Linux初步

《Linux初步》内容

- 课程目的
- 实验安排
- 时间安排
  - 集中授课/实验
    - 第1～5周

《Linux初步》课程目的

- 熟练掌握Linux操作系统的使用
- 了解Linux操作系统的运作过程，理解内核与外围支撑系统的关系
- 通过实验定制Linux系统内核与外围支撑系统，加深对开源操作系统的认识
- 课程输出：具有各自功能特色的自启动最小系统

实验安排

1.完成外围文件系统的定制

1.理解外围应用程序对操作系统整体的支撑作用

2.完成Linux内核的定制

1.支持模块的Linux内核定制

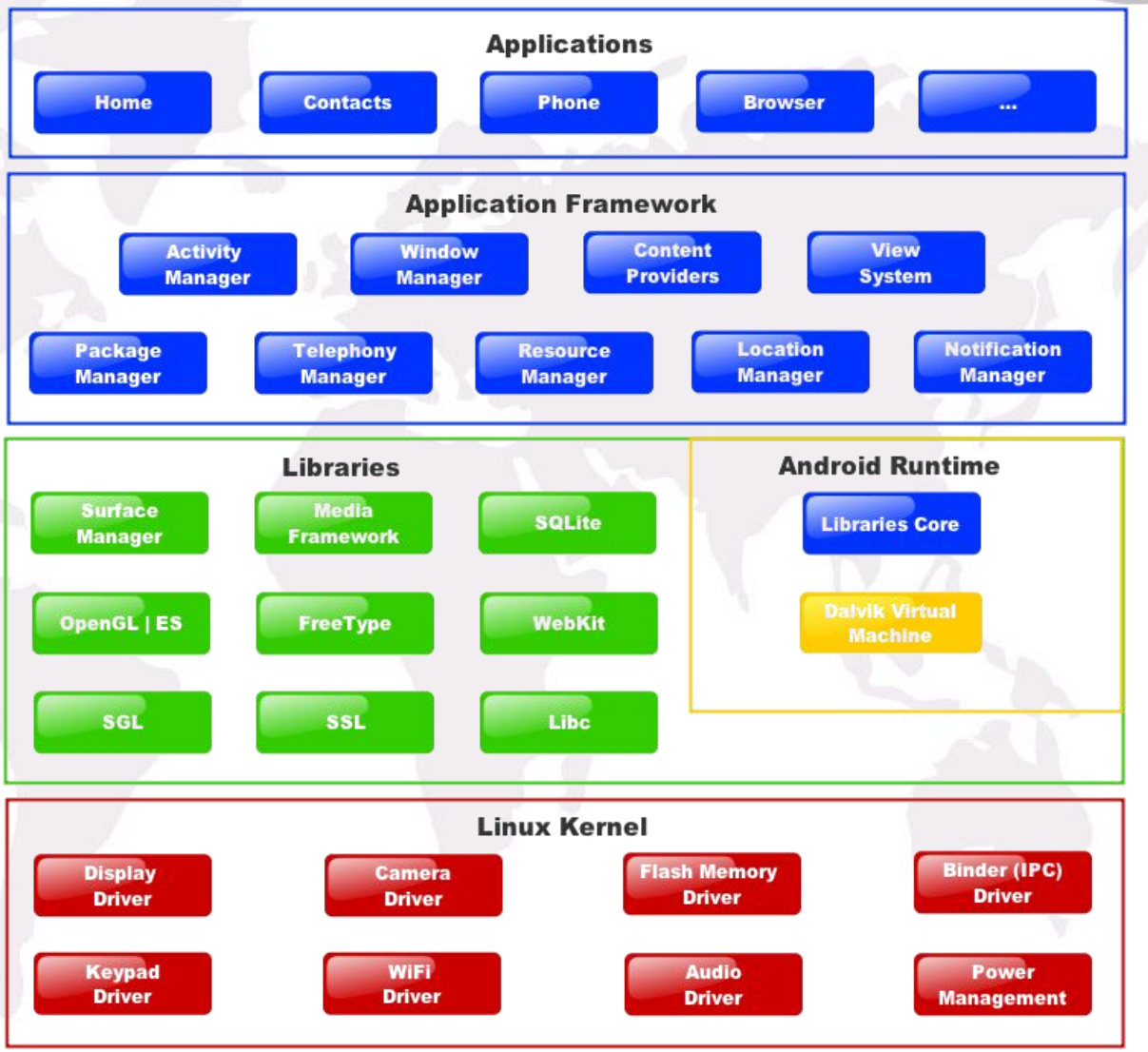
3.完成OS Loader的安装应用

课程内容（1）

- 什么是Linux
  - 由Linus Torvalds开发的操作系统？
- 什么是GNU/Linux 操作系统？
  - 以Linux内核为基础，GNU软件为支撑的类Unix操作系统
- 为什么选择学习Linux

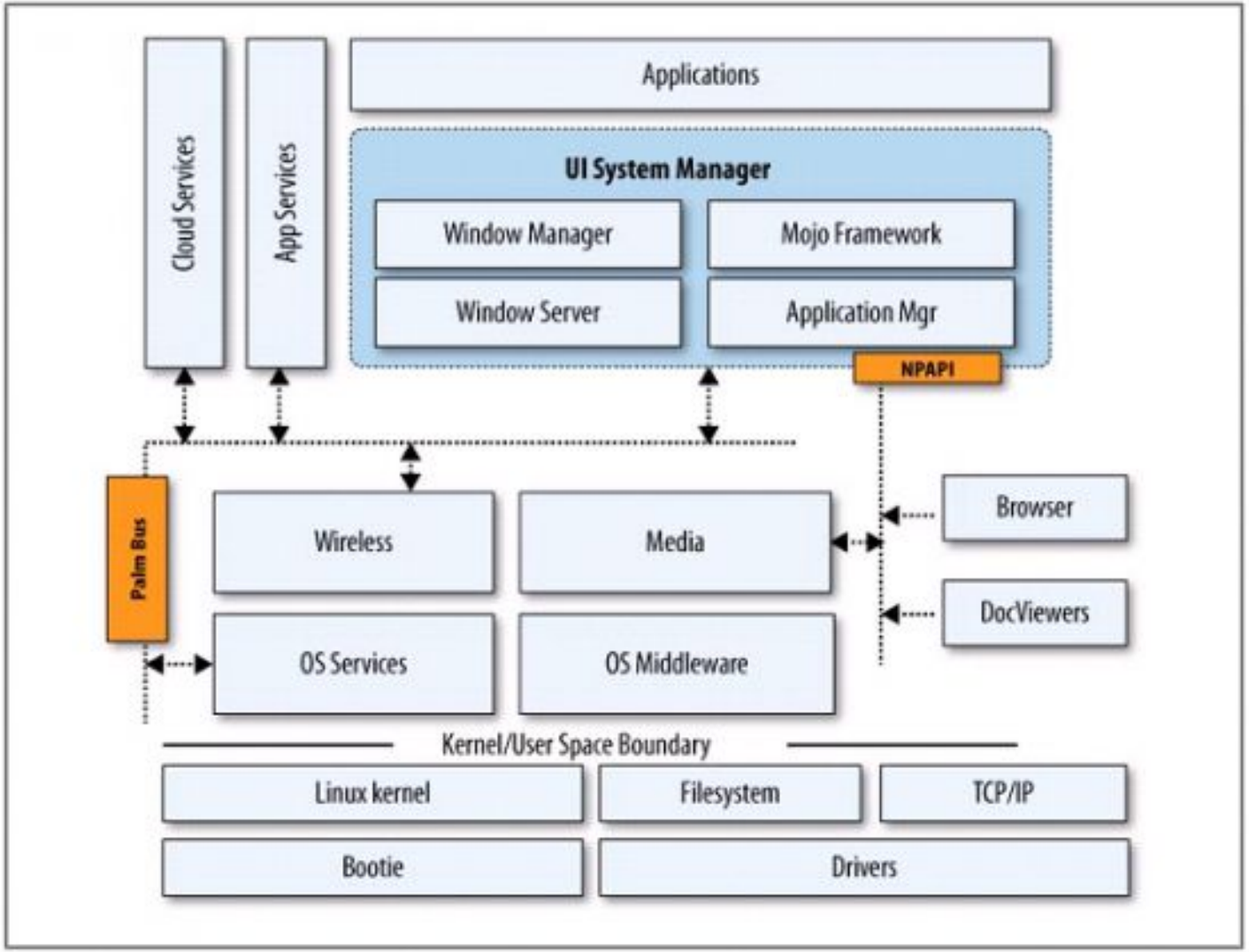
课程内容（1）

Android系统架构



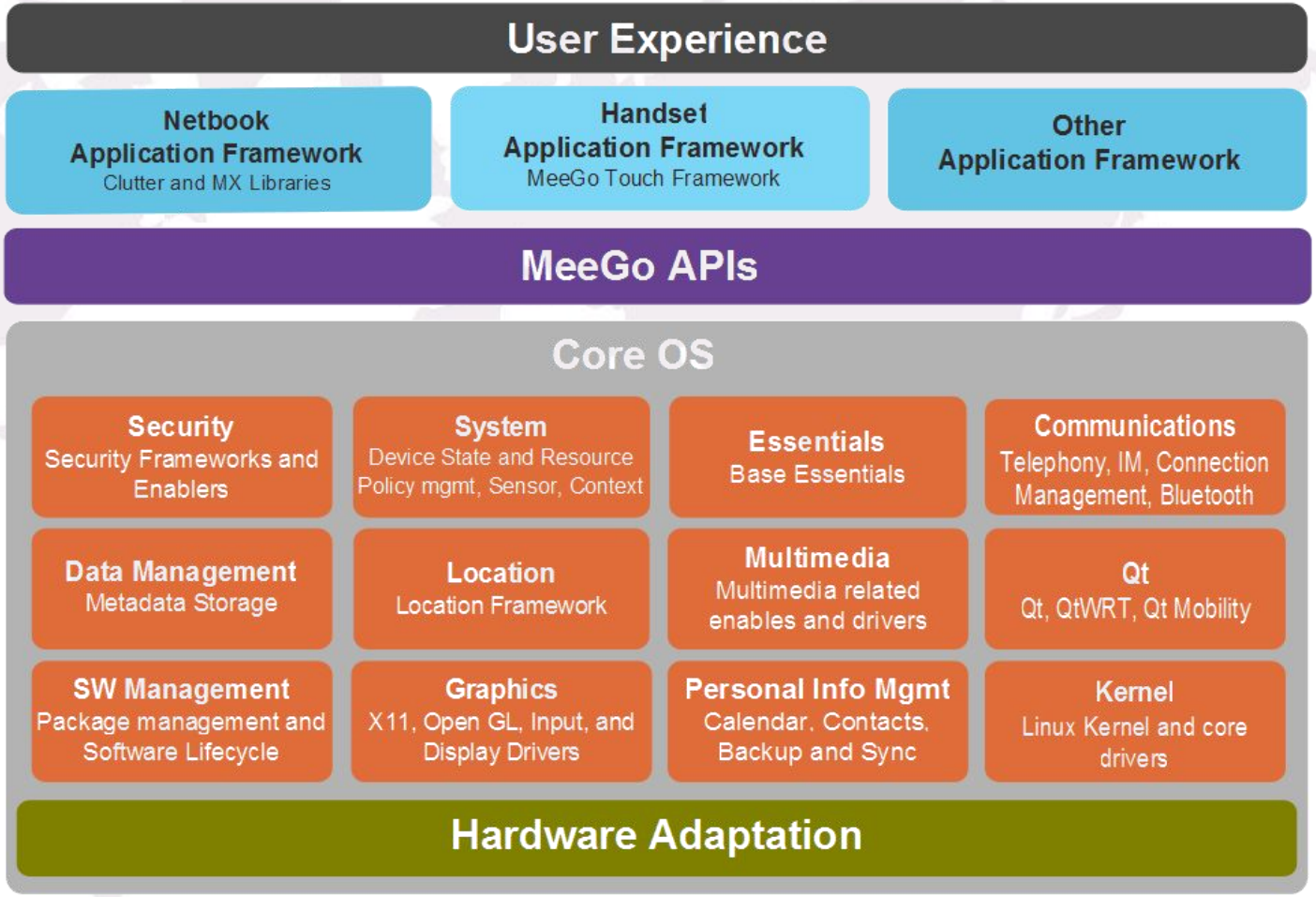
课程内容（1）

WebOS系统架构



课程内容（1）

Meego系统架构



课程内容（1）

- What happens after power on……?
  - BIOS
  - MBR/GPT
  - OS Loader
  - OS Kernel
  - Application Manager
  - Applications...

Master **B**oot **R**ecord
and Disk partitions /
Globally Unique Identifier
Partition **T**able Format

DOS/Windows/Linux相关文件对比

	DOS	Windows	Linux
BIOS	--	--	--
MBR	--	--	--
OS Loader	--	**NTLDR/BootMgr**	**GRUB/GRUB2**
OS Kernel	**IO.SYS** **MSDOS.SYS**	**ntoskrnl.exe**	**vmlinuz**
Application Manager	**command.com**	**explorer.exe**	**init**
Applications	...	...	...

实验时间(1/4)

- 目标：
  - 在VMWare当中安装Linux操作系统
    - 缺省字体建议用英文
  - 使用ssh客户端(putty)完成Linux远程登录
  - 了解并熟悉Linux基本命令行操作命令
  - 检视如下内容：
    - OS Loader位置、配置
    - 内核vmlinuz的位置
    - 应用程序管理器init的位置、配置

实验时间(2/4)

- 常用Linux命令行操作命令
  - 文件文本进程：ls、cat、cp、rm、ps、grep、mkdir、mv、less、vi、cpio、tar
  - 网络：ifconfig、ip、ssh、telnet、ftp
  - 关机启动：reboot、shutdown、init
  - 帮助：**man**
  - 编程：gcc、make、gdb
  - 软件安装升级：yum

课程内容（3/4）

- 从源码开始安装软件
  - 准备工作：必备的工具包
    - gcc、make、automake、autoconf……
  - 获得源码包并展开
    - wget、ftp、mount等命令获得源码包
    - tar、bzip2、gunzip等命令展开源码
  - 配置、编译软件
    - configuration、make
  - 安装软件
    - make install、make strip_install

需注意源码包当中
README、**INSTALL**
等文件的信息

课程内容（4/4）

- 配置、编译软件
  - 从hello.c说开去
    - gcc用法
    - make用法和依赖关系
  - 自动配置脚本 ./configure
    - 生成Makefile
  - make
    - 不同的软件包可能有不同的目标方式

```
all: hello

hello: 1.o 2.o
        gcc 1.o 2.o -o hello
1.o: 1.c
        gcc 1.c -c
2.o: 2.c
        gcc 2.c -c
```

实验小结

- VMWare
  - virtual machine
  - virtual network mode
    - bridge、host only and NAT
- Putty
  - Open source telnet/ssh client package
  - Help to connect Linux server
    - sshd server should be available

实验小结

OS Loader — GRUB(Legacy GRUB)

- 配置文件
  - /boot/grub/menu.lst
  - /boot/grub/grub.conf
- 配置项示例：

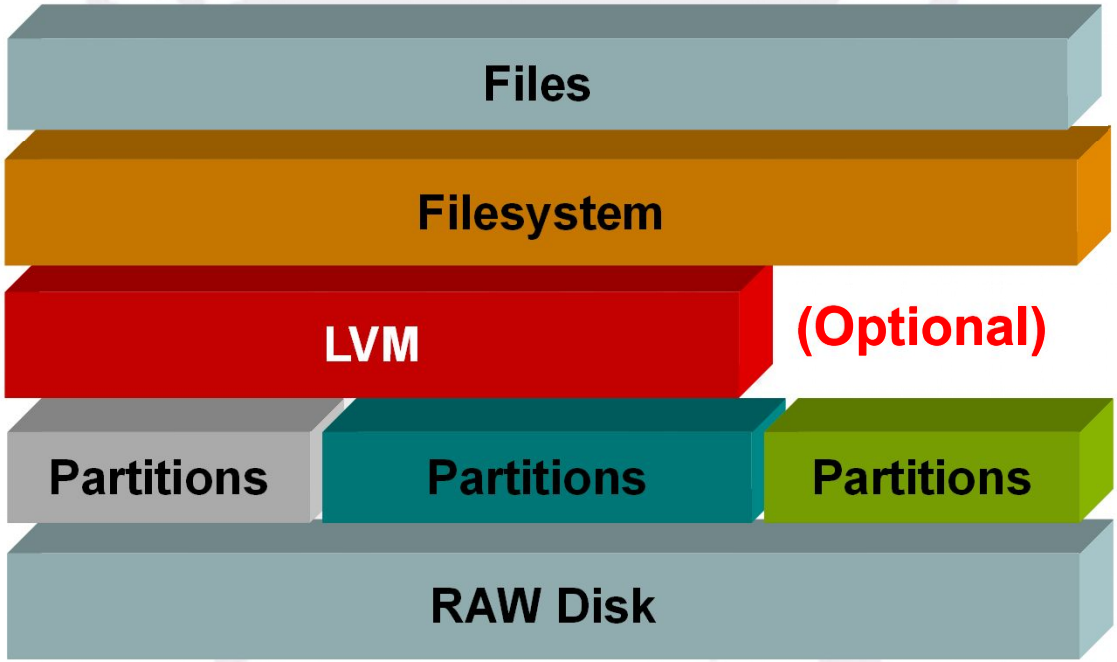
```
title Linux 2.6
        root (hd0,1)
        kernel /boot/vmlinuz root=LABEL=/
        initrd /boot/initrd.img
```

课程内容（2）

- What happens after power on……?
  - BIOS
  - MBR
  - OS Loader
  - OS Kernel
  - Application Manager
  - Applications...

课程内容（2）——Access File

- 从裸磁盘到文件的相关模块（工具）
  - 硬盘控制器
    - IDE
    - SATA(over SCSI)
  - LVM/RAID（可选）
    - device-mapper
    - 用户态工具
  - 文件系统支持
    - ext4 etc.



课程内容（2）——默默无闻的initrd.img

- 什么是initrd.img？
  - 一个由**OS Loader**载入的镜像文件
  - 一个临时的“**根文件系统**”
  - 它在哪里出现？

2.4与2.6(3.x)内核的initrd.img格式不同所致

```
title Linux 2.4
        root (hd0,1)
        kernel /boot/vmlinuz ramdisk_size=8192 root=LABEL=/
        initrd /boot/initrd.img
```

```
title Linux 2.6
        root (hd0,1)
        kernel /boot/vmlinuz root=LABEL=/
        initrd /boot/initrd.img
```

课程内容（2）——理解initrd.img

- initrd.img的作用
  - 作为临时“根文件系统”的载体（RAM Disk）
  - 加载必要的驱动以便内核可以访问“真正的根文件系统”
    - 磁盘控制器驱动
    - 文件系统驱动（ext3、ext4等等）
  - 挂载“真正的根文件系统”
  - 进行“根切换”操作，用“真正的根文件系统”作为根启动

课程内容（2）——理解initrd.img

- initrd.img的格式
  - initrd格式（**/linuxrc**，2.4内核格式，2.6可兼容）
    - 缺点：一旦创建大小即固定

```
dd if=/dev/zero of=filename bs=1M count=4
mkfs.ext2 filename
mount filename /mnt -o loop
# do anything to copy file into /mnt
umount /mnt
cat filename | gzip > initrd.img
```

课程内容（2）——理解initrd.img

- initrd.img的格式
  - initramfs格式（**/init**，2.6内核格式，2.4内核不适用）
    - 优点：大小自动随内容扩展
    - 缺点：必须自己完成根切换操作

```
mkdir tmpdir
# do anything to copy file into tmpdir
cd tmpdir
find . | cpio -H newc -o | gzip > /boot/initrd.img
```

创建方式知道了，那如何查看/解包已有的initrd.img？

课程内容（2）

- What happens after power on……?
  - BIOS
  - MBR
  - OS Loader
  - OS Kernel
  - **Initrd.img**
  - Application Manager
  - Applications...

Application Manager——/sbin/init

- 配置文件
  - /etc/inittab　　(sysvinit package)
  - /etc/init/*.conf　　(upstart package)
- 配置项示例（以sysvinit为例）：
  - 以**runlevel**运行级启动服务或脚本

```
id:3:initdefault:
si::sysinit:/etc/rc.d/rc.sysinit
l3:3:wait:/etc/rc.d/rc 3
1:2345:respawn:/sbin/mingetty tty1
```

Upstart /sbin/init 解析

- 以**事件驱动方式**启动服务或者脚本
- 配置文件放置在**/etc/init/*.conf**
- 系统事件包括startup、starting、started、stopping、stopped等等
- 例

```
start on startup
task
exec hostname -b -F /etc/hostname
```

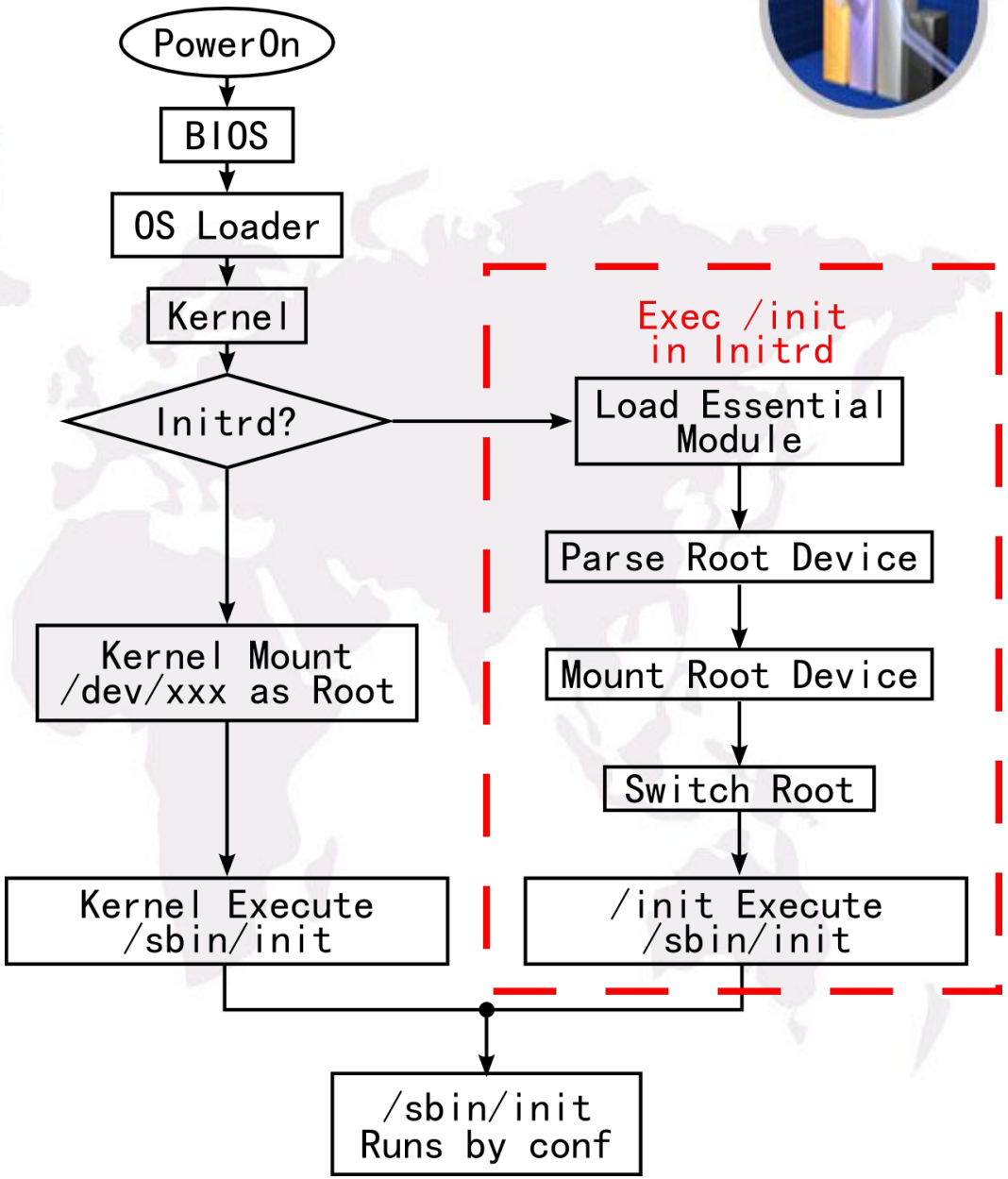
- 查询思路：
  - man init -> **FILES**、**SEE ALSO** 章节

实验返工时间

- 重新安装Linux系统？
  - 使用基础的磁盘分区而不是LVM安装Linux
  - 使启动默认状态为文本界面而非图形界面
- 对照系统的配置文件
  - 检查OS Loader设置
  - 查看系统启动时使用的initrd.img的内容
  - 检查init设置

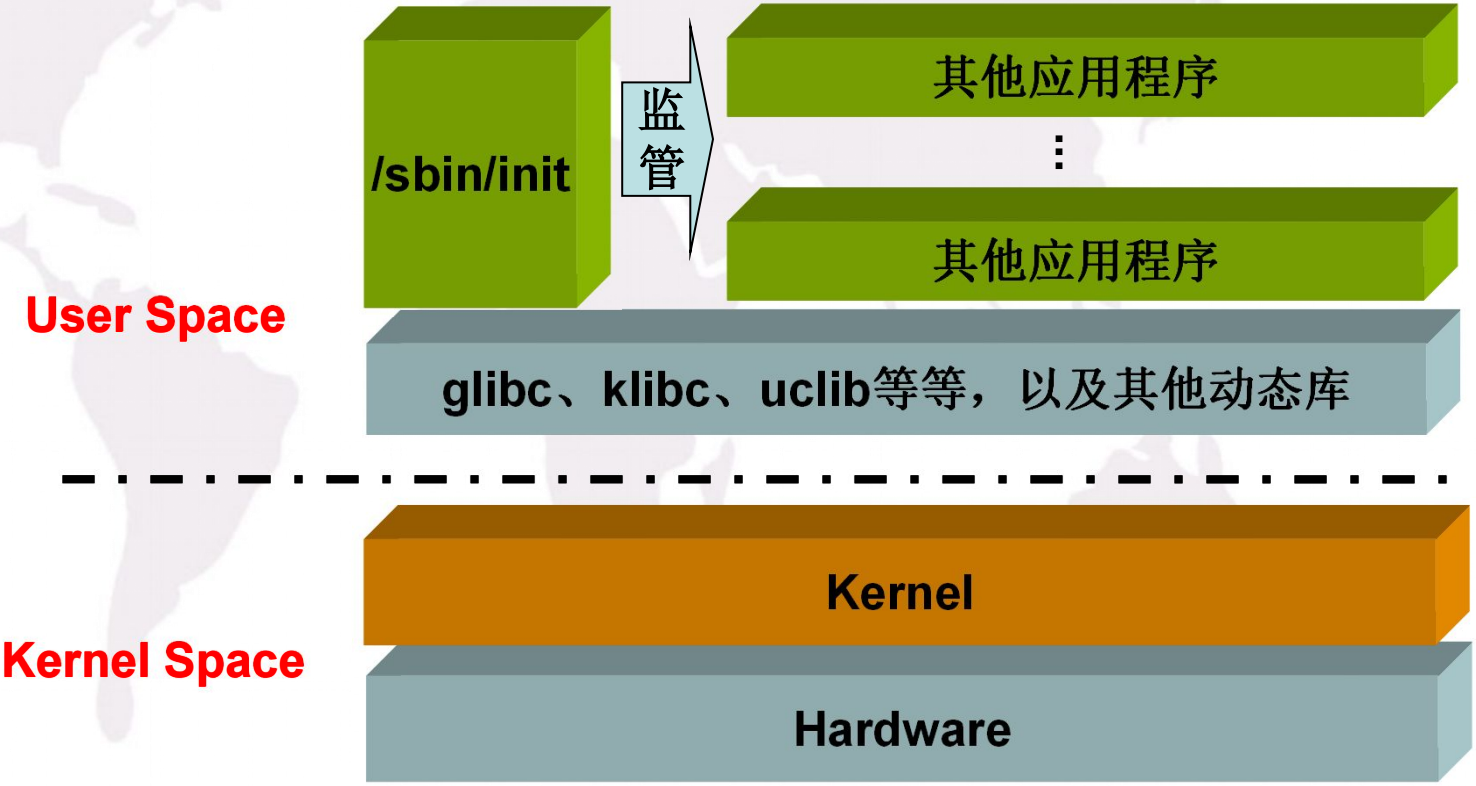
课程内容（3）

- 回顾启动过程



课程内容（3）

- 回顾启动过程
  - 最终运行形态



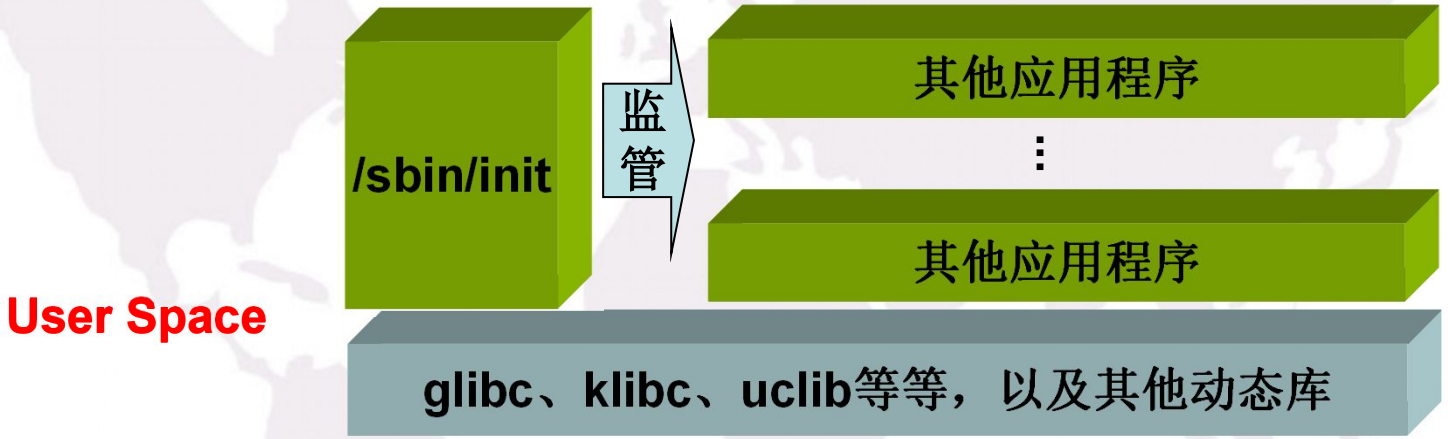
课程内容（3）

- 定制文件系统的思路（三种方式）：
  - 在原有系统的基础之上删减
  - 利用原有系统复制必备部件到新存储器
  - 利用initrd.img机制在RAM Disk中测试

优缺点各是什么？

课程内容（3）

- 构建全新根文件系统



- 面临的难题
  1. 应用程序及其依赖的动态库查询
  2. 应用程序及其依赖的配置文件

课程内容（3）

- 应用程序及其依赖的动态库查询
  - ldd命令查询
  - chroot命令辅助验证新环境下是否完备
- 应用程序及其依赖的配置文件
  - man命令查询 FILES 章节
  - strace命令获得应用程序的调用记录

实验时间

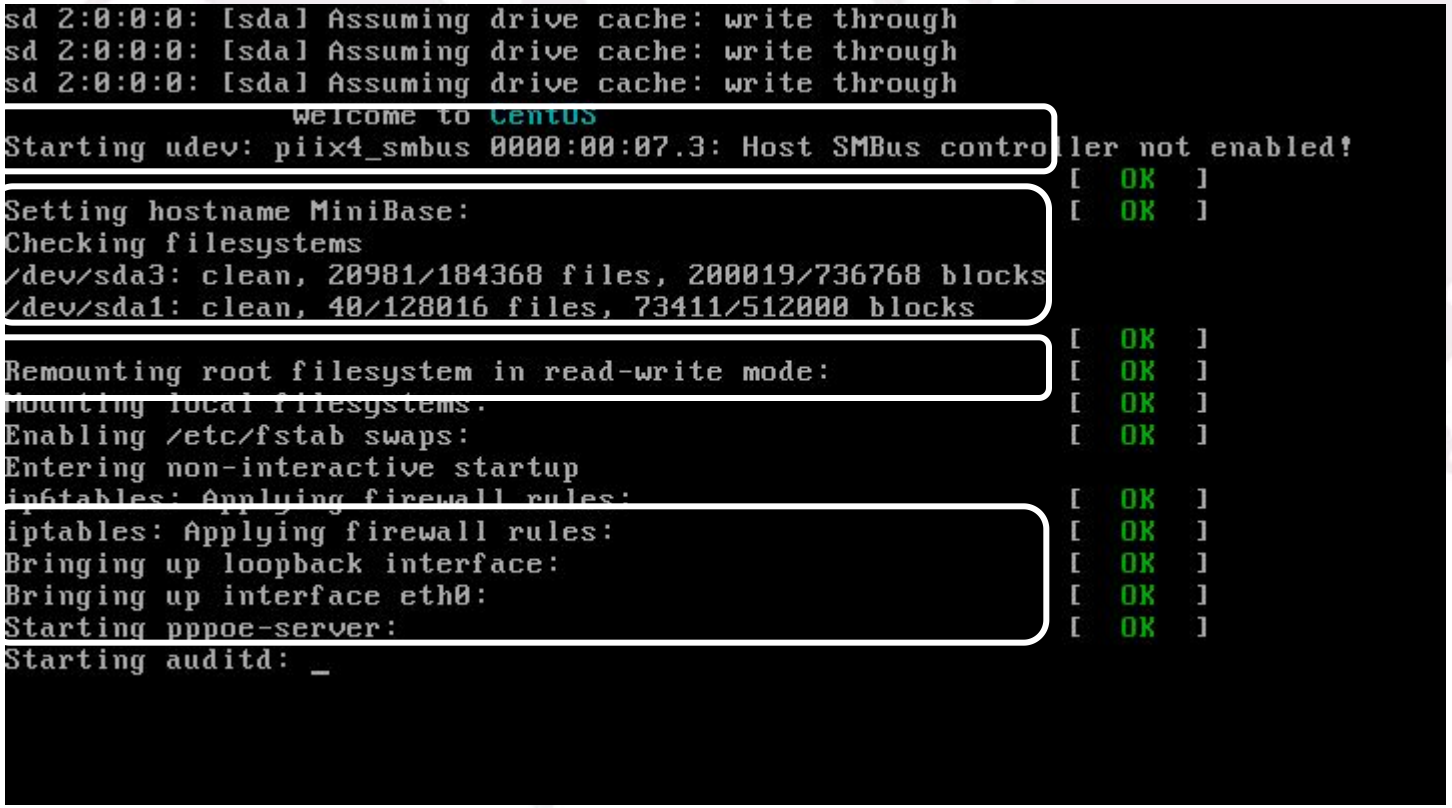
- 利用initrd.img机制，建立一个简单文件系统（v0.5版本），使得内核用该文件系统启动后可以直接获得一个shell
- 在grub启动配置文件当中增加一个入口用于测试新建的initrd.img
- 整个文件系统在启动后运行在内存中，不需要调用硬盘资源。

课程内容（4）

- 当Kernel初始化之后……
  - 不仅仅是initrd.img和/sbin/init
  - 从/sbin/init到login prompt(/etc/inittab, etc.)
    - fsck
    - mount /proc, /sys等必要文件系统
    - remount rootfs to readwrite mode
    - probe all hardware and load modules
    - start service, including network startup
    - start up getty etc. for login session

课程内容（4）

- 从/sbin/init到login prompt



probe dev

fsck

remount

services

课程内容（4）

- 从/sbin/init到login prompt
  - probe devices: udevd
  - fsck
  - remount
  - service
  - login prompt
    - mingetty + /bin/login

课程内容（4）

- udevd——管理、监控主机设备的服务程序
  - 依赖于sysfs文件系统（挂载于/sys目录下）
  - 规则文件/lib/udev
  - 配置文件/etc/udev
  - 自动在/dev目录下创建设备节点
    - 普通linux环境/dev目录是tmpfs虚拟磁盘文件系统
    - 设备节点可隶属于不同的用户和组
      - 由/etc/group、/etc/passwd指派uid、gid

课程内容（4）

- udevd——管理、监控主机设备的服务程序

```
bash-4.1# start_udev
/etc/init.d/functions: line 55: fstab-decode: command not found
Starting udev: /bin/chown: invalid user: 'root:disk'
/bin/chown: invalid user: 'root:disk'
/bin/chown: invalid user: 'root:disk'
/bin/chown: invalid user: 'root:disk'
/bin/chown: invalid user: 'root:disk'
/bin/chown: invalid user: 'root:disk'
/bin/chown: invalid user: 'root:disk'
/bin/chown: invalid user: 'root:disk'
/bin/chown: invalid user: 'root:lp'
/bin/chown: invalid user: 'root:lp'
/bin/chown: invalid user: 'root:lp'
/bin/chown: invalid user: 'root:lp'
udevd[92]: specified group 'dialout' unknown

udevd[92]: specified group 'disk' unknown

udevd[92]: specified group 'floppy' unknown

udevd[92]: specified user 'vcsa' unknown

udevd[92]: specified group 'tty' unknown

udevd[92]: specified group 'kmem' unknown
```

缺乏/lib/libnss_*
由/etc/nsswitch.conf
配置

课程内容（4）

- login prompt
  - 为什么不从mingetty入手？
    - strace -o log /sbin/mingetty /dev/tty1
    - 复杂的进程管理机制和名词（进程组、控制台等）
    - 只能由/sbin/init调用
  - login程序背后的配置
    - 认证体系（PAM）
      - /etc/pam.d的配置
      - /lib/security及其依赖的库文件

课程内容（4）

- 从/sbin/init到login prompt
  - probe devices: udevd
  - fsck
  - remount
  - service
  - login prompt
    - mingetty + /bin/login

1、谁把它们串起来？
2、它们分别在哪里被调用？

/sbin/init

课程内容（4）

- 从/sbin/init到login prompt
  - probe devices: udevd
  - fsck
  - remount
  - service
  - login prompt
    - mingetty + /bin/login

/etc/rc.sysinit

/etc/rc

/sbin/mingetty

又是通过什么方式把它们串起来的？

实验路线

- 在获得shell版本的initrd.img基础上（v0.5）
  - 完成拥有可以挂载原系统能力的v0.55
  - 完成拥有管理设备能力（udev）的v0.6
  - 完成拥有login登录能力（多窗口）的v0.7
  - 达到由/sbin/init管理的小系统原型v0.9

实验风险

- /sbin/init只能以进程号（pid）1进行启动
  - pid=1则为kernel初始化后的第一个进程
  - 实现方式：
    - initrd.img当中的/init以根切换并运行init的方式执行
    - initrd.img当中的/init指向/sbin/init
    - initrd.img当中的/init用/sbin/init替换自身
      - 在/init脚本当中执行 exec /sbin/init

实验风险

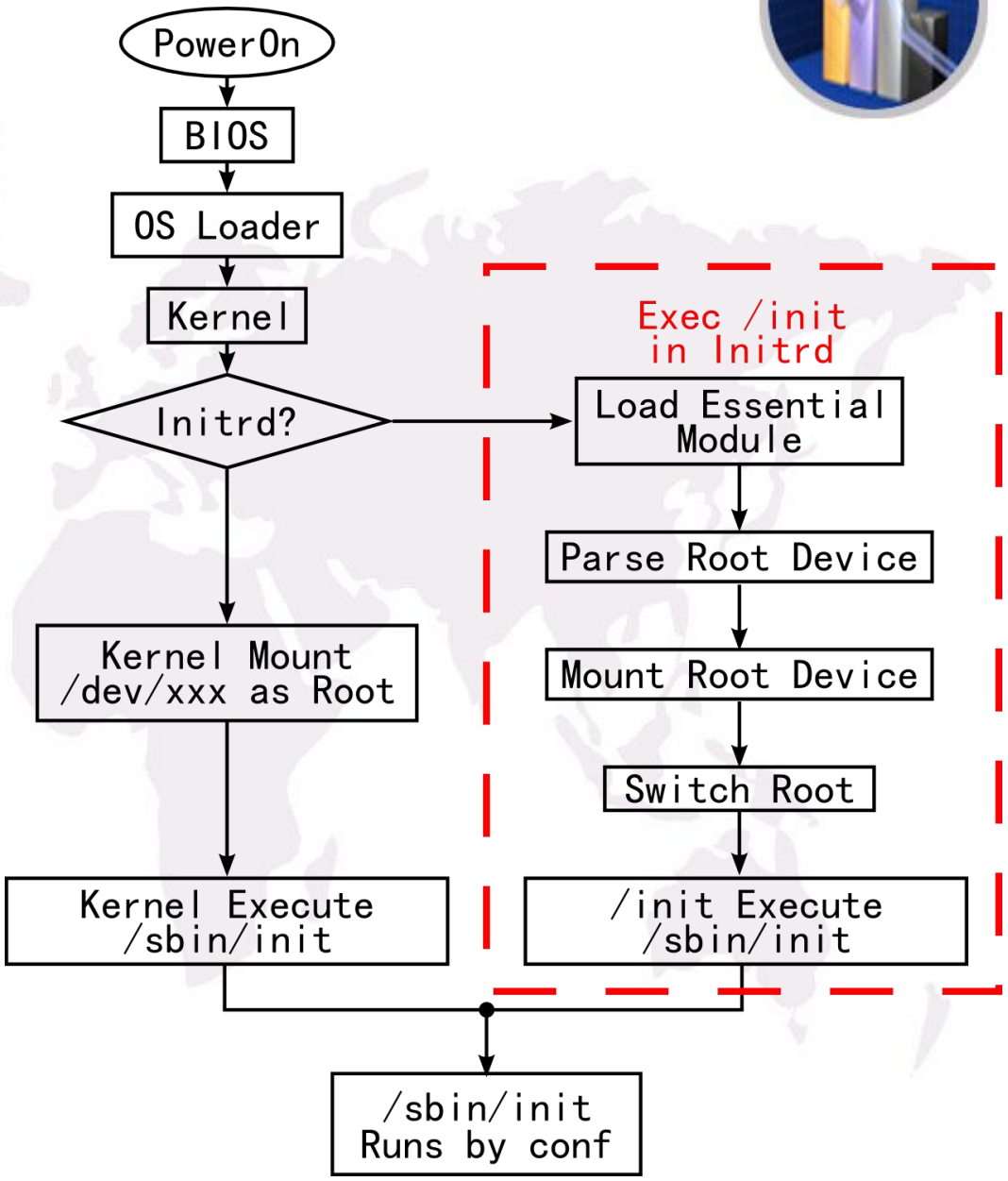
- /sbin/init只能以进程号（pid）1进行启动
  - 带来的问题：
    - 实验可调试性大大降低，需仔细阅读文档
    - 应用屏幕录像等方式辅助定位问题

实验时间

- 利用initrd.img机制，建立一个简单文件系统。该文件系统通过/sbin/init完成应用程序管理，并可让用户登录。

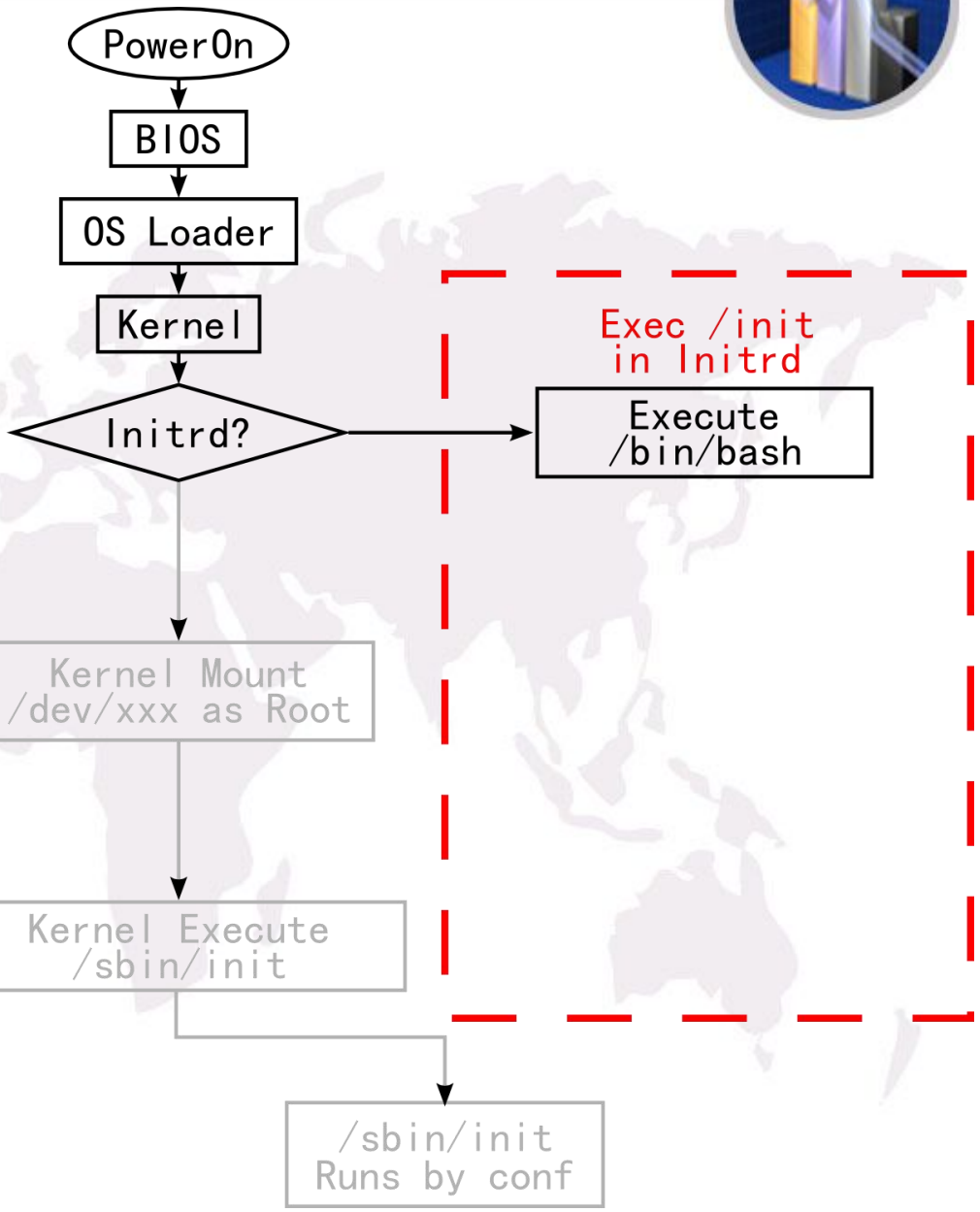
课程内容（5）

- 回顾启动过程



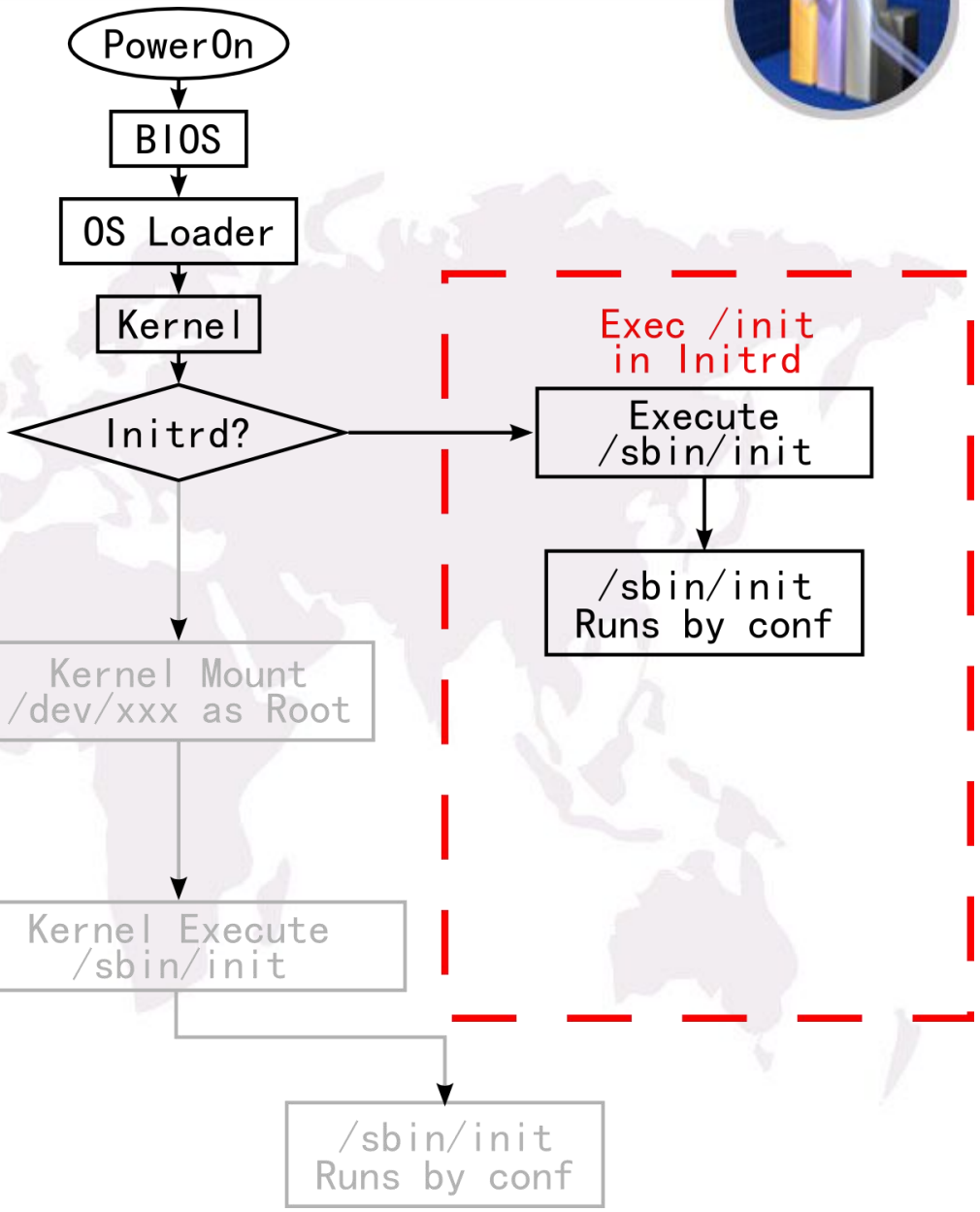
课程内容（5）

- 实验v0.5效果



课程内容（5）

- 实验v0.9效果



课程内容（5）

- 当文件系统的定制完成
  - 理解Linux系统外围支撑环境的组成
  - 获得一不影响原系统的测试环境
- 新的征程
  - 定制内核

课程内容（5）

- Linux内核
  - 由内核（kernel）与模块（module）共同组成
  - 其中
    - kernel在系统启动时由OS Loader加载
    - module（可选）在kernel初始化后由模块工具加载
      - 模块负责提供内核级的功能
      - 类似于windows系统上的驱动

课程内容（5）

- Linux内核的配置
  - 下载源码包后解开
  - 选项配置命令
    - make config (交互式问答)
    - make menuconfig （文本界面)
    - make xconfig （图形界面)
  - 内核源码配置的其他命令格式
    - make localmodconfig
    - make localyesconfig

课程内容（5）

- 内核的编译
  - make，大约需要20～40分钟
- 新内核的安装
  - 内核文件bzImage
    - OS Loader可识别的任意位置
  - 模块文件
    - make modules_install
    - 安装的位置？

实验时间(1/2)

- 下载最新的稳定版Linux kernel src
- 展开并检视所有的内核配置项
- 对照windows硬件设备管理器检视设备配置
  - CPU
  - 硬盘控制器
  - 网络控制器
  - USB控制器
    - HID、Mass storage
  - 声卡控制器（可选）

实验时间(2/2)

- 开始着手定制Linux内核
  - 内核？模块？
- 测试定制的内核？
  - 通过已定制完成的initrd.img文件系统作验证
  - 让原有的centos系统使用定制的内核

课程内容（6）

- 安装grub
  - 向目标分区/boot/grub下复制stage1、stage2
  - 在grub shell当中执行
    - root (hdX,Y)
    - setup(hdX)
      - setup (hdX)与setup(hdX,Y)的差异

实验时间

- 在U盘上安装grub
- 并将实验1的initrd.img文件系统与实验2生成的内核文件配合，在真实的计算机上启动

最终目标

- 完成最新版本Linux kernel内核及其配套的RAMDisk文件系统定制工作
- 要求：内核文件<4M，initrd.img < 24M
- 功能要求：
  - 通过U盘加载kernel和img启动进行验证
  - 支持多用户登录（console界面和ssh网络方式）
  - 系统支持通过ssh方式访问其他机器
  - 可挂载U盘
  - 可访问机器上的windows分区（ntfs-3g fs支持）