

<<嵌入式Linux应用开发精解>>

图书基本信息

书名：<<嵌入式Linux应用开发精解>>

13位ISBN编号：9787512410510

10位ISBN编号：7512410514

出版时间：2013-1

出版单位：北京航空航天大学出版社

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<嵌入式Linux应用开发精解>>

书籍目录

项目1构建嵌入式Linux开发环境 1.1知识背景 1.1.1嵌入式系统的组成 1.1.2嵌入式系统开发板 1.1.3交叉编译工具链 1.1.4交叉编译环境的组成 1.1.5 Linux服务 1.2项目需求 1.3项目设计 1.3.1开发板的选定 1.3.2开发方案的确定 1.4项目实施 任务一：组建开发平台 任务二：配置超级终端 任务三：安装与配置DNW 任务四：安装虚拟机 任务五：安装Linux操作系统 任务六：实现Windows共享 任务七：配置NFS服务 任务八：配置FTP服务 任务九：安装与配置交叉编译工具链 1.5项目小结 1.6工程实训 实训目的 实训环境 实训内容 实训步骤 1.7拓展提高 思考 操作 项目2开发简单应用程序 2.1知识背景 2.1.1嵌入式软件系统的组成 2.1.2 BootLoader的功能与使用 2.1.3内核移植与下载 2.1.4根文件系统的建立 2.1.5嵌入式应用程序开发 2.2项目需求 2.3项目设计 2.4项目实施 任务一：下载BootLoader 任务二：移植内核 任务三：移植根文件系统 任务四：编写“Hello World”应用程序 任务五：实现开机自启动“Hello World” 2.5项目小结 2.6项目实训 实训目的 实训环境 实训内容 实训步骤 2.7拓展提高 思考 操作 项目3开发设备驱动程序 3.1知识背景 3.1.1设备驱动程序的概念 3.1.2设备驱动程序的结构 3.1.3设备驱动开发的API函数 3.1.4设备驱动程序的调试 3.2项目需求 3.3项目设计 3.3.1理解驱动开发的本质 3.3.2驱动开发的一般流程 3.3.3内核态Hello World驱动程序设计 3.3.4 LED驱动程序设计 3.3.5按键驱动程序设计 3.4项目实施 任务一：实现内核态的驱动程序 任务二：实现LED驱动程序 任务三：实现键盘驱动程序 3.5项目小结 3.6工程实训 实训目的 实训环境 实训内容 实训步骤 3.7拓展提高 思考 操作 项目4实现图形用户界面应用程序 4.1知识背景 4.1.1 Qt简介 4.1.2 Qt开发环境 4.1.3 Qt编程机制 4.2项目需求 4.3项目设计 4.3.1构建Qt集成开发环境 4.3.2 Qt应用程序开发流程 4.3.3 Qt应用程序开发方法 4.3.4关闭系统的实现 4.3.5菜单命令的实现 4.3.6系统标准对话框的使用 4.3.7部署Qt应用程序 4.3.8实现Qt应用程序的开机自启动 4.4项目实施 任务一：建立Qt开发环境 任务二：建立Qt运行环境 任务三：编写Qt下的“Hello world”程序 任务四：实现开机自启动Hello_Qt4 4.5项目小结 4.6工程实训 实训目的 实训环境 实训内容 实训步骤 4.7拓展提高 思考 操作 项目5开发多线程程序 5.1知识背景 5.1.1进程的概念 5.1.2线程的概念 5.1.3 Qt中的线程类 5.1.4用户自定义事件在多线程编程中的应用 5.1.5利用定时器机制实现多线程编程 5.1.6利用QProcess实现多线程编程 5.2项目需求 5.3项目设计 5.3.1 LED控制原理 5.3.2 LED开发控制 5.3.3按钮控制的灵活性 5.3.4不同线程之间通信的实现 5.4项目实施 任务一：建立项目文件 任务二：设计程序界面 任务三：修改系统主界面类文件 任务四：添加LED控制子线程类定义文件 任务五：添加LED控制子线程类实现源文件 项目6 开发串口通信应用程序 项目7开发多媒体应用程序 项目8开发数据库应用程序 项目9开发网络应用程序 参考文献

<<嵌入式Linux应用开发精解>>

章节摘录

版权页：插图：（3）直接访问内存（DMA）方式利用中断，系统和设备之间可以通过设备驱动程序传送数据，但是当传送的数据量很大时，因为中断处理上的延迟，利用中断方式的效率会大大降低。

而直接内存访问（DMA）可以解决这一问题。

DMA可允许设备和系统内存间在没有处理器参与的情况下传输大量数据。

设备驱动程序在利用DMA之前，需要选择DMA通道并定义相关寄存器以及数据的传输方向，即读取或写入，然后将设备设定为利用该DMA通道传输数据。

设备完成设置之后，可以立即利用该DMA通道在设备和系统的内存之间传输数据，传输完毕后产生中断以便通知驱动程序进行后续处理。

在利用DMA进行数据传输的同时，处理器仍然可以继续执行其他指令。

5.内核空间 and 用户空间 在Linux系统中，软件的运行可在“内核空间”和“用户空间”中进行。

设备驱动程序可以编译到内核之中，也可以以模块形式存在。

模块在“内核空间”运行，而应用程序则是在“用户空间”运行。

它们分别引用不同的内存映射，也就是程序代码使用不同的“地址空间”。

Linux通过系统调用和硬件中断完成从用户空间到内核空间的控制转移。

执行系统调用的内核代码在进程的上下文中执行，它执行调用进程的操作而且可以访问进程地址空间的数据。

但处理中断与此不同，处理中断的代码相对进程而言是异步的，而且与任何一个进程都无关。

模块的作用就是扩展内核的功能，是运行在内核空间的模块化的代码。

模块的某些函数作为系统调用执行，而另一些函数则负责处理中断。

各个模块被分别编译并链接成一组目标文件，这些文件能被载入正在运行的内核，或从正在运行的内核中卸载。

必要时内核能请求内核守护进程Kerneld对模块进行加载或卸载。

根据需要动态载入模块可以保证内核达到最小，并且具有很大的灵活性。

内核模块一部分保存在Kernel中，另一部分在Modules包中。

在许多应用系统中，对设备安装、使用和改动都是通过编译进内核来实现的，对驱动程序稍微做点改动，就要重新烧写一遍内核，而且烧写内核经常容易出错，还占用资源，给具体应用带来不便。

模块采用的则是另一种途径，内核提供一个插槽，模块就像一个插件，在需要时插入内核中使用，不需要时从内核中拔出。

这一切都由一个称为Kerneld的守护进程自动处理。

内核模块的动态加载具有以下优点：将内核映像的空间保持在最小，并具有最大的灵活性。

这便于检验新的内核代码，而不需要重新编译内核并重新引导。

但是，内核模块的引入也对系统性能和内存的利用有负面影响。

装入的内核模块与其他内核部分一样，具有相同的访问权限，由此可见，差的内核模块会导致系统崩溃。

为了使内核模块能访问所有内核资源，内核必须维护符号表，并在加载和卸载模块时修改这些符号表。

。

<<嵌入式Linux应用开发精解>>

编辑推荐

《高职高专“十二五”规划教材:嵌入式Linux应用开发精解》既可作为高职院校计算机、物联网、电子工程和机电一体化等相关专业“嵌入式Linux应用开发”课程的教材，也可用作各类培训机构的培训教材，还可作为嵌入式Linux系统开发专业人员和业余爱好者的参考书和工具书。书中提供的项目源代码稍加移植、修改、扩充和组合，即可构建实用的嵌入式Linux系统。

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>