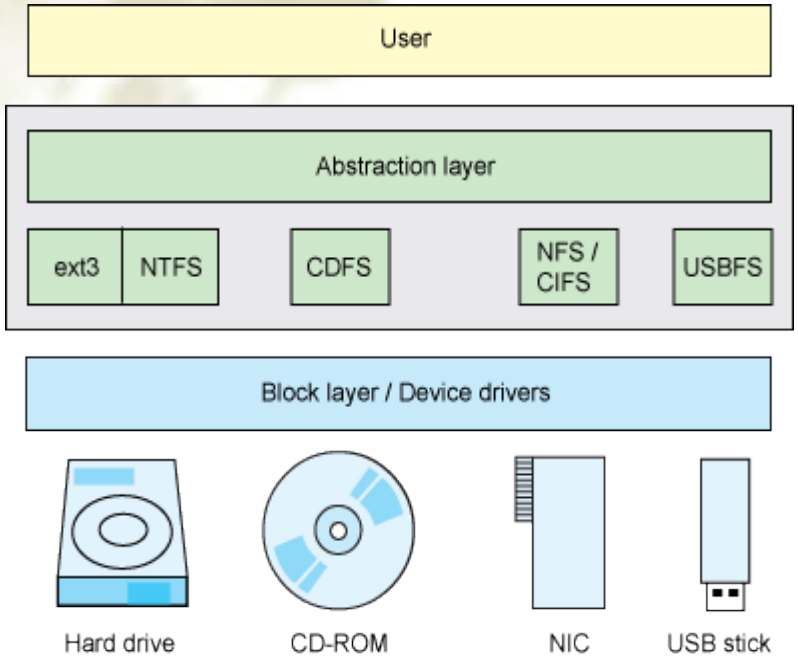
# 9.1 设备驱动概述


西安邮电大学

# 为什么引入设备驱动程序

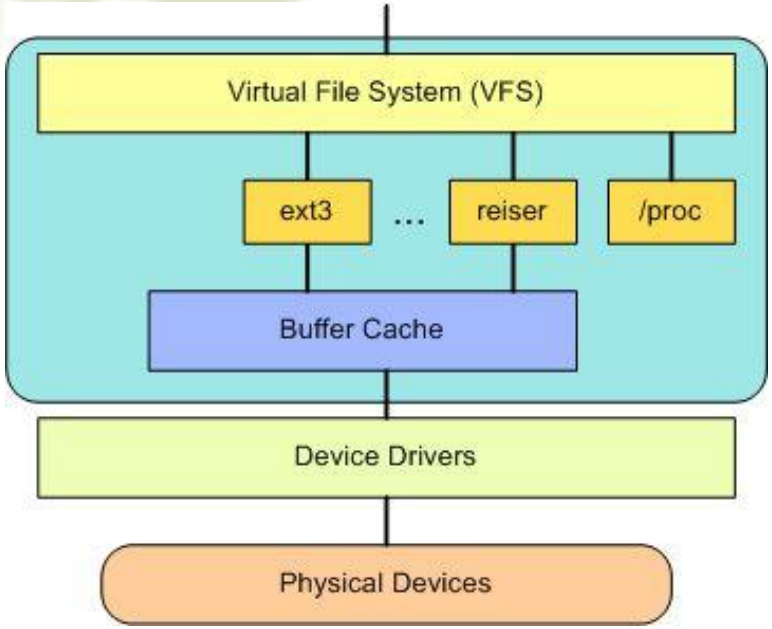



计算机中三个最基本的物质基础是CPU、内存和输入输出（I/O）设备。

与I/O设备相比，文件系统是一种逻辑意义上的存在，它只不过使对设备的操作更为方便、有效、更有组织、更接近人类的思维方式。可以说，文件操作是对设备操作的组织和抽象，而设备操作则是对文件操作的最终实现。

那么，如何才能使没有感觉的硬件，变得有“灵性”，从而控制设备像操作普通文件一样方便有效，这就是本章要讨论的设备驱动问题。

# 设备驱动程序-隐藏设备的细节

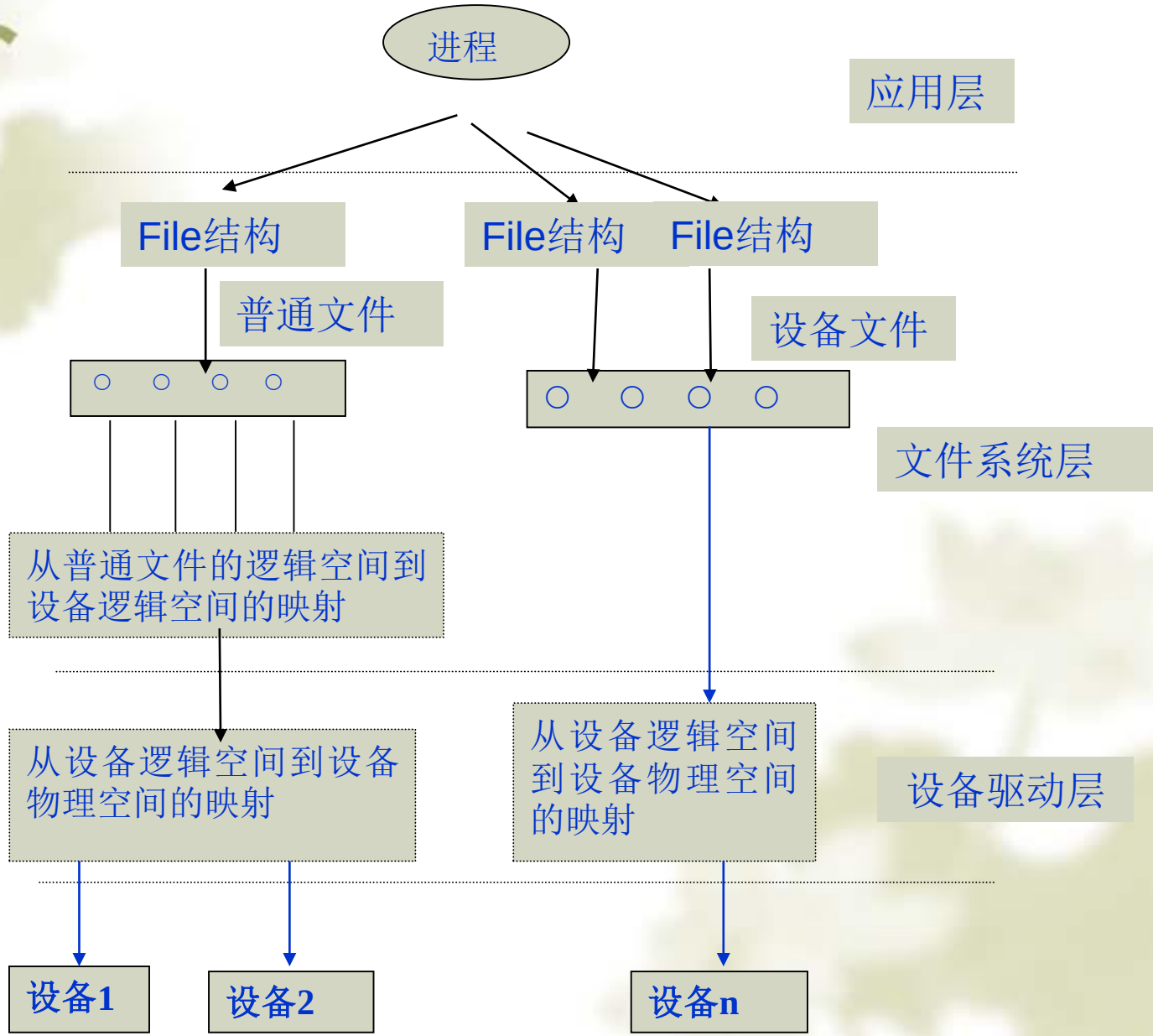



设备驱动程序在Linux内核中扮演着特殊的的角色，它们是一个个独立的“黑盒子”，使某个硬件响应一个定义良好的内部编程接口，这些接口完全隐藏了设备的工作细节。用户的操作通过一组标准化的调用执行，而这些调用独立于特定的驱动程序的任务。将这些调用映射到作用于实际硬件的设备特有操作上，则是驱动程序的任务。

# 设备纳入文件管理体系下

Linux操作系统把设备纳入文件系统的范畴来管理

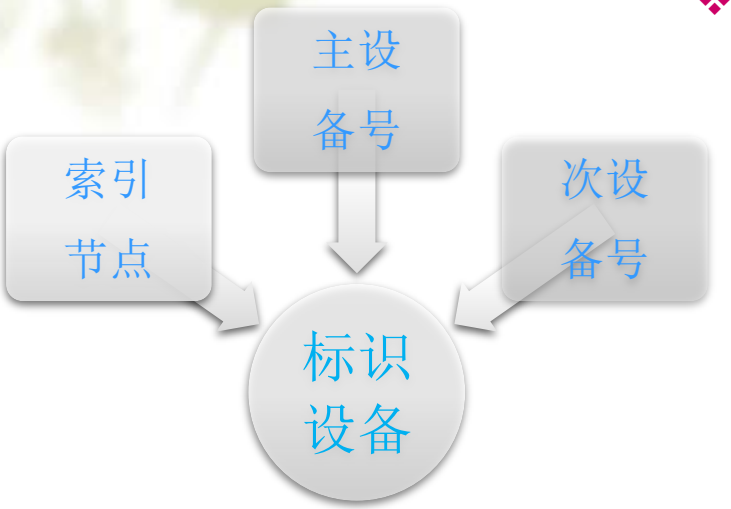
- 每个设备都对应一个文件名，在内核中也就对应一个索引节点
- 对文件操作的系统调用大都适用于设备文件
- 从应用程序的角度看，设备文件逻辑上的空间是一个线性空间（起始地址为0，每读取一个字节加1）。从这个逻辑空间到具体设备物理空间（如磁盘的磁道、扇区）的映射则是由内核提供，并被划分为文件操作和设备驱动两个层次


进程
应用层
File结构
File结构
File结构
普通文件
设备文件
文件系统层
从普通文件的逻辑空间到设备逻辑空间的映射
从设备逻辑空间到设备物理空间的映射
从设备逻辑空间到设备物理空间的映射
设备驱动层
设备1
设备2
设备n

# 设备驱动的层次观

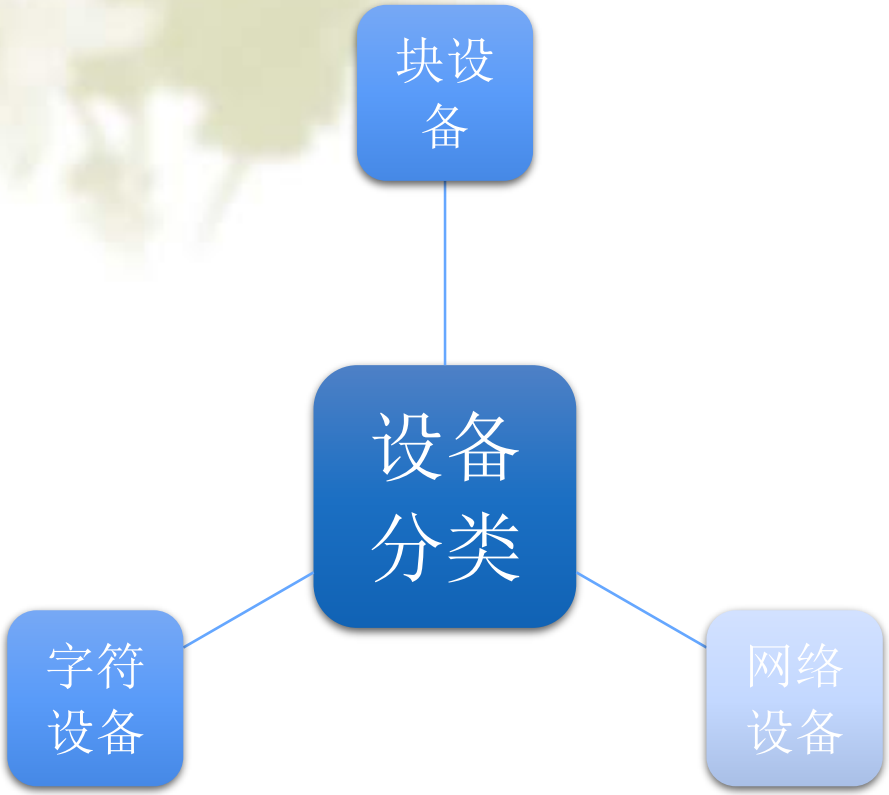
对于一个具体的设备而言，文件操作和设备驱动是一个事物的不同层次。从这种观点出发，从概念上可以把一个系统划分为应用、文件系统和设备驱动三个层次 。如图所示对于不同的设备，其文件系统层的“厚度”有所不同。对于像磁盘这样结构性很强，并且内容需要进一步组织和抽象的设备来说，其文件系统就很“厚重”，这是由磁盘设备的复杂性决定的。一方面是对磁盘物理空间的立体描述，如柱面、磁道、扇区；另一方面是从物理空间到逻辑空间的抽象，如第一层抽象，即线性地址空间中的“块”，以及在块之上又一层组织和抽象，即“磁盘文件”。这样，在物理介质上的第一层抽象使操作者不必关心读／写的物理位置究竟在哪一个磁道，哪一个扇区；而第二层抽象则使操作者不必关心读／写的内容在哪一个逻辑“块”中。于是，我们把第一层抽象归为设备驱动，而把第二层抽象归为文件系统。另一方面，还有一些像字符终端这样的字符设备，其文件系统就比较“薄”，其设备驱动层也比较简单。

# 如何标识设备




- 与文件用唯一的索引结点标识相似，一个物理设备也用唯一的索引节点标识，索引节点中记载着与特定设备建立连接所需的信息。这种信息由三部分组成括设备的类型、主设备号和次设备号。其中设备类型和主设备号结合在一起唯一地确定了设备的驱动程序及其接口，而次设备号则说明目标设备是同类设备中的第几个。
- 如图所示，通过索引节点，主设备号+次设备号来标识一个设备。

# 设备的分类




Linux将设备分成三大类：

1. 一类是像磁盘那样以块或扇区为单位，成块进行输入/输出的设备，称为块设备；

2. 另一类像键盘那样以字符为单位，逐个字符进行输入/输出的设备，称为字符设备。

3. 还有一类是网络设备。

# 不同设备类型的差异性

- Linux网络设备驱动与字符设备和块设备有很大的不同。
- 字符设备和块设备对应/dev下的一个设备文件。而网络设备不存在这样的设备文件。网络设备使用套接字socket访问，虽然也使用read,write系统调用，但这些调用只作用于软件对象。
- 块设备只响应来自内核的请求，而网络驱动程序异步接收来自外部世界的数据包，并发送到内核。
- 文件系统通常都建立在块设备上，也有很多文件系统放在内存，比如/proc，不需要驱动程序。

# 如何建立设备文件

系统调用

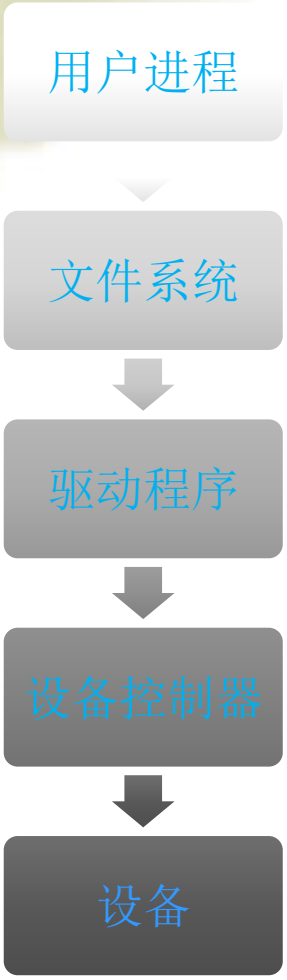
- mknod（）

命令

- 通过主设备号就可以把设备文件与驱动程序关联起来
- sudo mknod /dev/mycdev c 231

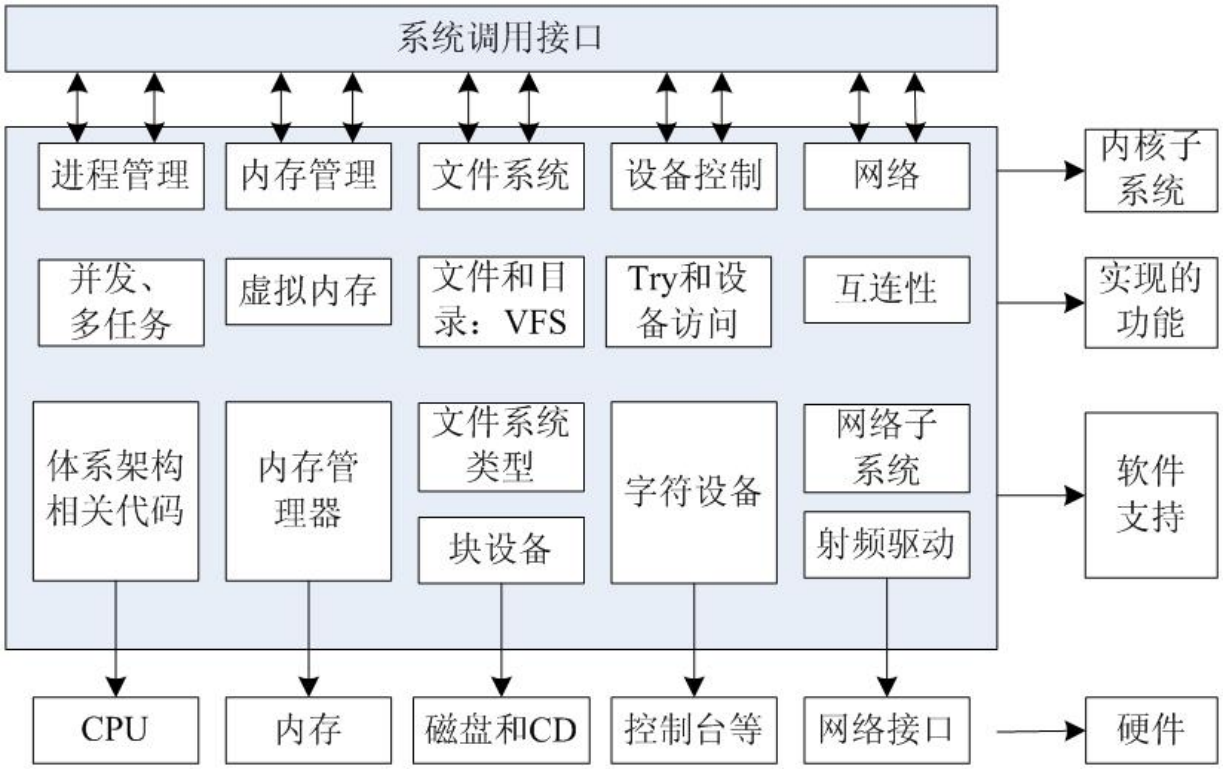
建立设备文件有两种方式，一是通过系统调用mknode(),一是通过命令mknod，如图所示。

# 设备驱动程序的位置




系统调用是内核和应用程序之间的接口，而驱动程序是内核和硬件之间的接口，也就是内核和硬件之间的桥梁。它为应用程序屏蔽了硬件的细节，这样在应用程序看来，硬件设备只是一个设备文件，应用程序可以像操作普通文件一样对硬件设备进行操作。

# 设备驱动程序的在子系统中的位置

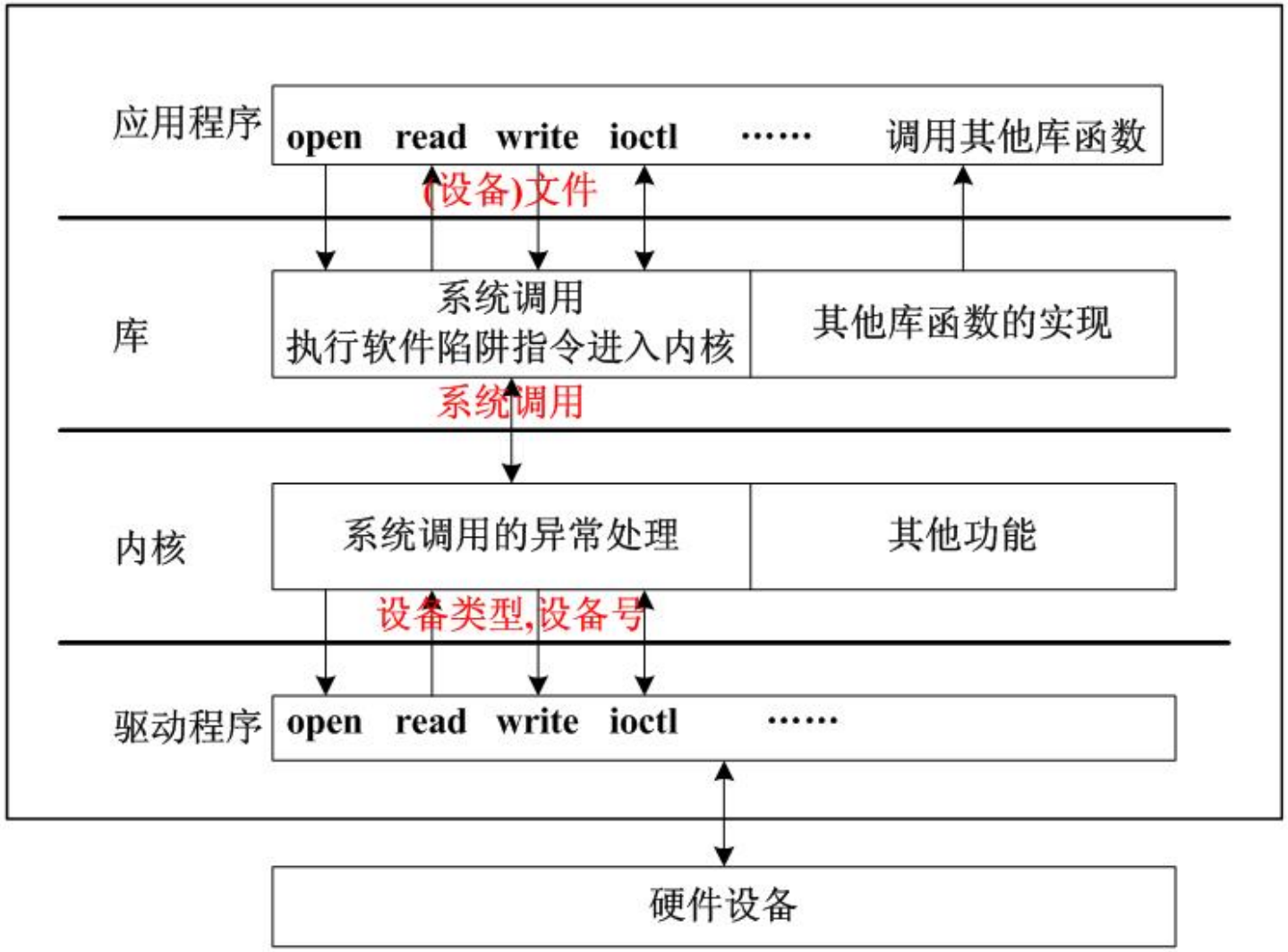



Linux内核分为5大部分：进程管理、内存管理、文件系统管理、设备管理、网络管理；

每一部分都有承上下的作用，对上提供API接口，提供给应用开发工程师使用；

对下通过驱动程序屏蔽不同的硬件构成，完成硬件的具体操作，如图所示。

# 设备驱动程序在应用开发中的位置



# 设备驱动程序在应用开发中的位置

1) 应用程序调用一系列函数库，通过对文件的操作完成一系列功能：

应用程序以文件形式访问各种硬件设备函数库；

部分函数无需内核的支持，由库函数内部通过代码实现，直接完成功能；

部分函数涉及到硬件操作或内核的支持，由内核完成对应功能，我们称其为系统调用

2) 内核处理系统调用，根据设备文件类型、主设备号、次设备号调用设备驱动程序；

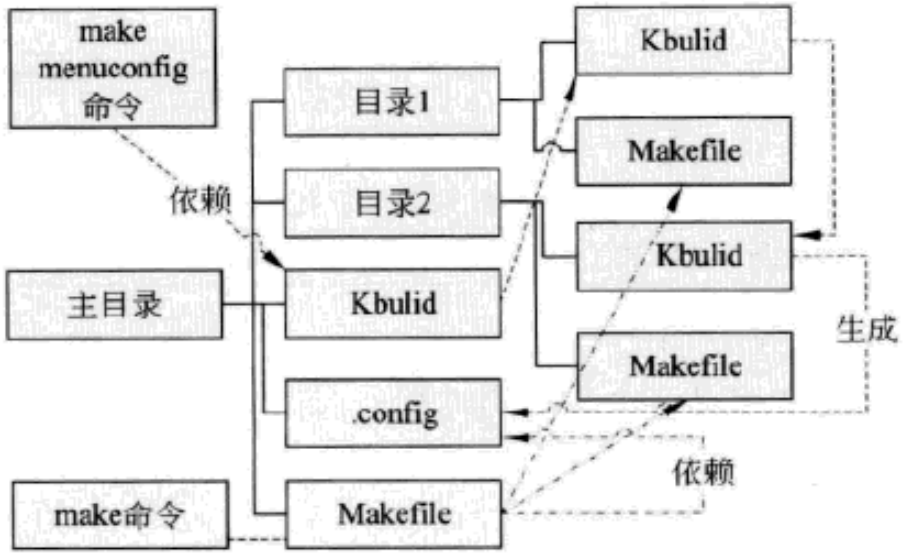
3) 设备驱动直接与硬件通信；

# 内核配置选项及驱动程序编译

我们要在内核中增加程序（比如驱动程序），并且使这个驱动程序能够编译进内核，基本分为两大部分。首先我们要告诉内核“请您下次编译的时候捎带上我”，也就是说需要我们进行内核的相关配置，这就需要对相关Makefie和Kconfig文件进行修改，以便让内核知道将要对这个新的驱动程序进行编译，即编译内核。为了简化内核的编译，有如下机制：

| 文件名 | 功能 |
| --- | --- |
| Makefile 文件 | 分布在 Linux 内核源代码根目录及各层目录中，定义 Linux 内核的编译规则 |
| Kconfig 文件 | 为用户提供一个层次化的配置选项集 |
| 配置文件（.config） | 当用户配置完后，将配置信息保存在.config 文件中。 |
| 配置工具 | 包括配置命令解释器和配置用户界面 |

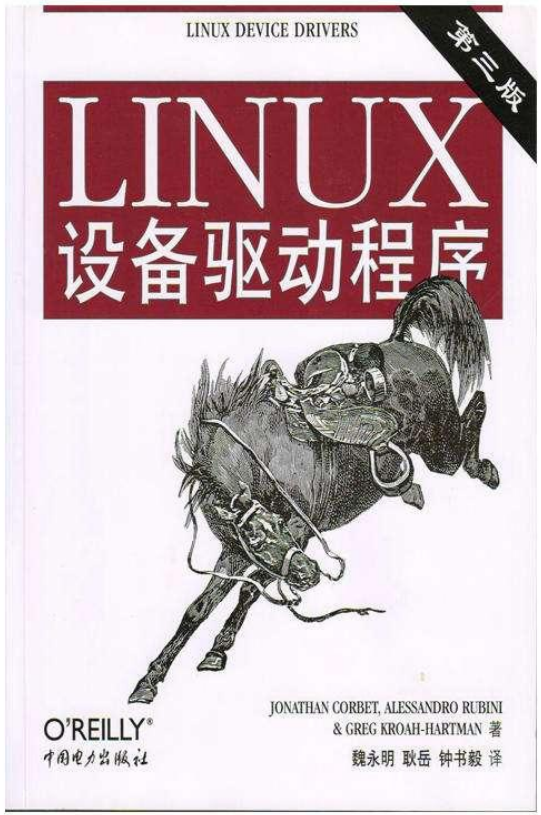
# 内核配置选项及驱动程序编译




当执行menuconfig 命令时，配置程序会依次从目录由浅入深查找每一个Kbuild文件，依照这个文件中的数据生成一个配置菜单。Kbuild像是一个分布在各个目录中的配置数据库，通过这个数据库可以生成配置菜单。在配置菜单中根据需要配置完成后会在主目录下生成一个.config文件，此文件保存了配置信息。

然后执行make命令，会依赖生成的.config文件，以确定哪些功能将编译入内核中，哪些功能不编译入内核中。然后递归地进入每一个目录，寻找Makefile文件，编译相应的代码。

# 参考文献




以上对驱动程序相关的基础知识进行了概要介绍，下面介绍一下参看文献

1.《Linux 驱动开发》（Linux Device Driver）是最经典的参考书

2.网上有大量详尽的驱动开发资料，读者可自行查阅，推荐一篇 :https://blog.csdn.net/Decisiveness/article/details/45060931

3.文中的很多图片，来自google搜索，版权归原作者所有

# 带着疑问上路

应用程序什么形式访问各种硬件设备？为什么采用这方式？

谢谢大家

# 谢谢大家！

THANK YOU