# Chapter 1

第一章主要介绍了apple的一些过往历史，感觉对于内核部分的理解屁用没有，可以不看。

Start Trek

NuKernel

TalOS

Copland

BeOS

NEXTSTEP

然后最终决定用NeXT（OKOK，Jobs）

然后大概介绍了一下，mach内核和NEXTSTEP操作系统的变迁直到其被苹果再次收购

OpenStep API

Mach相关部分

RIG（Rochester’s Intelligent Gateway）系统和一个alpha 内核，当时就做出来mach的核心的port通信的东西。Port是一个核内的消息队列，用整数点对来identify。用一个process和一个port number

后来又去做了Accent，据说非常先进，但是没有考虑UNIX的一些特性，然而unix又大行其道，于是改成unix去了

这玩意就是mach

他最重要的是提供了四个抽象概念

Task，是一个或者多个threads的资源容器，包括虚拟内存，端口，处理器，等等其他

Thread，就是正常thread的概念。

Port

Message

Mach3做了很多改进

我看了一圈觉得最重要的是允许一部分的system call在用户空间进行处理

Mach3不再是一个真微内核

原因是一些实际的IPC的原因

MKLinux

Apple和OSF的一个合作项目 osfmk，最后的结果就是所谓的MkLinux，看这个介绍，估计是Linux的封装，里面弄了一个Mach的情况。

Rhapsody

Blue Box（macos compatibility subsystem）

Yellow Box（OpenStep API）

然后就是MacOS X，也就是各个大猫的版本

# Chapter 2

Chapter 2主要介绍了关于MACOSX的上层应用的一些情况，以及整体的框架

是一层一层堆叠起来的

一层可以包含applications, libraries, and frameworks。

多层可以包含同样名称的实体

从底向上

## 固件层（firmware）

这并不是内核的一部分

PowerPC用的是Open Firmware，而X86则是用的EFI（UEFI应该在后来变成主要的，原因在于有安全启动，显然苹果在安全上做的还是很好的，而这本书太早了）

（注：我查了QEMU里面需要加-bios选项添加uefi）

## Bootloader

在PowerPC上面是BootX

X86上叫做boot.efi

## Darwin

Darwin操作系统包含了一大堆packages，不同发行版包含的packages是不同的

对于license，有的是Apple Public Source License（APSL），有的是按照他们软件package各自的发行版的License。

Darwin并不是macos X，比如Aqua这种图形界面就不是内核中的一部分

Darwin包括了xnu和其他一些系统服务。

XNU kernel可以看做基于mach的核心，基于BSD的一个操作系统界面和一个对象为核心的运行环境和其他内核扩展。

一个基于mach 3，一个基于FreeBSD 5

XNU内核其实分为以下几个部分

* Mach
* BSD
* Libkern
* Libsa
* I/O kit
* The Platform Expert
* Kernel extensions

其中BSD占用了一半的代码量，而Mach则约占用了三分之一。剩余的是一些扩展。

使用kextstat命令，来列出所有的目前所装载的所有扩展。/System/Library/Extensions/这个文件夹下面。

## Mach

Mach是XNU的核心。

Mach负责的部分包括了

1、硬件抽象

2、处理器管理

3、多任务抢占

4、虚拟内存管理

5、低级IPC

6、实时性的支持

7、Kernel debug

8、终端I/O

虽然按照传统来说，Mach是一个微内核，但是，具体实现中，Mach和其他的比如I/O kit和BSD处于同一个地址空间

## BSD

BSD大部分跟原版一样，但是也有少部分为了能跟I/O kit和Mach兼容，从而造成了改变

BSD包括以下几个部分

BSD-style 进程模型

信号量

用户id权限

POSIX API

异步IO

BSD-style的system call

TCP/IP协议栈和BSDsocket

NKE网络核心扩展（和xnu兼容的部分）

## VFS

System V（我个人认为这里面应该指的是进程间通信的标准，包括消息队列，共享内存，信号量）和POSIX进程间通信机制

内核部分密码框架

巴拉巴拉

## UBC（unified buffer cache）统一缓存。

让文件可以和虚拟内存一样缓存在内存中

I/O Kit

这玩意是用C++写的，但是阉割了很多特性不让用，然后自己实现了一个RTTI系统

包括一个内核内的C++库，和一个用户空间的框架

I/O kit本身是模块化的以及分层的，它给捕捉，表示，维护涉及到I/O联系的不同硬件软件组件之间的关系提供了一个基础设施。

他封装了底层的抽象给系统的其他部分。

Libkern 库

这玩意就提供了前面说的I/O kit的runtime system，用C++自己实现的那个

看了一圈，就好像，实现了C++的一些特性。

Libsa 库

在系统早期启动时用于加载内核扩展

就是说，正常系统扩展是用的/user/libexec/kextd来启动的，但是启动早期，这些都没有，Libsa则提供了一个kextd的子集。

The platform Expert

IOPlatformExpertDevice是当成设备树的根

IO kit会注册一个nub到Platform Expert

Nubs

相当于IOkit注册的一个AP（access point）

Kernel extensions

就是一些扩展

User-Space View of The file system

按照unix系统的观点，一些根目录下必然有的文件夹，macos也有。

他分为四个domain：

User：就是用户文件夹

Local：对于任何一个用户都可以访问，包括公共的应用和文件，同样，典型的也是放在root卷上面的，只有系统用户才可以更改他

Network：网络域，只有网络管理员才可以。

System：属于MacOSX的资源

系统寻找资源比如说font是按照user，local，network，system四个域的顺序来进行寻找的。

Library目录是所有domain都有的目录，系统的在/system/library

该目录下有很多关键目录都各有用途，不做赘述。

Runtime Architecture

MacOS X只有一个runtime架构： Mach-O，也就是Mach Object File Format。

苹果系统只有这一种运行时。

为此，苹果实现了以下七种格式的文件：

Bundles

动态链接库

Frameworks

Umbrella Frameworks

内核扩展

链接obj

静态achives

可执行文件

除此之外需要了解以下几种用于管理MACH-O文件的工具

As

Dyld默认的dynamic link editor

Gcc g++

Ld

Libtool

Nm，一个用来显示obj符号标的工具

Otool

一个mach-o文件具有固定的头部，然后是可变大小的load 命令，然后是几个segment，每个segment可以包含一个或多个sections

头部，magic，cputype，cpusubtype，filetype，ncmds，sizeofcmds，flags

Fliletype表明了这个文件的类型

在头后面的load命令中指明了只想linker的路径，默认是/usr/lib/dyld，这也是一个mach-o文件，MH\_DYLINKER（我觉得他在说加载器，也许动态链接器，以后就这么称呼吧）

内核检查mach-o头，并且决定类型

内核翻译load命令

加载指定的动态链接器到内核

然后内核跑dynamic linker，将argc和argv传递过去。

动态链接器翻译对应的load命令，它下载依赖的共享库，