西南交通大學

嵌入式系统及应用

西南交通大学

电气工程学院

孙永奎博士

E-mail: yksun@swjtu.edu.cn

第8章 嵌入式linux程序开发

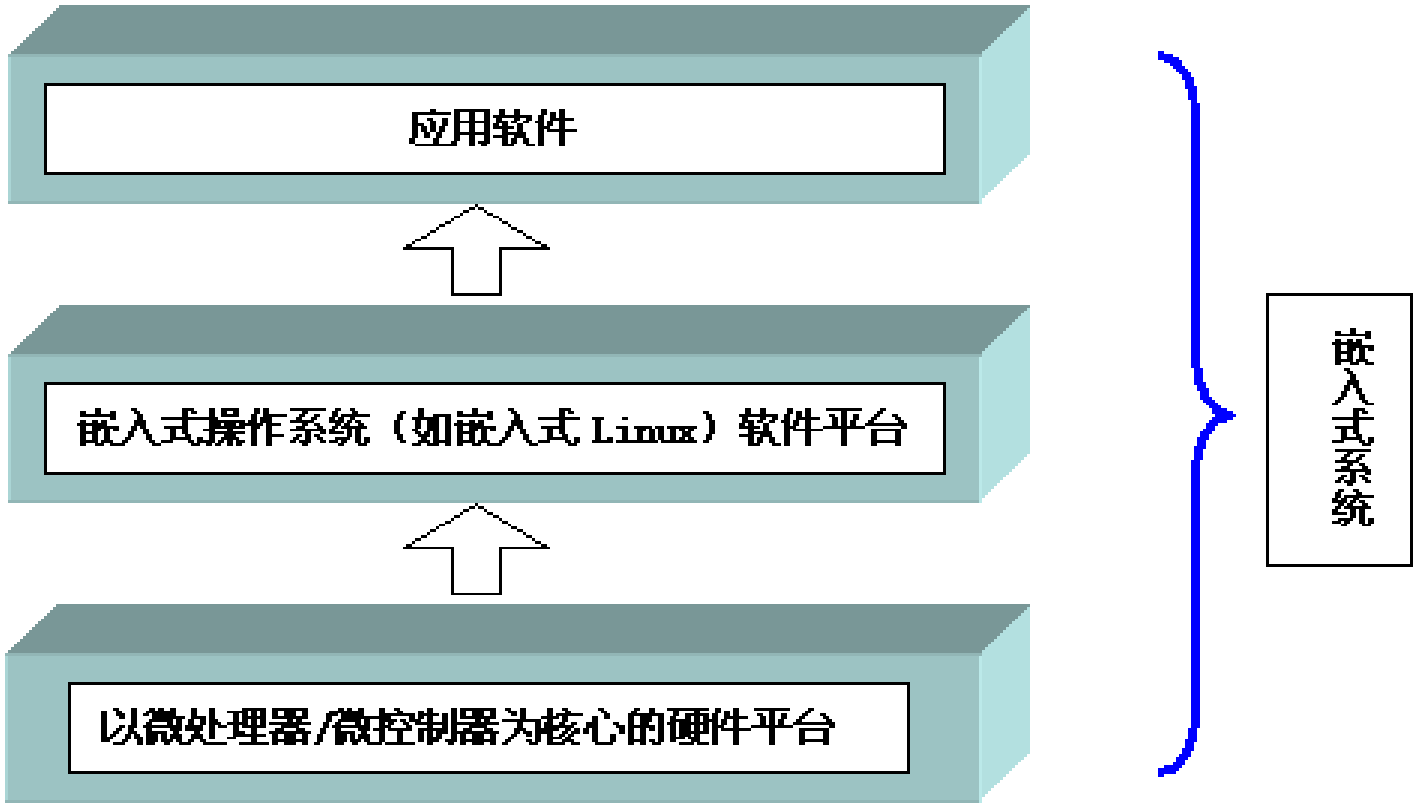
- 嵌入式linux的学习方法(中级,实践)
  - 1.嵌入式调试流程,
    - 1) 应用程序交叉编译,下载调试
    - 2) 配置Bootloader,linux内核,文件系统以及下载调试
  - 2.嵌入式应用程序开发,图形用户界面开发,简单的驱动程序开发
- 嵌入式linux的学习方法(高级,实践)
  - 1.复杂的驱动程序开发,(比如USB,编解码器外部硬件的控制等)
  - 2.文件系统的移植(jffs2,yaffs), 内核的移植,Bootloader的移植

第8章 嵌入式linux程序开发

- 在大型嵌入式应用系统中，为了使嵌入式开发更加方便、快捷，需要具备一种稳定、安全的软件模块集合，用以管理存储器分配、中断处理、任务间通信和定时器响应，以及提供多任务处理等，这就是**嵌入式操作系统**。
- **Linux本身所具备的源码开放、内核可裁减等种种特性使其成为嵌入式开发的首选。**在进入市场的前两年中，嵌入式Linux的设计通过广泛应用而获得了巨大成功。随着嵌入式Linux技术的成熟，定制需要的尺寸更加方便，同时支持更多的平台。

第8章 嵌入式linux程序开发

❖嵌入系统的构成



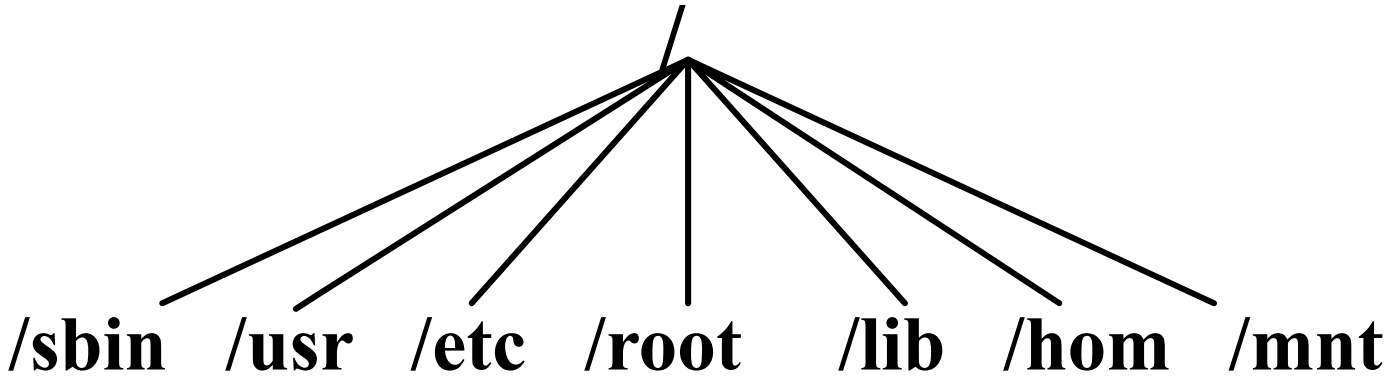
第8章 嵌入式linux程序开发

- 8.1 嵌入式linux的组成
- 8.2 嵌入式linux交叉开发环境
- 8.3 嵌入式linux驱动程序设计
- 3学时

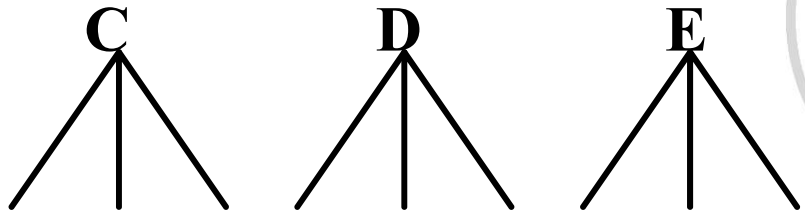
8.1 嵌入式linux的组成

❖1、linux的目录结构

- **Linux文件系统是一个目录树结构，最上层是根目录，其他的所有目录都是从根目录出发而生成的**

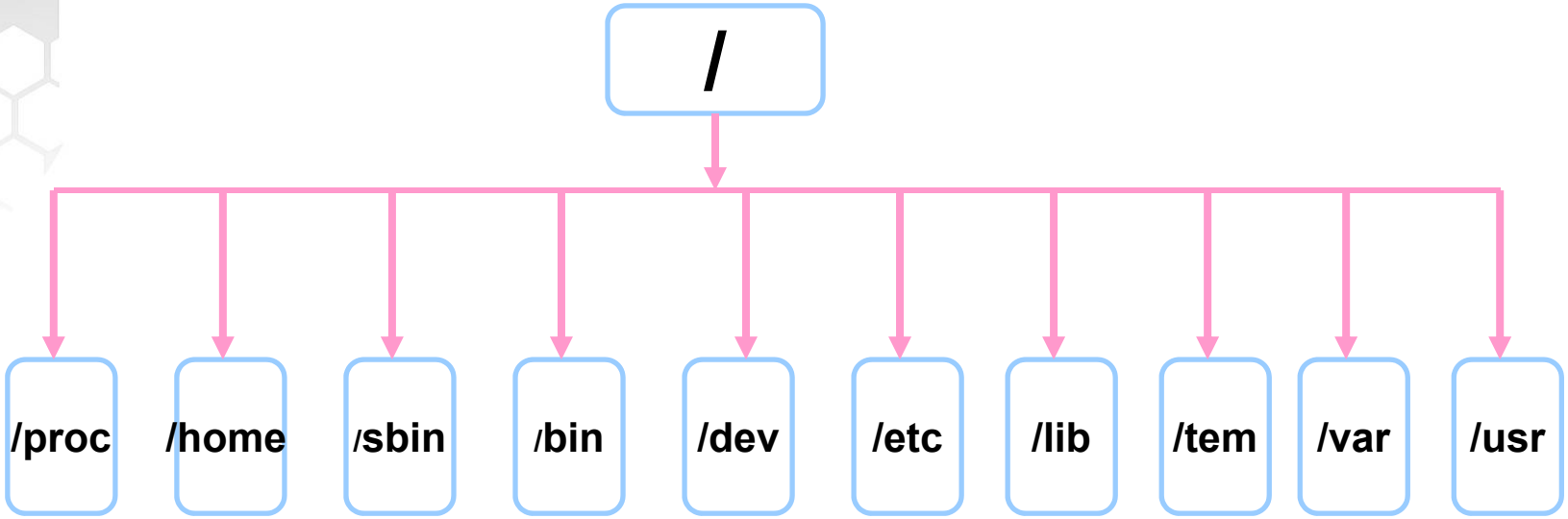


- **Dos文件系统也采用目录树的结构，但DOS以每个分区为树根，有几个分区就有几个树型结构 。**



8.1 嵌入式linux的组成

- uClinux生成的目录结构



- “/home”目录。 包含用户的文件：参数设置文件、个性化文件、文档、数据、EMAIL、缓存数据等。这个目录在系统升级时应该保留。

8.1 嵌入式linux的组成

- “/mnt”目录：　一般用来临时挂载别的文件系统，如FAT32，NTFS等
- “/bin”目录：　系统启动时需要的执行文件（二进制），这些文件可以被普通用户使用。
- “/sbin”目录：　系统执行文件（二进制），这些文件不打算被普通用户使用。（普通用户仍然可以使用它们，但要指定目录。）
- “/etc”目录：　操作系统的配置文件目录。

8.1 嵌入式linux的组成

- “/root”目录： 系统管理员（也叫超级用户或根用户）的Home目录。
- “/dev”目录： 设备文件目录。LINUX下设备被当成文件，正常情况下，设备会有一个独立的子目录。这些设备的内容会出现在独立的子目录下。
- “/lib”目录：根文件系统目录下程序和核心模块的共享库。
- “/boot”目录：用于启动加载程序的文件。当计算机启动时（如果有多个操作系统，有可能允许你选择启动哪一个操作系统），这些文件首先被装载。

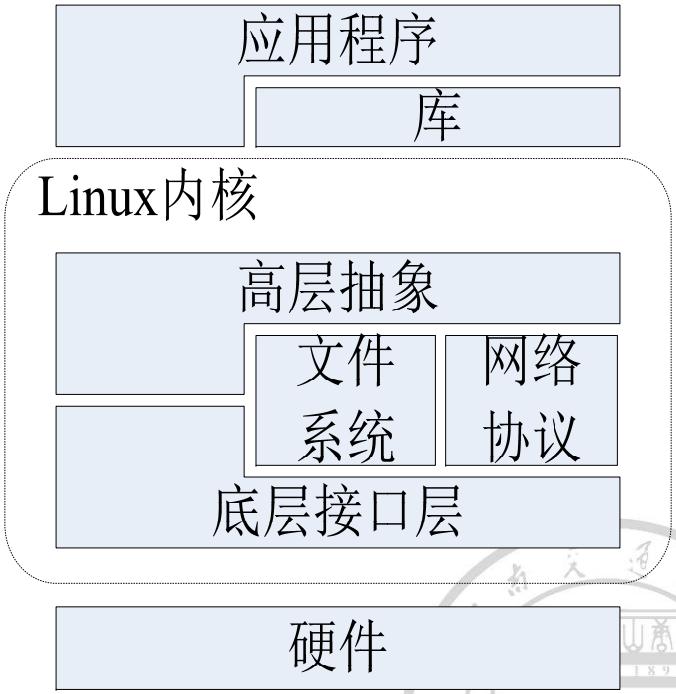
8.1 嵌入式linux的组成

- “/opt”目录：　可选的应用程序。
- “/tmp”目录：　临时文件。该目录会被自动清理干净。
- “/lost+found”目录：　在文件系统修复时恢复的文件。

8.1 嵌入式linux的组成

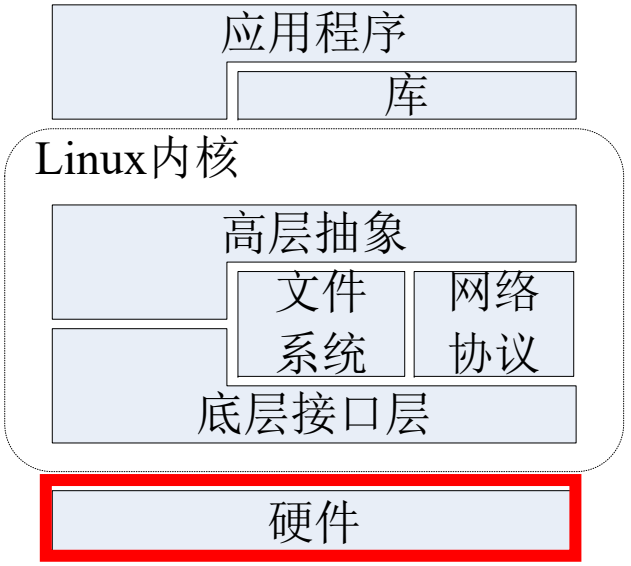
❖2、嵌入式Linux系统的一般架构

- （1）硬件
- （2）内核
- （3）文件系统等
- （4）应用程序/库



8.1 嵌入式linux的组成

- （1）硬件——一些要求方能执行Linux系统。
  - 至少32位CPU
  - 一般情况下必须配备MMU（对于不配备MMU的考虑使用uClinux）
  - RAM容量必须满足系统的需要
  - 一些最起码的I/O能力，以便在线调试
  - 具有某种形式的永久性或网络存储设备以便内核加载及（或）存取根文件系统



8.1 嵌入式linux的组成

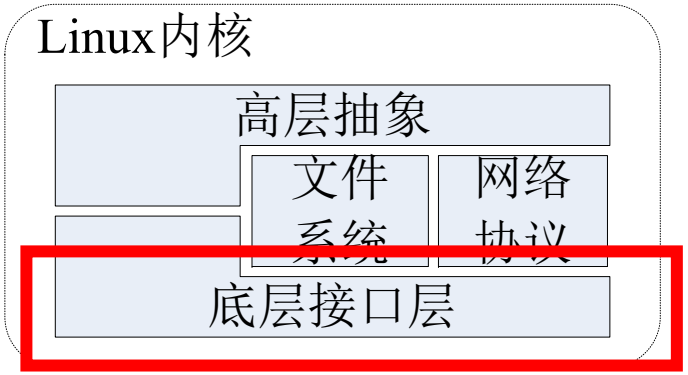
（2）Linux内核

- Linux内核是Linux操作系统的中心组件。使用内核的目的是希望以一致的方式管理硬件，以及为用户软件提供高层抽象层。

  内核大致可以分成两个部分：底层接口层和高层抽象层

8.1 嵌入式linux的组成

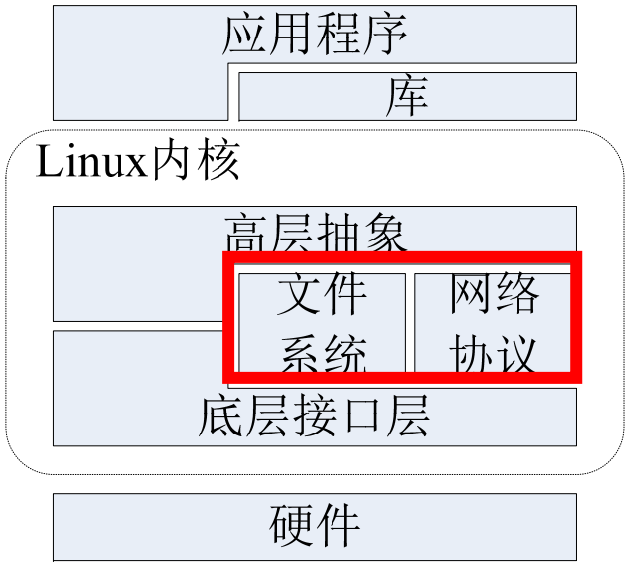
- 底层接口层专属于硬件配置，内核运行其上，并以硬件无关的高层抽象层提供对硬件资源的直接控制。
  - 比如，对于PC 机系统，尽管其寄存器或内存分页的处理方式不同，但却可以使用通用的API来存取内核里高层的组件
- 通常底层部分会处理CPU特有的操作、架构特有的内存操作以及设备的基本I/O



8.1 嵌入式linux的组成

❖ （3）文件系统和网络协议

- ❖ 在Linux内核的底层接口层与高层抽象层之间，内核有时会用到与特定设备上的结构化数据交互的组件，例如文件系统和网络协议。
- ❖ 通常，Linux内核至少需要一个具有合适结构的根文件系统。Linux内核会从中加载第一个应用程序、加载模块并为进程提供工作目录。



8.1 嵌入式linux的组成

（4）应用程序/库

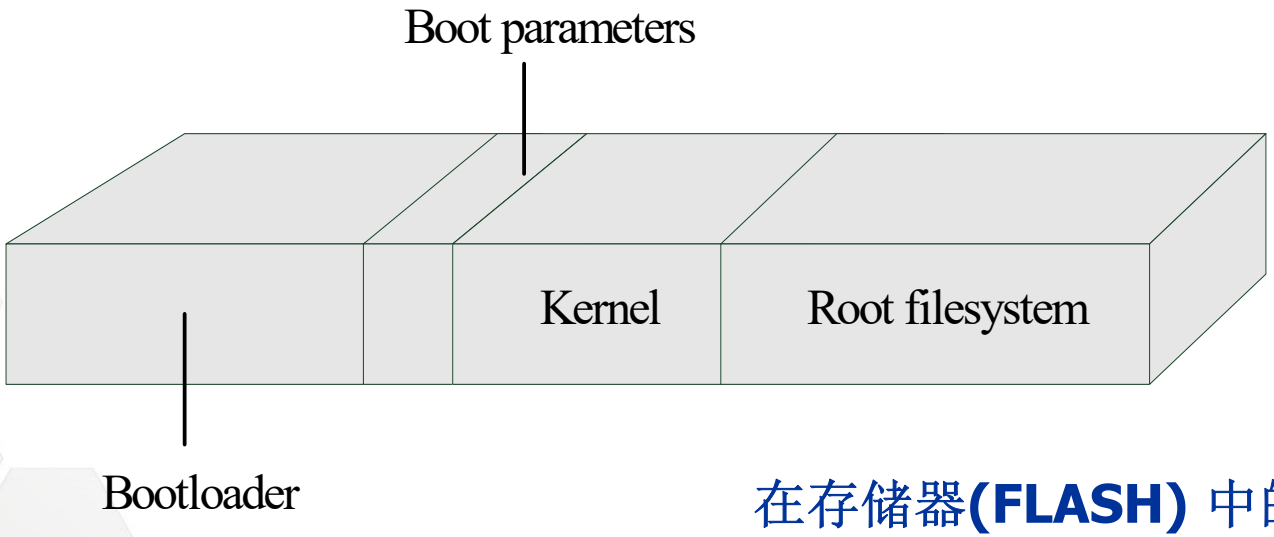
- 内核上面是应用程序和工具程序。链接库通常与应用程序动态链接在一起

8.1 嵌入式linux的组成

❖3、嵌入式Linux系统映像文件结构

- Bootloader
- Kernel(内核)
- Filesystem(文件系统)

（1）uClinux的映像组成



在存储器(FLASH) 中的布局

8.1 嵌入式linux的组成

❖ Bootloader

- 在操作系统内核运行之前运行的一段程序，它类似于PC机中的BIOS程序。通过这段程序，可以完成硬件设备的初始化，并建立内存空间的映射关系，从而将系统的软硬件环境带到一个合适的状态，为最终加载系统内核做好准备
- Bootloader比较依赖于硬件平台，特别是在嵌入式系统中，更为如此。因此，在嵌入式世界里建立一个通用的Bootloader是一件比较困难的事情。

8.1 嵌入式linux的组成

（2）uClinux（Linux）的启动流程

- **0x0地址的存储器读取Bootloader,并运行**
  - 初始化硬件，包括内存的初始化，串口的初始化等
  - 将Kernel从FLASH拷贝到SDRAM,配制的内核启动参数，跳到Kernel运行，Kernel获得CPU控制权
- **Kernel运行**
  - 对各个硬件单元进行初始化，必要的内核软件的初始化
  - 挂接根文件系统
  - 执行必要的应用程序的环境配制，调用开机自动运行的程序
- **应用程序开始执行**

第8章 嵌入式linux程序开发

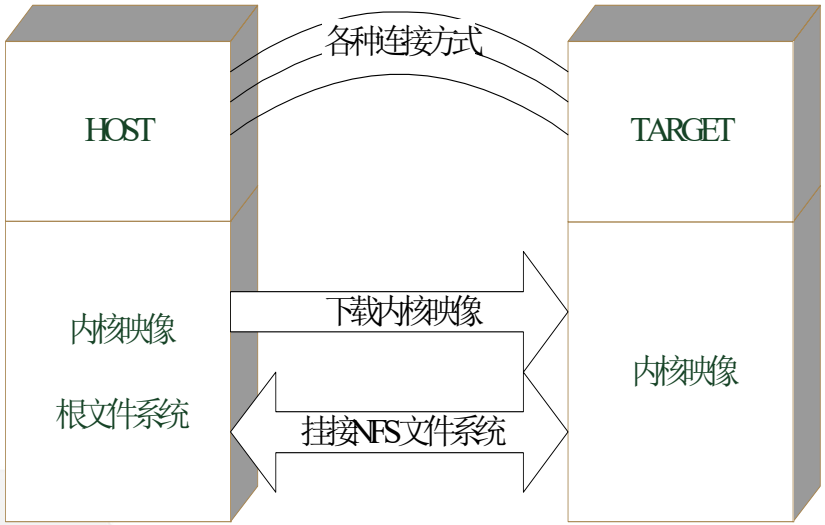
- 8.1 嵌入式linux的组成
- 8.2 嵌入式linux交叉开发环境
- 8.3 嵌入式linux驱动程序设计
- 3学时

8.2 嵌入式linux交叉开发环境

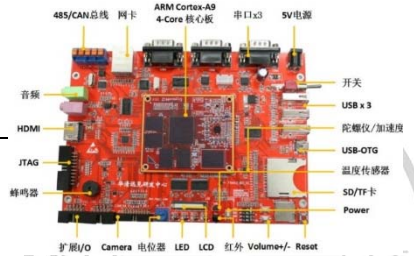
❖1、交叉开发概述

❖交叉开发（Cross Developing）

- 开发系统在PC机（一般称为宿主机或host）上，即完成软件的编辑、编译、链接等工作
- 软件的运行是在嵌入式设备（一般称为目标机或target）上



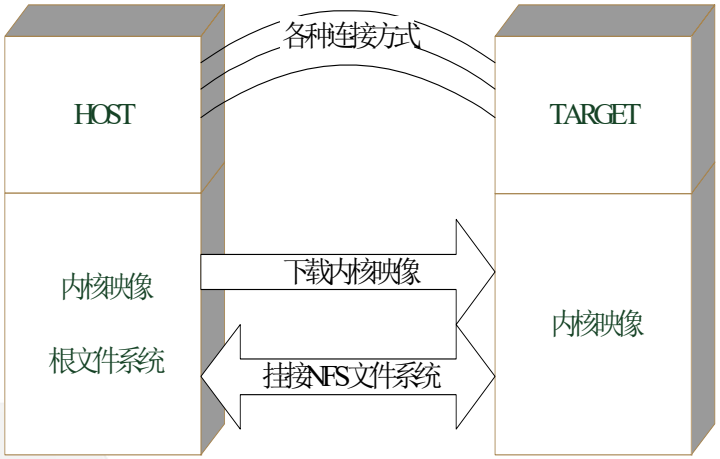
宿主机



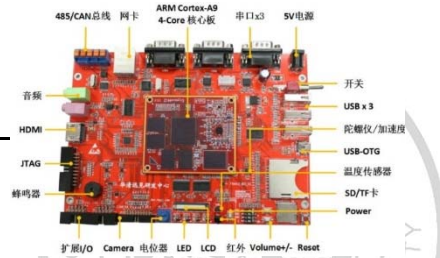
目标机

8.2 嵌入式linux交叉开发环境

- 交叉编译定义
  - 交叉编译是指：在宿主机上进行软件编辑、编译、链接等，并生成能够在目标机上运行的可执行程序的过程。
- 使用交叉编译的原因：
  - 由于嵌入式系统的硬件资源有限，不能安装编译所需要的资源。



宿主机



目标机

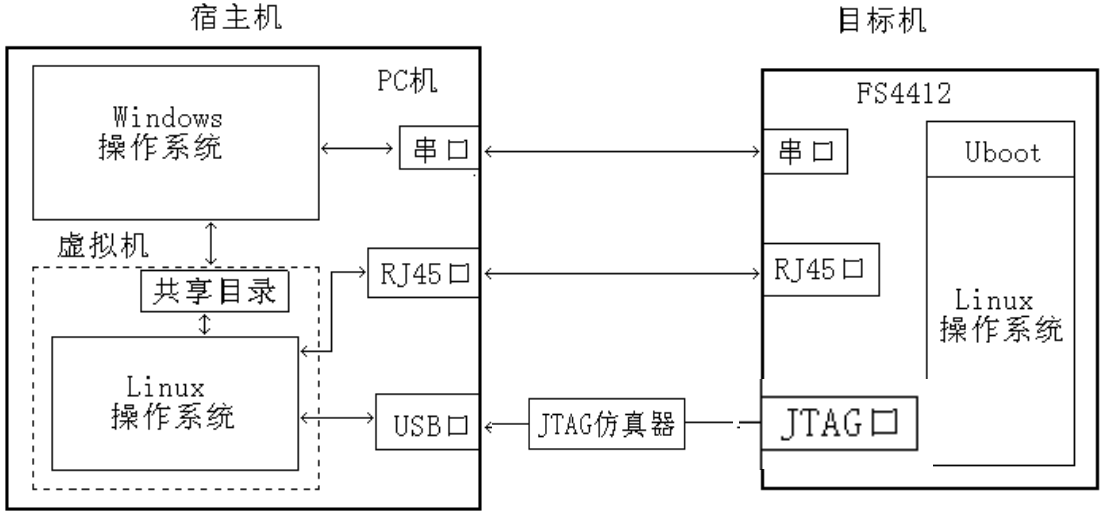
8.2 嵌入式linux交叉开发环境

- 嵌入式linux交叉开发流程
  - 1.安装交叉工具链
  - 2.编译生成Bootloader
  - 3.编译生成Kernel
  - 4.生成根文件系统
  - 5.运行调试

8.2 嵌入式linux交叉开发环境

❖2、交叉编译环境构建

- 宿主机（嵌入式交叉编译环境）PC+Linux+交叉编译工具链（ toolchain-4.4.6 ）
- 目标机（嵌入式软件运行环境）引导程序+嵌入式操作系统

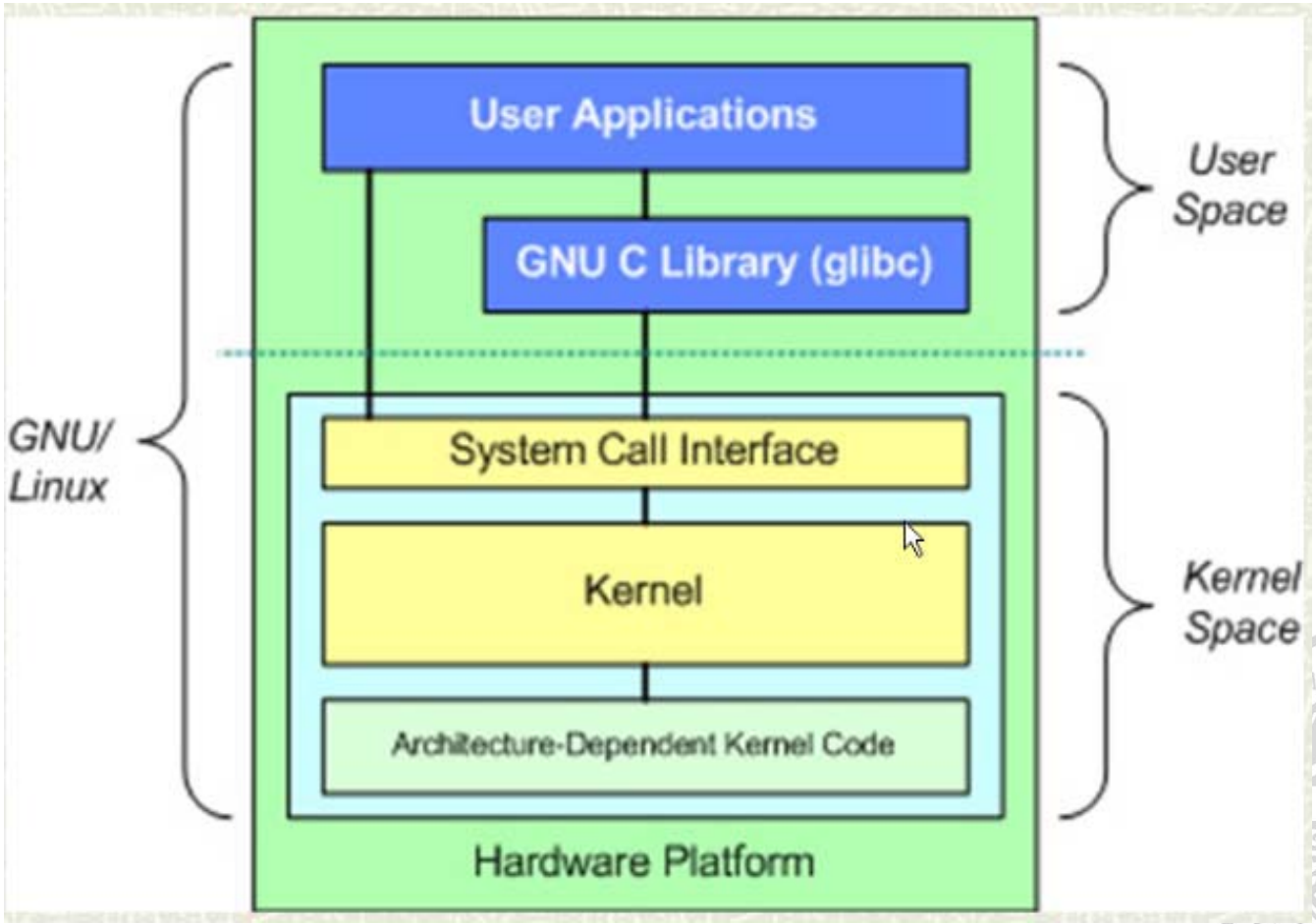


8.2 嵌入式linux交叉开发环境

- 交叉编译环境（交叉工具链）
  - 安装在/usr/local/toolchain/toolchain-4.6.4/bin/目录
  - 配置过程参考实验教学
- 目标机运行环境构建（软件部署）
  - Bootloader
  - Kernel(内核)
  - Filesystem(文件系统)
  - 烧写过程参考实验教学

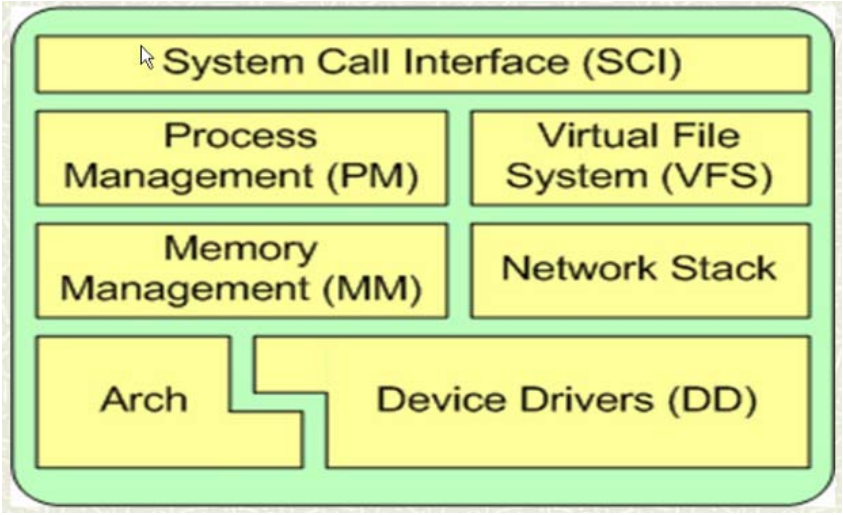
8.2 嵌入式linux交叉开发环境

❖3、Linux内核介绍



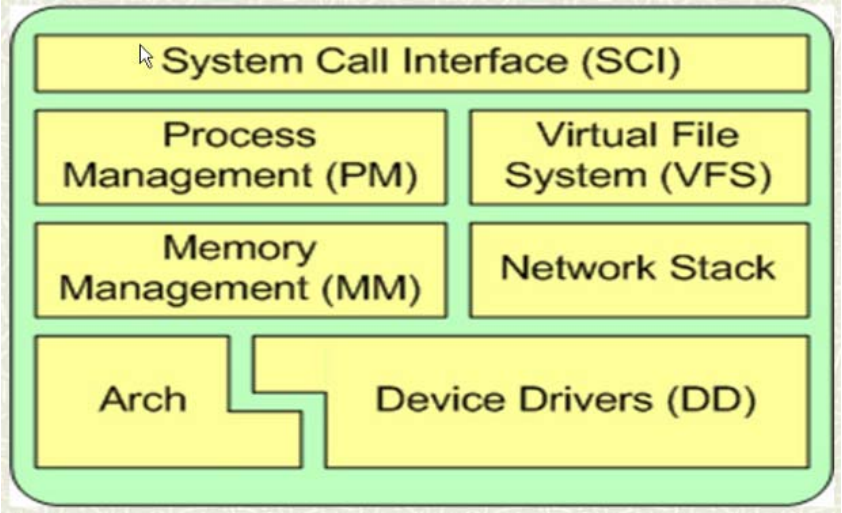
8.2 嵌入式linux交叉开发环境

- （1）系统调用接口（SCI）
  - 用户空间提供了一套标准的函数来访问Linux内核，搭起了用户空间到内核空间的桥梁
- （2）进程管理（PM）
  - 创建进程（fork、exec），停止进程（kill、exit），并控制它们之间的通信（signal或者POSIX机制），进程调度。



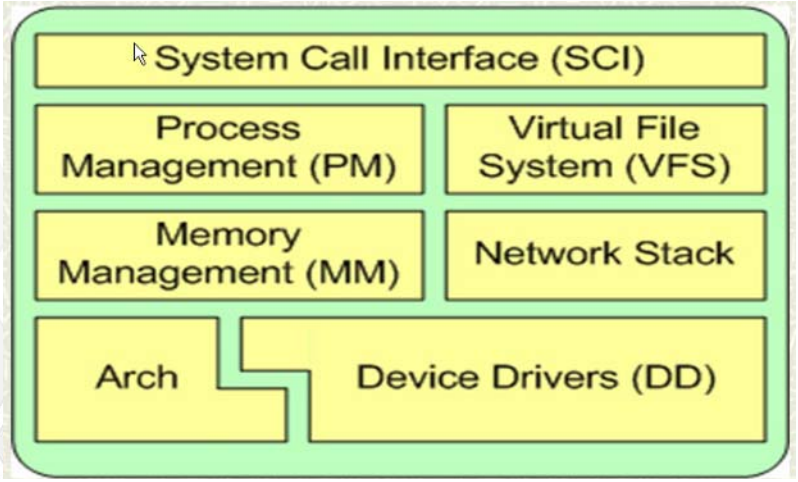
8.2 嵌入式linux交叉开发环境

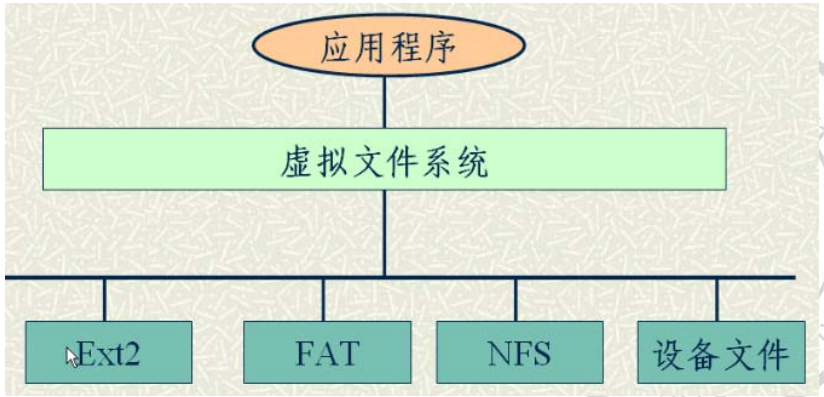
- （3）内存管理（MM）
  - 控制多个进程安全地共享内存区域。
- （4）网络管理（协议栈，NS）
  - 实现（Tcp/Ip、PPPoE）PPPoE全称Point to Point Protocol over Ethernet（基于以太网的点对点协议）。



8.2 嵌入式linux交叉开发环境

- （5）文件系统（虚拟文件系统VFS）
  - VFS隐藏了各种文件系统的具体细节，为所有类型的文件系统提供统一的操作接口
- （6）设备驱动（DD）
  - Linux内核中有大量代码都在设备驱动程序中，它们控制特定的硬件设备





8.2 嵌入式linux交叉开发环境

❖ Linux内核源代码结构

- Linux内核源代码通常位于/usr/src/目录下

目录	用途
arch	体系结构相关的代码，如 arch/i386、arch/arm等。
drivers	设备驱动程序，例如，drivers/char、drivers/block等。
fs	文件系统，例如，fs/ext3、fs/jffs2等。
include	头文件，include/asm 是体系结构相关的头文件，include/linux是Linux 内核头文件
init	Linux 初始化，如 main.c等。
ipc	进程间通信的代码。

8.2 嵌入式linux交叉开发环境

kernel	Linux内核核心代码。
lib	库文件，如 zlib、crc32等。
mm	内存管理代码。
net	网络支持代码，主要是网络协议。
sound	声音驱动的支持。
scripts	使用的脚本。
usr	用户代码。

8.2 嵌入式linux交叉开发环境

❖ 根文件系统的构建

- 数据保存到设备上所使用的一种组织结构或格式。
- 操作系统访问外部设备数据所约定的一种通用访问接口格式
- 根文件系统是Linux（或者说是UNIX类）操作系统运行时所需要的特有文件系统
- 包括操作系统运行时的配置数据文件（通常位于/etc目录下)和设备文件（位于/dev目录下）

8.2 嵌入式linux交叉开发环境

❖4、内核的移植

- 针对具体的硬件平台，修改Linux源代码（主要修改与体系结构相关的代码），然后重新编译，生成能够在目标台上运行的映像文件，
- Linux内核结构要非常熟悉，还要对目标平台的硬件结构非常熟悉。
- 这部分工作一般由目标平台提供商来完成

8.2 嵌入式linux交叉开发环境

目录	用途
/bin	系统的一些重要的执行文件。如：登录命令login；文件操作命令cp、mv、rm、ln等；文件编辑器ed、vi等；磁盘管理程序dd、df、mount等；系统实用程序uname、hostname等等
/boot	引导加载(bootstrap loader)使用的文件。
/dev	系统的设备文件。
/etc	系统的配置文件。如：/etc/inittab文件决定系统启动的方式。
/home	系统用户的工作目录。
/lib	系统上所需的函数共享库。

8.2 嵌入式linux交叉开发环境

/mnt	系统管理员临时mount的安装点。如：光驱、软盘都挂接在此目录的cdrom和floppy子目录中。
/proc	记载整个系统的运行信息。
/root	根用户的主目录。
/tmp	临时文件而保留的目录。
/usr	是Linux中内容最多、规模最大的一个目录。它包含所有命令、库、man页和其他一般操作中所需的不改变的文件。
/var	包括系统运行时要改变的数据。

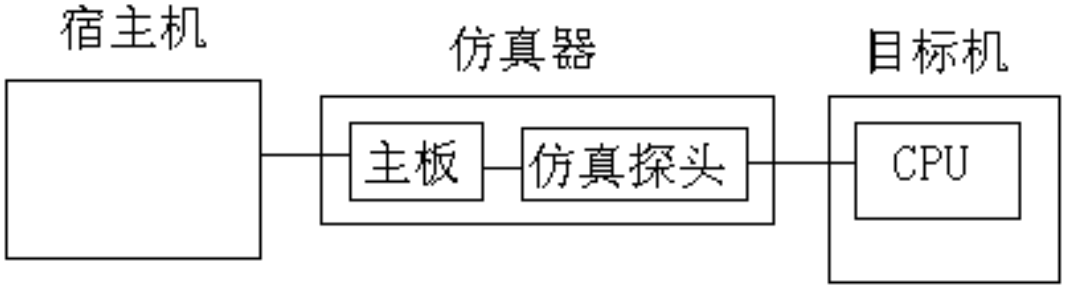
8.2 嵌入式linux交叉开发环境

❖5、嵌入式常用的调试方法

- （1）实时在线仿真
- （2）模拟调试
- （3）软件调试
- （4）OCD调试（片上调试器）

8.2 嵌入式linux交叉开发环境

- （1）实时在线仿真（ICE）方式
  - 一种用于替代目标上CPU的设备，可以执行目标机CPU指令，能够将内部的信号输出到被控的目标机，ICE上的内存也可以被映射到用户的程序空间。



  - 优点：功能非常强大，软硬件均可做到完全实时在线调试。
  - 缺点：价格昂贵。

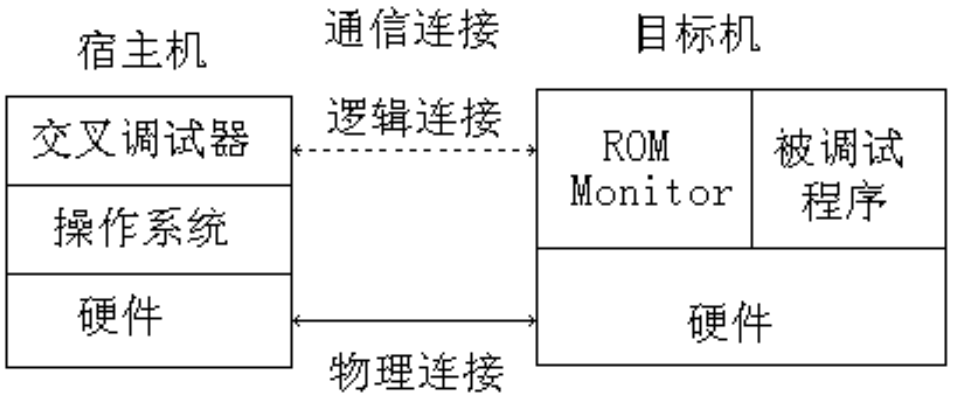
8.2 嵌入式linux交叉开发环境

- （2）模拟调试
  - 调试工具和待调试的嵌入式软件都在宿主机上运行，由宿主机提供一个模拟的目标运行环境，可以进行语法和逻辑上的调试。
  - 优点：简单方便，不需要目标机，成本低。
  - 缺点：功能非常有限，无法实时调试。

8.2 嵌入式linux交叉开发环境

（3）软件调试

- 宿主机和目标机通过某种接口（通常是串口）连接，宿主机上提供调试界面，待调试软件下载到目标机上运行。(目标机上需要固化监控程序）



- 优点：纯软件，价格较低，简单，软件调试能力较强
- 缺点：需要事先在目标机上烧写监控程序，目标机能正常工作，功能有限，特别是硬件调试能力较差。

8.2 嵌入式linux交叉开发环境

- （4）OCD方式（片上调试器）
  - 将ICE提供的实时跟踪和运行控制分开，使用很少的实时跟踪功能放弃，而大量使用的运行控制放到目标机的CPU核内，OCD可以提供ICE80%的功能，成本还不到ICE的20%。

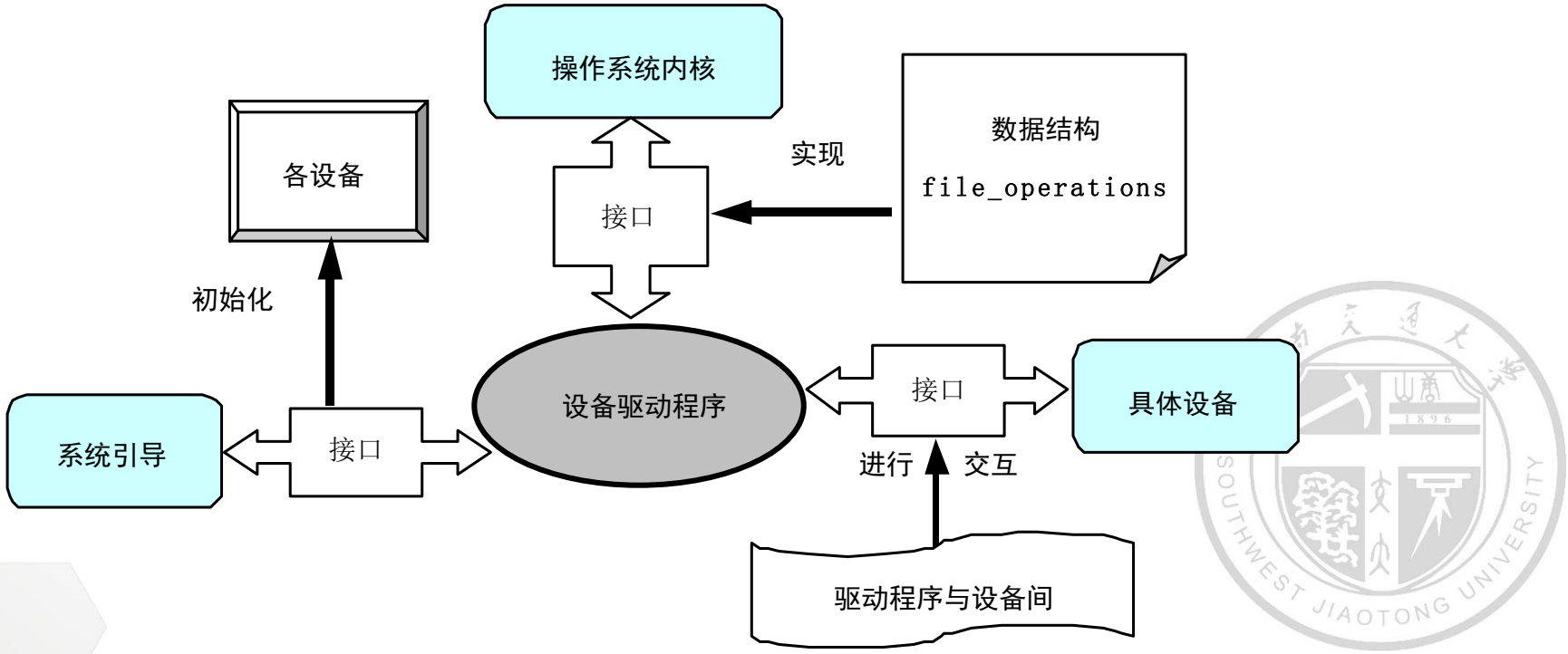
第8章 嵌入式linux程序开发

- 8.1 嵌入式linux的组成
- 8.2 嵌入式linux交叉开发环境
- 8.3 嵌入式linux驱动程序设计
- 3学时

8.3 嵌入式linux驱动程序设计

1、 嵌入式Linux驱动程序概述

- 驱动程序定义
  - 指挥硬件工作的软件。它是应用程序与硬件之间的一个中层软件层，为应用程序屏蔽硬件的细节。



8.3 嵌入式linux驱动程序设计

❖设备驱动程序的特点

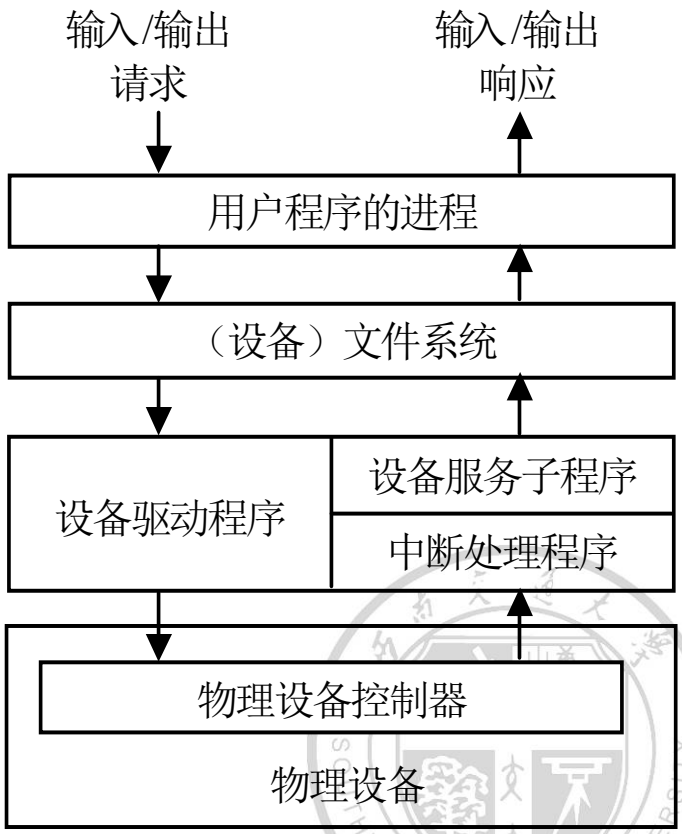
- （1）内核代码：设备驱动程序是内核的一部分，如果驱动程序出错，则可能导致系统崩溃。
- （2）内核接口：设备驱动程序必须为内核或者其子系统提供一个标准接口。
- （3）内核机制和服务：设备驱动程序使用一些标准的内核服务，如内存分配等。
- （4）可装载：大多数的Linux操作系统设备驱动程序都可以在需要时装载进内核，在不需要时从内核中卸载。
- （5）可设置：Linux可以根据需要把其中的某一部分集成到内核中。
- （6）动态性：在系统启动且各个设备驱动程序初始化后，驱动程序将维护其控制的设备。

8.3 嵌入式linux驱动程序设计

- (1)驱动程序是如何加载到Linux系统？
  - 系统中登记注册设备及其驱动程序，以使系统知道设备的存在以及状态
  - 从进程的角度看，外部设备的驱动程序就是一组包括中断服务程序的操作函数集合。重点要关心如何向应用进程提供那些供进程调用的操作函数及管理方法。

8.3 嵌入式linux驱动程序设计

- (2)驱动程序内部有几类模块（向下：硬件、向上：应用程序）
- (3)应用程序如何调用（**字符设备**）驱动程序
  - 设备节点（设备文件）
- (4)驱动层与应用层的分工问题
  - 驱动层：为应用层提供实现灯的亮和灭的功能函数。
  - 应用层：实现什么时间亮（或灭）。亮（或灭）多长时间等



8.3 嵌入式Linux驱动程序设计

- Linux驱动程序的知识结构：
  - 1、Linux驱动设计规范（50%）
  - 2、内核相关知识（25%）
  - 3、硬件相关知识（25%）

8.3 嵌入式Linux驱动程序设计

❖2、设备驱动原理

❖驱动程序的功能

- ➢对设备初始化和释放。
- ➢检测和处理设备出现的错误。
- ➢为应用程序提供统一的接口，用于数据传送。

❖驱动程序的组成

- ➢自动配置和初始化子程序。
- ➢服务于I/O请求的子程序，又称驱动程序的上半部分。
- ➢中断服务子程序，又称驱动程序的下半部分。

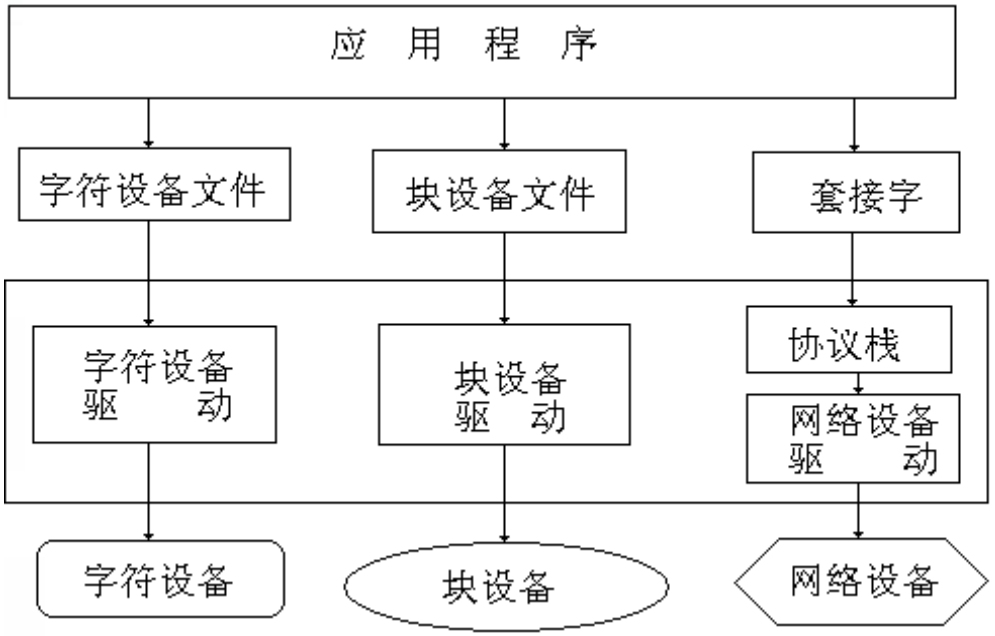
8.3 嵌入式linux驱动程序设计

❖驱动程序与应用程序的区别

- 应用程序有一个main函数，总是从该函数开始主动执行一个任务，而驱动程序安装之后，便停止工作，并等待被应用程序调用。
- 使用的库函数不同
- 程序运行的区域不同。驱动程序工作在内核态；应用程序工作在用户态。

8.3 嵌入式linux驱动程序设计

- Linux将设备分为3类
  - 字符设备（character devices）（重点）
  - 块 设 备（block devices）
  - 网络设备（network devices）



8.3 嵌入式linux驱动程序设计

- 字符设备（用C表示）
  - 指数据处理以字节为单位，并按顺序进行访问的设备。
  - 如：简单按键、触摸屏、AD转换。
- 块设备（设备用B表示）
  - 指在输入/输出时数据处理以块为单位的设备，它一般都采用缓存技术，支持数据的随机读写。
  - 典型设备：硬盘、U盘、内存、Flash、CD-ROM等
- 字符设备和块设备的区别
  - 最小访问单元不同。
  - 访问方式不同（顺序、随机）

8.3 嵌入式linux驱动程序设计

❖ 网络接口设备

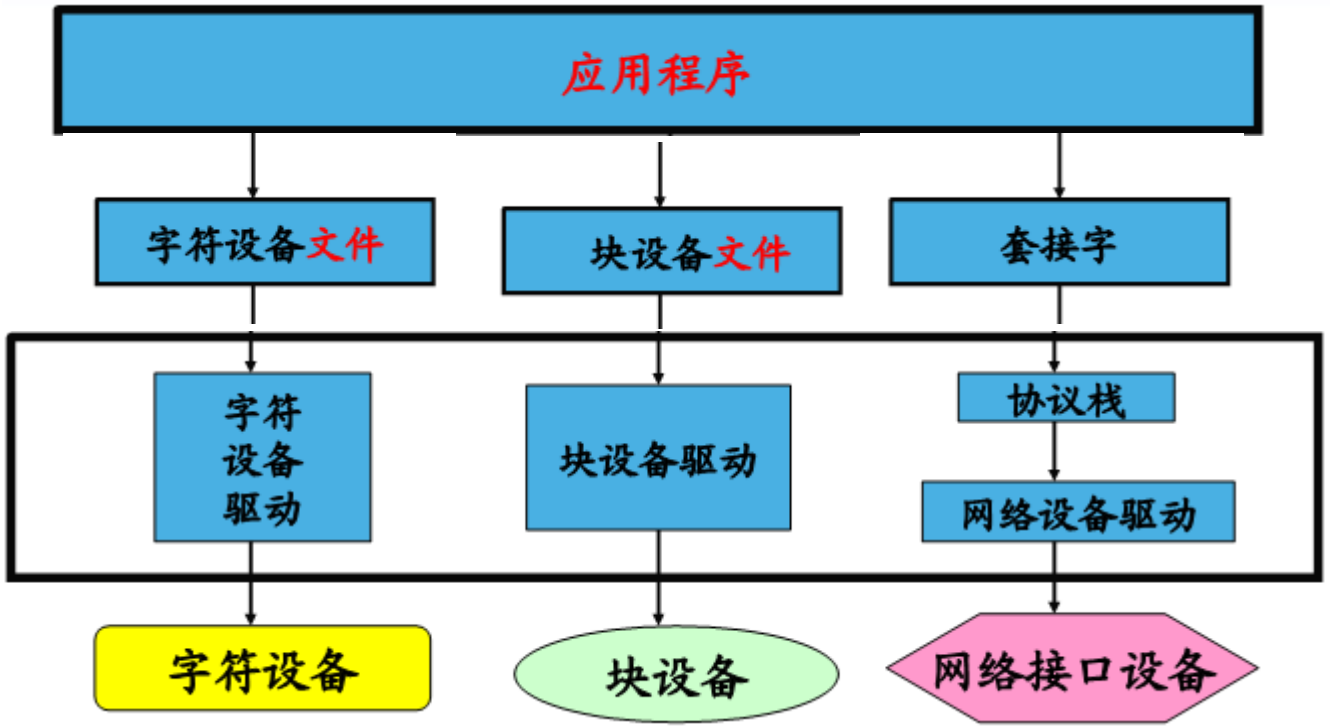
- 网络接口（network interfaces），用于网络通信。既可以是硬件设备，也可以是一个纯软件设备（如回环接口loopback）。
- 网络接口由内核中网络子系统驱动，负责发送和接收数据包，而且它并不需要了解每一项事务是如何映射到实际传送的数据包。

8.3 嵌入式linux驱动程序设计

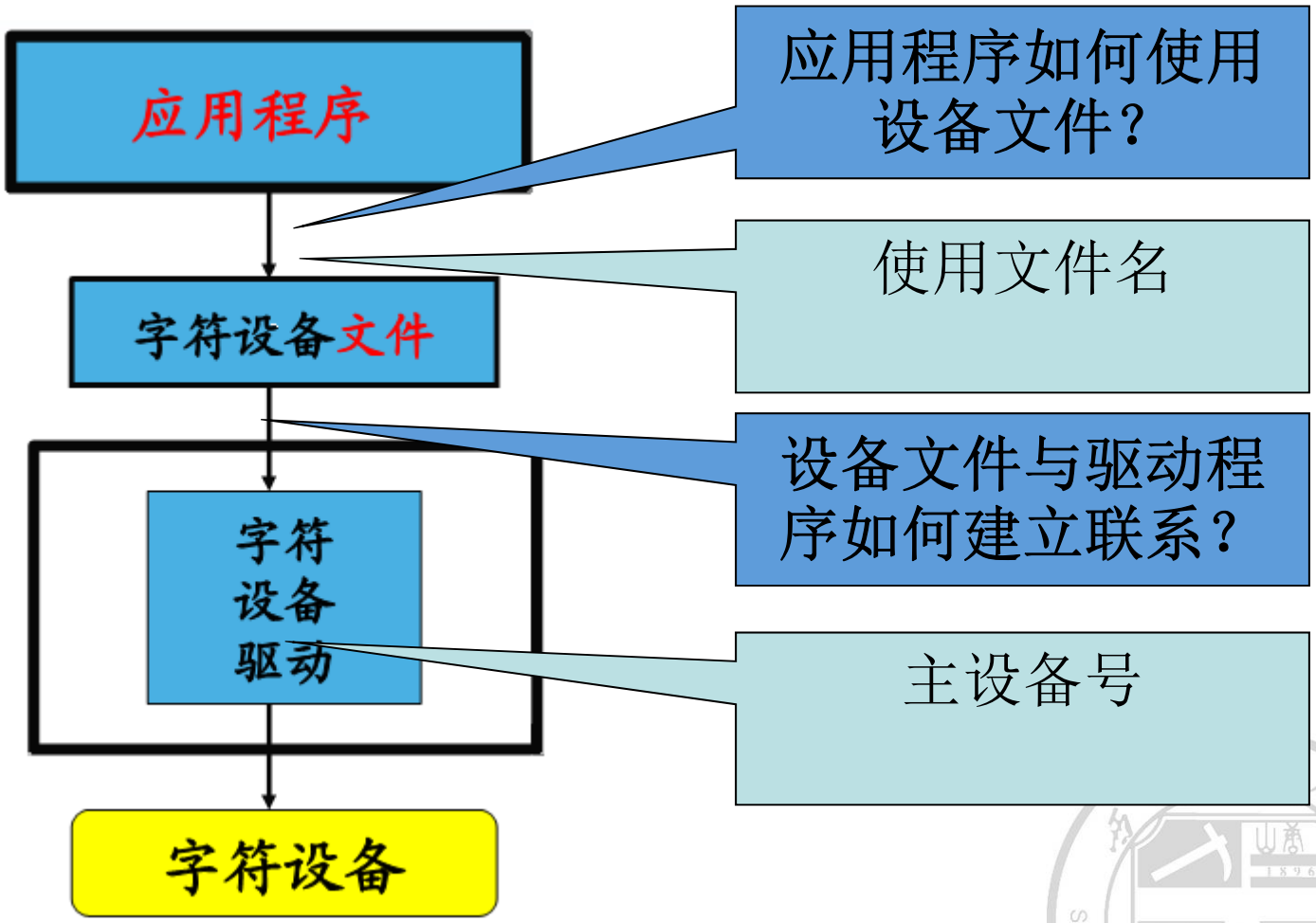
- 设备号
  - 设备号用于区分具体的设备。设备号包括主设备号（major number）和次设备号（minor number）
- 标识设备要作用3个参数
  - 设备类型：
  - 主设备号：用于标识设备对应的驱动程序（区分设备的类型）
  - 次设备号：用来区分具体设备的实例（区分同类型的设备）

8.3 嵌入式linux驱动程序设计

设备文件、主设备号和驱动程序之间的关系



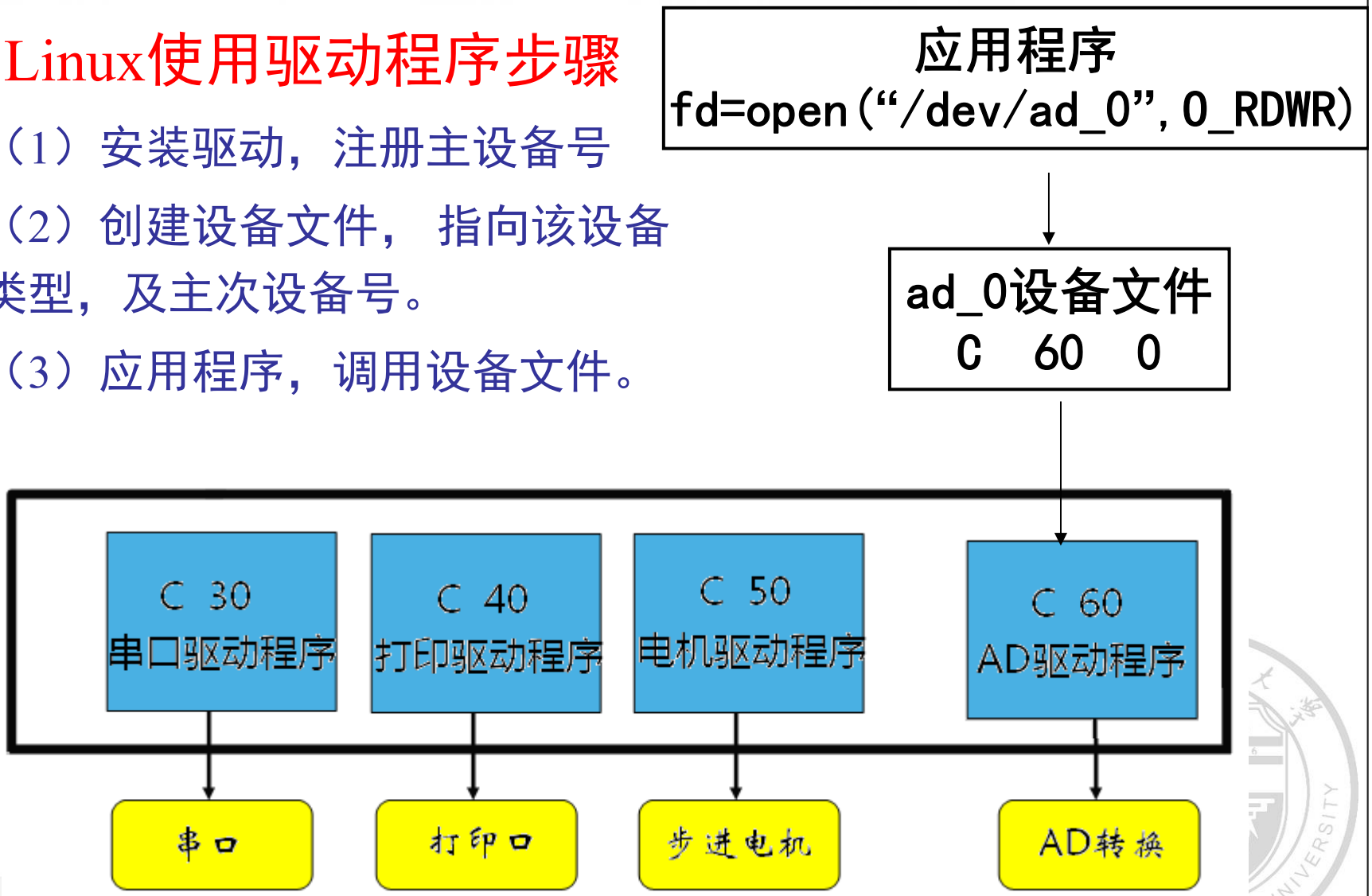
8.3 嵌入式linux驱动程序设计



8.3 嵌入式linux驱动程序设计

❖3、Linux使用驱动程序步骤

- （1）安装驱动，注册主设备号
- （2）创建设备文件， 指向该设备类型，及主次设备号。
- （3）应用程序，调用设备文件。



8.3 嵌入式linux驱动程序设计

- Linux内核分配设备主设备号方法
  - 静态申请
  - 动态申请
- 静态申请（由人工分配）
  - （1）查找没有使用的主设备号（cat /proc/devices）
  - （2）在驱动程序中，使用register_chrdev函数向内核注册主设备号。
  - 优点：简单
  - 缺点：如果驱动程序被广泛使用，有可能导致设备号冲突。

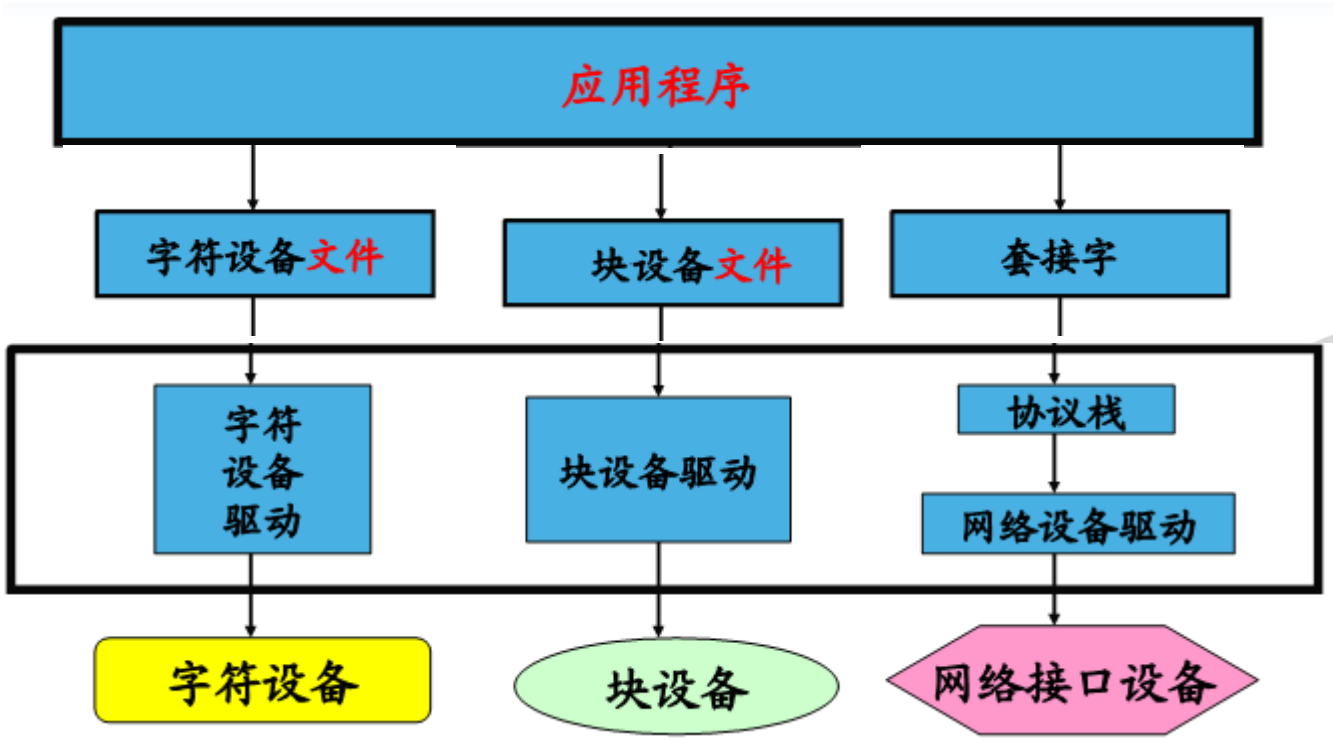
8.3 嵌入式linux驱动程序设计

- 动态申请分配主设备号（由内核分配）
  - （1）在驱动程序中，使用alloc_chrdev_region函数向内核申请动态分配，并注册主设备号。
  - 优点： 方便驱动程序推广。
  - 缺点：无法在安装驱动程序之前创建设备文件（因为安装之前还没有分配主设备号）。
  - 解决办法：安装后，使用命令cat /proc/devices查看分配到的设备号

8.3 嵌入式linux驱动程序设计

❖设备文件

- Linux应用程序是通过设备文件（又名：设备节点）来使用驱动程序操作字符设备和块设备。
- /dev目录是用于存放设备文件的



8.3 嵌入式linux驱动程序设计

- 设备文件的创建方法
  - 自动创建：在安装驱动程序时，自动创建设设备文件
  - 手动创建
- （1）手动创建命令——mknod
  - mknod name type major minor
- 例子：
  - mknod /dev/demo C 100 0
  - 创建了一个字符设备文件，主设备号是100，次设备号是0，设备文件名是/dev/demo

8.3 嵌入式linux驱动程序设计

❖4、驱动程序的加载方法

- 直接编译进内核，又称静态连接。
- 模块方式，又称动态连接。

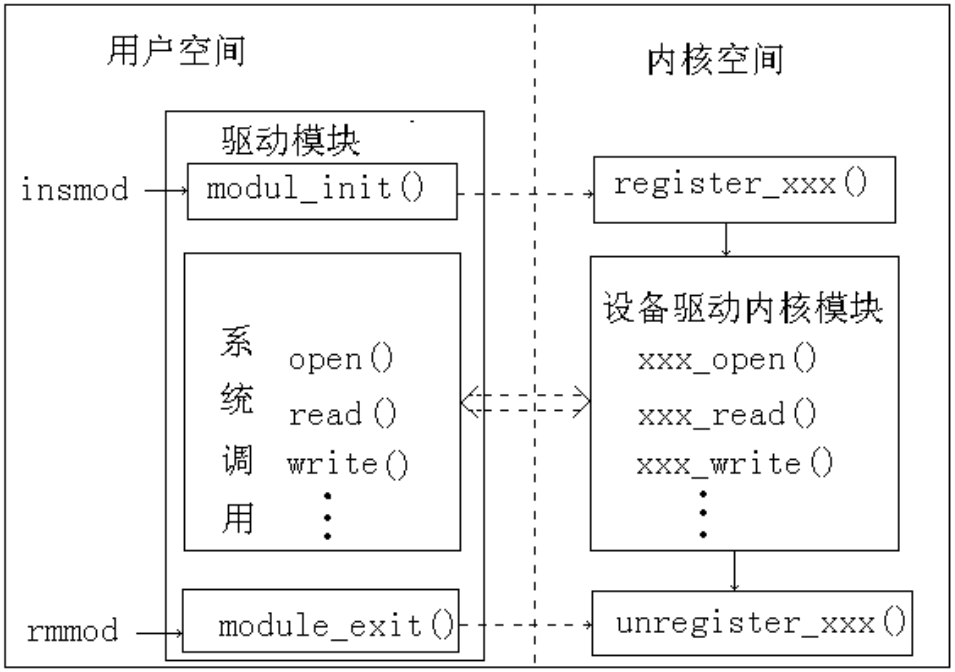
设备驱动模块化编程一般分为加载、系统调用和卸载等3个过程

❖insmod命令

- 将驱动模块加载到操作系统内核。

❖rmmod命令

- 将驱动模块从内核中删除



8.3 嵌入式linux驱动程序设计

- Linux驱动程序的学习方法

理论→实验→碰到问题→理论→实验

作业