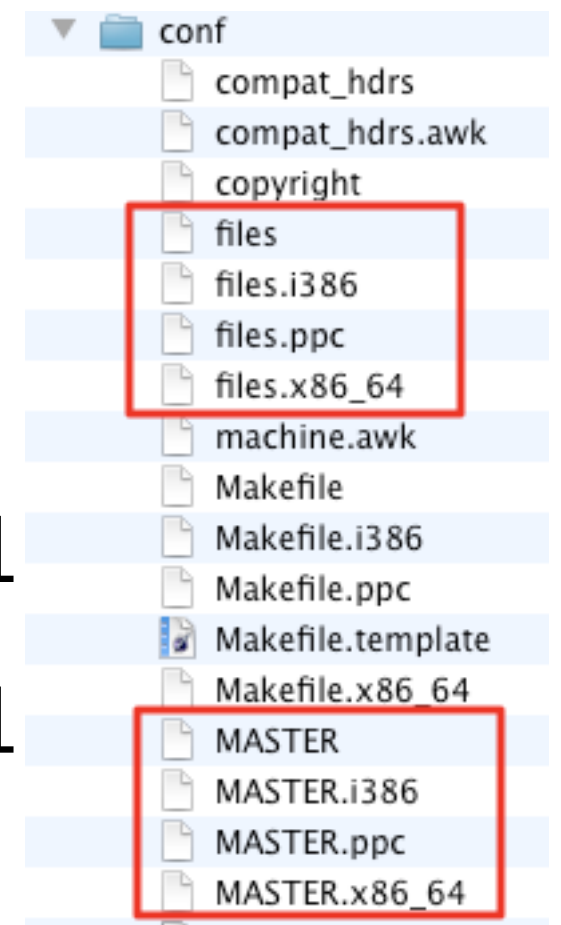


Eigene Module

- Konfigurationsdateien liegen in
 - `xnu/bsd/conf` für BSD
 - `xnu/osfmk/conf` für Mach
- `files`, `files.ppc`, `files.i386`, `files.x86_64`
- `MASTER`, `MASTER.pcc`, `MASTER.i386`, `MASTER.x86_64`

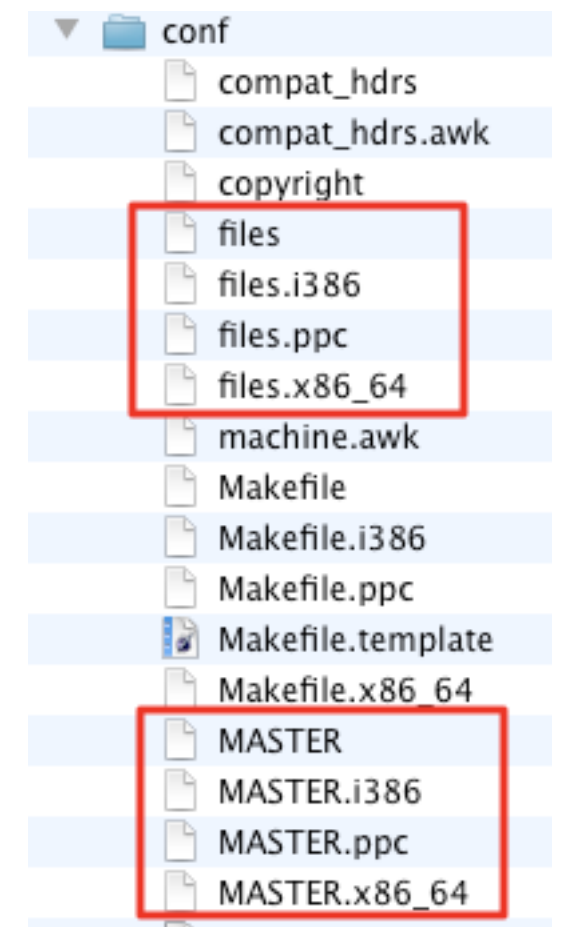
Eintragen

- Neues Modul: mach_mal
- In files
 - OPTIONS/mach_mal optional mach_mal
 - osfmk/foo/mal_main.c optional mach_mal
 - osfmk/foo/mal_code.c optional mach_mal
- Ohne modul:
 - osfmk/blah/wichtige_datei.c standard



Eintragen

- In MASTER
 - options MACH_MAL
- Für Pseudo-Device Driver
 - pseudo-device mach_mal
 - pseudo-device ppp 2



Kernel Source

- Alle .c Datei werden gebaut
- Was gebaut wird ist Verantwortlichkeit der Datei

```
#include <mach_mal.h>
```

```
#if (MACH_MAL > 0)
```

sysctl

- `SYSCTL_INT()` Macro
- Irgendwo im Kern
- In kext: `sysctl_register()`

Selber bauen

- Darwinbuild
- Wenn das nicht geht...
- xnu laden
- Tools laden (10.6.4):

cxxfilt

dtrace

kext_tools

bootstrap_cmds

cxxfilt

```
% cd cxxfilt-9
% mkdir -p obj sym dst
% make install RC_ARCHS="i386 x86_64" RC_CFLAGS="-
arch i386 -arch x86_64 -pipe" RC_OS=macos
RC_RELEASE=SnowLeopard SRCROOT=$PWD OBJROOT=$PWD/obj
SYMROOT=$PWD/sym DSTROOT=$PWD/dst
...
% sudo ditto $PWD/dst/usr/local /usr/local
```

dtrace

```
% cd dtrace-78
% mkdir -p obj sym dst
% xcodebuild install -target ctfdump -target ctfconvert -target
ctfmerge ARCHS="i386 x86_64" SRCROOT=
$PWD OBJROOT=$PWD/obj SYMROOT=$PWD/sym DSTROOT=$PWD/
dst
...
% sudo ditto $PWD/dst/usr/local /usr/local
```

kext_tools

```
% cd kext_tools-177
% mkdir -p obj sym dst
% xcodebuild install -target kextsymboltool -target
setsegname ARCHS="i386 x86_64" SRCROOT=$PWD OBJROOT=
$PWD/obj SYMROOT=$PWD/sym DSTROOT=$PWD/dst
...
% sudo ditto $PWD/dst/usr/local /usr/local
```


bootstrap_cmds

```
% cd bootstrap_cmds-72
% mkdir -p obj sym dst
% make install RC_ARCHS="i386" RC_CFLAGS="-arch i386
-pipe" RC_OS=macos RC_RELEASE=SnowLeopard SRCROOT=
$PWD OBJROOT=$PWD/obj SYMROOT=$PWD/sym DSTROOT=$PWD/
dst
...
% sudo ditto $PWD/dst/usr/local /usr/local
```

xnu

```
% make ARCH_CONFIGS="I386 X86_64" KERNEL_CONFIGS="RELEASE"
```

- DEBUG für ddb Support

Wohin jetzt?

- „lucy“, 24C3: Inside the Mac OS X Kernel
- Amit Singh: „OS X Internal“
- <http://developer.apple.com>
- darwin-kernel@lists.apple.com.

Fragen?

Vielen Dank



Macoun'10