VxWorks 内存管理机制的研究

李加泗，宁洪

（国防科学技术大学，长沙 410073）

摘 要： 本文对嵌入式操作系统 VxWorks 的内存管理机制进行了研究。首先分析了系统启动阶段以及启动后系统的内存视图，然后分析了 VxWorks 中内存的分配与释放，最后对 VxWork 的内存保护机制进行了研究。本文对嵌入式系统的应用与开发有一定的参考价值。
关键字： VxWorks 内存视图 分配算法 内存保护

Research Of Memory Management Mechanism In VxWorks

Li Jiasi， Ning Hong

(National University Of Defense Technology, Changsha 410073)

Abstract This article makes a study of the memory management mechanism in VxWorks. First it gives the memory views of the system both during and after the booting of VxWorks. Then it talks about the dynamic allocating and freeing of memory in VxWorks. In the end it gives a discussion of the memory protection in VxWorks.
Keywords VxWorks; memory view; allocation algorithm; memory protection;

1. 序言

由于嵌入式系统对功能、可靠性、成本、体积以及功耗等的严格要求，系统内部的存储器容量往往比较小，如何利用有限的内存资源构建高可靠高实时性的嵌入式系统是嵌入式系统领域一个难点问题也是一个热点问题，因此搞清楚嵌入式操作系统的内存管理机制，有利于开发适合于各种实时应用的嵌入式系统。

VxWorks 是美国风河（WindRiver）公司于 1983 年推出的嵌入式实时操作系统（RTOS）。它以其良好的可靠性和卓越的实时性被广泛地应用在通信、军事、航空、航天等高精尖技术及实时性要求极高的领域中，如医疗器械、卫星通讯、军事演习、弹道制导、飞机导航等。本文主要研究 VxWorks 的内存管理机制。

2.VxWorks 内存的初始化

2.1 系统上电之后到系统初始化程序 usrInit()执行前的系统 RAM 视图。

由于系统镜像的不同，这一阶段的系统 RAM 视图有两种。与系统镜像类型相关的代码有：romInit()，romStart()，sysInit()。对于烧录到 ROM 中的镜像，系统启动时执行 romInit() 和 romStart()；而对于可加载的 VxWorks 镜像则只执行 sysInit()。之后统一跳转到 usrInit() 继续进行系统的后续初始化操作。这两种视图如下所示：

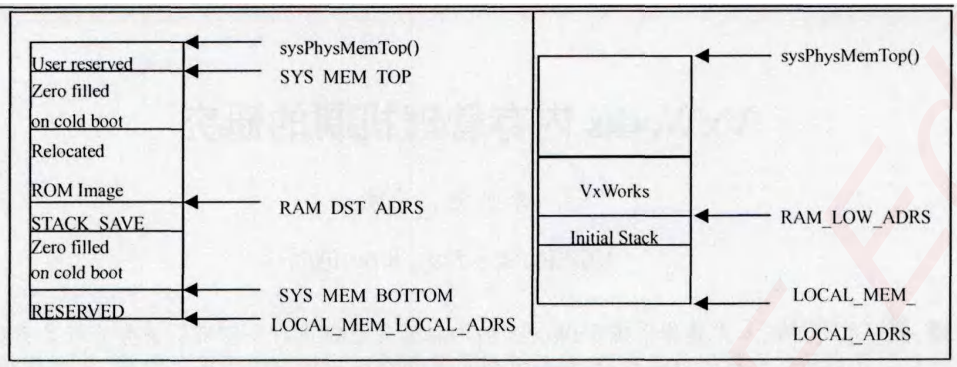

图 1 两种不同的 RAM 视图

这两种视图的主要区别在于程序 romInit()（执行完跳转到 romStart()）执行时初始化了内存系统，而 sysInit()则没有做这个工作。在 romStart()将 ROMA 镜像重定位到 RAM 中之后，除了保留内存以及 STACK_SAVE 外，它将除镜像代码段和数据段之外的内存全部清零。

2.2 系统初始化完毕后的内存视图

usrInit()是系统启动后执行的第一段 C 代码，它完成了剩下的系统初始化工作，包括内存的初始化。程序首先清零镜像的 BSS 段，之后调用 sysHwInit()进行部分硬件初始化工作，设置中断/异常向量表，最后调用两个内核初始化函数 usrKernelInit()和 KernelInit()。在 KernelInit()完成了系统内存分区的初始化工作。此时的内存视图如下所示：

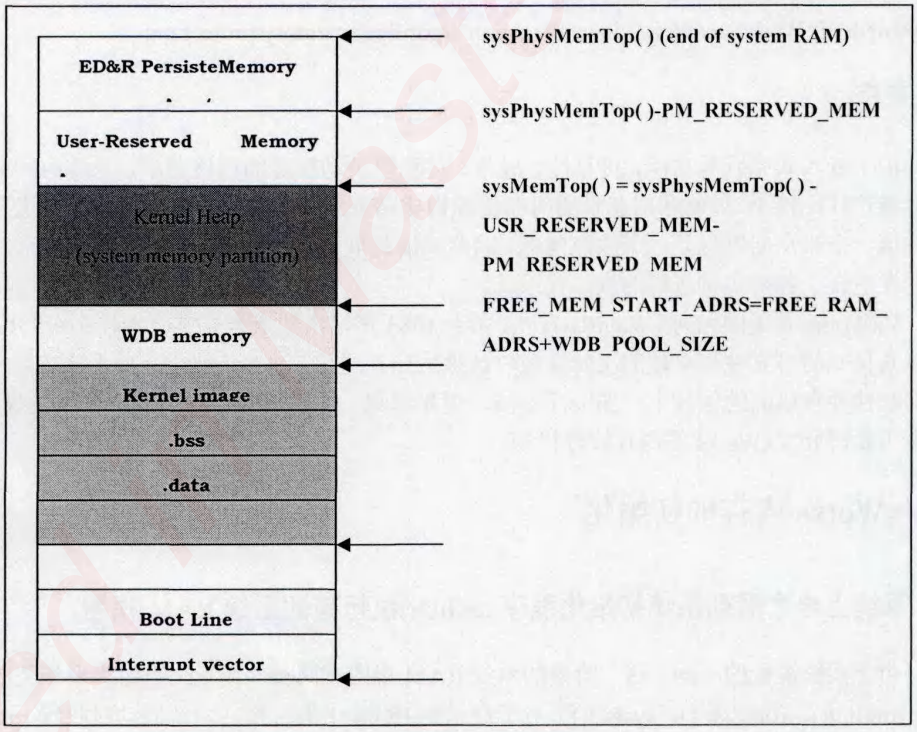

图 2 系统镜像最终加载完毕后系统 RAM 视图

3. VxWorks 中动态内存的分配与释放

嵌入式系统中的内存分配主要有两种方式：静态内存分配和动态内存分配。对于实时性和可靠性要求比较高的任务系统必须进行静态内存分配，但是这样会带来内存使用的浪费。

如系统内存视图中的 User-reserved memory 以及 Persistent memory，它们在系统初始化完成后大小就已经分配好了。User-reserved memory 作为独立于内核分区的内存供内核任务的特殊使用；Persistent memory 则用于实现错误报告和检测机制。随着应用复杂性的不断提高，很难估算一个应用程序所需要的内存量，使用静态内存分配时必须按照最坏的情况进行最大的配置，这样必然导致极大的浪费。因此有必要引入动态内存分配机制来增强内存管理的灵活性。

系统启动后所创建了唯一一个分区——系统内核分区（system memory partition），也称为内核堆栈（Kernel Heap）。主要用来为内核代码（包括内核库和组件，内核任务）的运行以及进程运行时的系统调用动态分配内存。VxWorks 中的动态内存的管理主要是通过 2 个函数库 memLib 以及 memPartLib 来完成的。第二部分主要围绕动态内存管理展开讨论。

3.1 内存分配

VxWorks6.0 以及以后的版本中的内存分配采取优先匹配算法（best fit search）。相比于早期版本中采用的最先匹配算法（first fit search），优先匹配算法采用的数据结构和算法更为复杂。在 VxWorks 中，空闲内存块被组织成不同大小的空闲双链。相同大小的空闲块被放置在同一个空闲链中，所有的空闲链被组织到一个 AVL 树中。在进行内存分配时，搜索二叉树，如果不存在满足分配需求的最小的内存块的 AVL 树结点，则返回 NULL，动态内存申请失败。如果存在这样的结点，就在结点指针指向的空闲链表中的表头结点所指向的内存块中进行内存分配。所选中的内存块在分配后如果剩余内存的大小足以创建一个空闲块（考虑到空闲块块头等的开销）则创建一个空闲块，此时 AVL 结点中如果已经存在和这个创建的空闲块的大小相同的空闲链则加入到链尾部，如果不存在就必须重新创建一个新的 AVL 树结点，相应的创建一个新的空链表并将这个新创建的结点加入链表中去，此时链中只有一个空闲块。实际上这种情况就是产生了内存碎片了。如果分配后剩余内存不足以创建一个空闲内存块就将这块剩余内存标记为已分配，也就是不进行内存的分割。在更新完空闲链表以及 AVL 树之后需要更新分配块的块头信息以及内存分区的相关统计信息。

这种分配算法相比于最先匹配算法的优势在于：首先减少了大块内存被分割的几率，从而减少了内存外部碎片；其次相同大小的结点被组织成一个空闲链表，这样减少了内存分配时的搜索次数；最后，对于平衡二叉树存在高效的搜索算法（算法复杂度为 O（log(n)），线性搜索算法的复杂度为 O（n））。这些都使得系统的动态内存分配时间更为确定，有利于提高嵌入式操作系统的实时性。下面是一个有着 5 个空闲块的系统中内存的组织示意图：

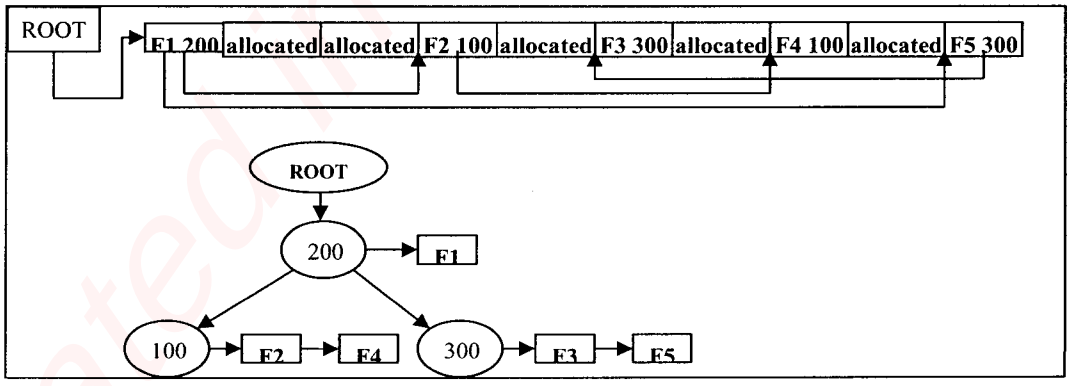

图 3AVL 树中空闲内存块的组织

系统中的内存分配函数的调用关系图如下所示：

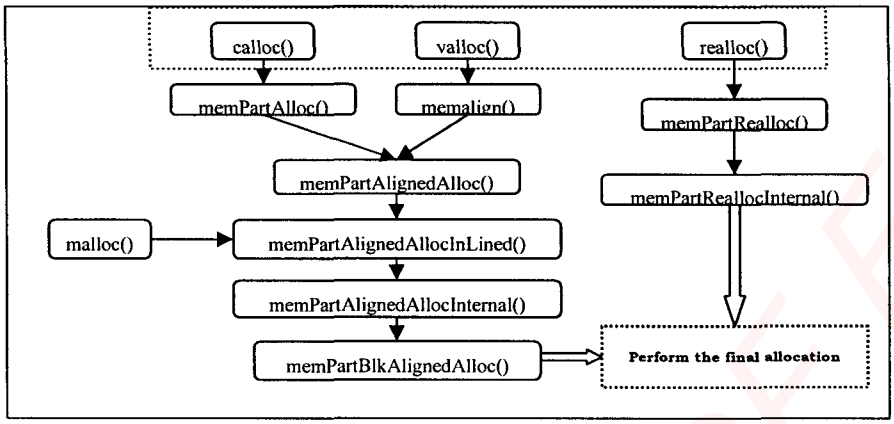

图 4 内存分配函数调用关系图

malloc()、calloc()以及 realloc()是传统的 ANIC 兼容的内存分配函数。其中 malloc()是从系统内存分区动态分配内存；calloc()用于数组的分配并将所得空间清零；realloc()用于重新分配一块内存；valloc()用于在系统内存分区中分配一块以页边界对齐的内存，对齐参数是与具体的硬件体系结构相关的。这些函数都是从系统内存分区分配内存，但是由于系统内存分区是临界资源，进行动态内存分配时可能由于得不到信号量从而导致申请内存的任务会被挂起，因此考虑到实时性在 ISRs 中是不能调用这些函数进行动态内存分配的。

优先匹配算法较之最先匹配算法减少了系统中内存碎片的产生，但是长时间的对内核分区进行动态内存的申请释放还是会使得系统中的内存碎片增加。为了减少内存碎片，同时进一步提高内存使用效率以及使用的灵活性，系统除了提供了对系统内存分区操作的相关函数外，还允许用户创建自己的内存分区。memPartCreate()允许用户在指定位置创建指定大小的内存分区。分区的内存可以来自于内核堆栈，可以是系统内存顶端，也可以来自于板外扩展内存。通过内存分区机制，隔离开系统内存分区和用户自己创建的分区，对分区的操作不影响到系统内存分区，可以减少内存碎片的产生。

3.2 内存释放

内存释放时，可以选择对内存块的一致性进行检测，如果此内存块已被破坏则释放失败。如果此内存块的相邻的内存块是空闲块必须进行合并，然后将合并后的空闲内存块加入到系统的空闲链表中去。相应的要更新 AVL 树以及内存分区的统计信息。内存释放的函数调用关系图：

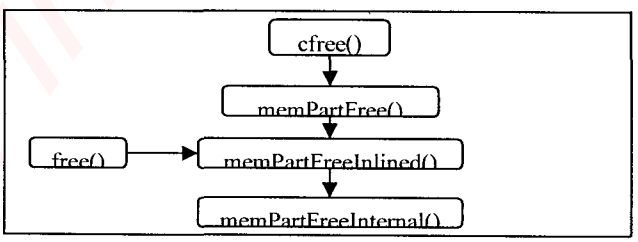

图 5 内存释放函数调用关系图

4.VxWorks 中的内存保护机制

VxWorks 是基于任务的嵌入式实时操作系统，应用程序是以内核任务的模式运行的，在特权模式下所有任务的代码运行在同一地址空间，任务能够快速共享系统中的绝大部分资源，这有效的保证了系统的实时性和灵活性。然而由于所有的任务共享同一个地址空间，

个潜在的风险就是任何一片内存区域（比如应用程序代码、数据、甚至是内核本身）都可能被软件漏洞或者应用程序错误所破坏。因此系统的正确运行取决于程序员编写完全正确的程序以及完全的软件测试。根据经验有时一个任务的崩溃可以导致整个系统的崩溃。

4.1 内存分配与释放时对内存错误的检测

VxWorks 为每一个内存块添加了块头，对已分配块和空闲块分别用数据结构 ALLOC_CHUNK_HDR 和 FREE_CHUNK_HDR 来进行管理（见图 6）。

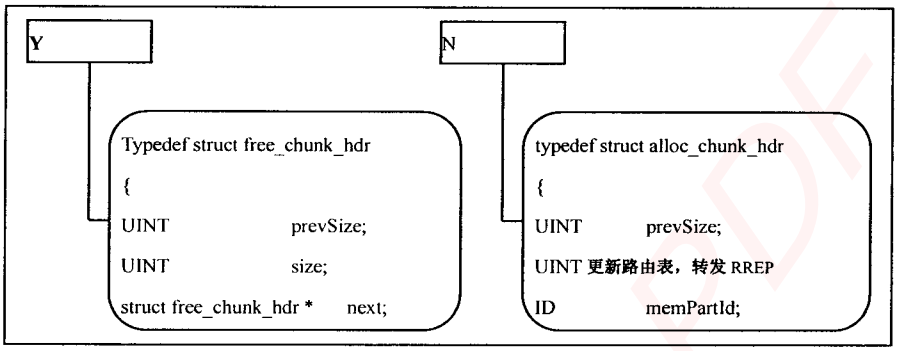

图 6 内存块块头结构

内存块在被分配时块头结构中记录了一些信息，这些信息在内存释放时可以用来进行内存的一致性检测。函数 memPartBlockIsValid()和 memPartBlockValidate()对内存块的有效性进行了验证，其中 memPartBlockIsValid() 在函数 memPartFreeInternal() 中被调用，memPartBlockValidate()在内存释放钩子函数（如果钩子函数已经安装）中被调用。如果内存块在释放前被破坏，那么释放函数将会返回失败。

4.2MMU 提供的内存保护

目前很多的处理器提供了片上或者板上的内存管理单元（MMU）。内存管理单元是许多操作系统用来访问物理内存的硬件单元，它通过提供内存保护功能来提高系统的可靠性，因此被越来越多的实时操作系统所采用。系统内存被划分为大小为 VAGE_PAGE_SIZE 的内存页，通过设置页属性并对属性进行操作实现内存保护功能。VxWorks 经过配置可以为 CPU 的 MMU 提供虚拟内存支持（见表 1）。

基于 MMU 的虚拟内存机制为系统提供了一个更可靠的运行环境，实现了如下一些内存保护功能（见表 1）。这些内存保护功能使得程序在开发调试阶段的效率得到了很大的提高，增加了系统的可靠性和鲁棒性，提高了嵌入式设备的整体性能。

表 1 虚拟内存支持以及由此实现的内存保护功能

虚拟内存支持	在系统启动时设置内核内存上下文 完成内存页面从逻辑地址到物理地址的映射 基于每一个内存页设置缓存属性 基于每一个内存页设置保护属性 设置页映射为有效或者无效 对 TLB 中的页面进行加锁解锁 页优化使能。动态改变页面的大小和属性
内存保护功能	任务栈的上溢出和下溢出检测 中断栈的上溢出和下溢出检测 不可执行任务栈设置 代码段写保护 异常向量表写保护

5 实时进程（RTP）中对内存的保护

目前的嵌入式操作系统的普遍采用基于分页的内存保护机制，它的原理是逻辑地址和物理地址一一映射，任何逻辑地址和它相对应的物理地址具有相同的值，这样逻辑地址空间与物理地址空间有相同的大小。而且由于所有的逻辑页都在内存中，所以无需页调度机制。这样可以避免由于页面调度所用的开销。在这种原理基础上除了传统的 VxWorks 内核任务执行模式之外，在 VxWorks6.0 以后的版本中提出了实时进程（RTP）的概念。即用户程序以进程的方式运行，这里的实时进程的概念类似于通用操作系统中的进程的概念，但是针对嵌入式实时系统进行了许多的优化。实时进程的一些特点见表 2。

表 2　RTP 的特点

每个进程拥有独立的地址空间，内核空间加入到每个 RTP 的上下文中，降低系统调用开销。
非重叠的地址模型。所有 RTP 和内核共享同一个 4G 的逻辑地址空间。
不同上下文之间的访问保护通过保护属性实现
RTP 镜像是浮动镜像，加载时 RTP 的地址由虚存管理器决定。
不支持页面交换，保证了系统的实时性。
实现内存保护。通过使能 MMU，实现对代码段/数据段/堆栈的保护。
实现内核隔离。通过系统调用，TRAP 指令使用内核功能。不需要静态链接。
RTP 本身不被调度，RTP 的任务在系统范围内被调度执行。
RTP 结束后，它所占有的资源（包括动态申请的内存）被回收，防止了内存泄露的产生。

RTP 使得内核可以免受用户程序的影响，同时用户程序之间也互不影响，保证了实时系统的可靠性。

结束语

本文研究了嵌入式实时操作系统 VxWorks 的内存管理机制。主要讨论了系统启动过程中的系统 RAM 视图，系统中动态内存的分配与释放，最后讨论了 VxWorks 的内存保护机制。嵌入式操作系统的实时性和可靠性等很大程度上依赖于系统内存的有效管理，VxWorks 是一个可裁剪的微内核的实时操作系统，针对不同的具体应用可以对内存管理机制进行相应的配置来达到最佳的性能。比如通过提供对 MMU 的支持来增强内存保护功能，可以通过配置实施进程（RTP）组件隔离用户与用户任务、用户与内核任务从而实现一定的内存保护功能。甚至可以考虑进一步的内存保护，比如可以将内存泄露检测与垃圾回收模块设计到系统中，在运行过程中可以根据系统对实时性的不同要求选择开启或者关闭此项功能，从而实现更加灵活的内存管理。

参考文献：

[1]　陈洋，胡向宁，杨坚华. VxWorks 下的内存管理[J]. 计算机工程, 2007, 33(8).
[2]　柴继国. 嵌入式系统内存管理的研究与实现[D]. 成都：电子科技大学, 2006 .
[3]　Wind River Systems, Inc. Memory Allocation in VxWorks 6.0, 2005
[4]　Wind River Systems, Inc. VxWorks_kernel_programmers_guide_6.6, 2007.
[5]　Wind River Systems, Inc. VxWorks_kernel_api_reference_vol1_6.6, 2007.

作者简介：
李加泗（1986-），男，硕士研究生生，主研方向：嵌入式操作系统；
宁洪，教授。

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究
23. IEEE1588 精密时钟同步关键技术研究
24. GPS 信号发生器射频模块的一种实现方案
25. 基于 CPCI 接口的视频采集卡的设计
26. 基于 VPX 的 3U 信号处理平台的设计
27. 基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计
28. AT89C52 单片机与 ARINC429 航空总线接口设计
29. 基于 CPCI 总线多 DSP 系统的高速主机接口设计
30. 总线协议中的 CRC 及其在 SATA 通信技术中的应用
31. 基于 FPGA 的 SATA 硬盘加解密控制器设计
32. Modbus 协议在串口通讯中的研究及应用
33. 高可用的磁盘阵列 Cache 的设计和实现
34. RAID 阵列中高速 Cache 管理的优化

35. 一种新的基于 RAID 的 CACHE 技术研究与实现
36. 基于 PCIE-104 总线的高速数据接口设计
37. 基于 VPX 标准的 RapidIO 交换和 Flash 存储模块设计
38. 北斗卫星系统在海洋工程中的应用
39. 北斗卫星系统在远洋船舶上应用的研究
40. 基于 CPCI 总线的红外实时信号处理系统
41. 硬件实现 RAID 与软件实现 RAID 的比较
42. 基于 PCI Express 总线系统的热插拔设计
43. 基于 RAID5 的磁盘阵列 Cache 的研究与实现
44. 基于 PCI 总线的 MPEG2 码流播放卡驱动程序开发
45. 基于磁盘异或引擎的 RAID5 小写性能优化
46. 基于 IEEE1588 的时钟同步技术研究
47. 基于 Davinci 平台的 SD 卡读写优化
48. 基于 PCI 总线的图像处理及传输系统的设计
49. 串口和以太网通信技术在油液在线监测系统中的应用
50. USB30 数据传输协议分析及实现
51. IEEE 1588 协议在工业以太网中的实现
52. 基于 USB30 的设备自定义请求实现方法
53. IEEE1588 协议在网络测控系统中的应用
54. USB30 物理层中弹性缓冲的设计与实现
55. USB30 的高速信息传输瓶颈研究
56. 基于 IPv6 的 UDP 通信的实现

VxWorks：

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植

15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发
21. 基于 VxBus 的驱动程序架构分析
22. 基于 VxBus 的高速数据采集卡驱动程序开发
23. Vxworks 下的冗余 CAN 通讯模块设计
24. WindML 工业平台下开发 S1d13506 驱动及显示功能的实现
25. WindML 中 Mesa 的应用
26. VxWorks 下图形用户界面开发中双缓冲技术应用
27. VxWorks 上的一种 GUI 系统的设计与实现
28. VxWorks 环境下 socket 的实现
29. VxWorks 的 WindML 图形界面程序的框架分析
30. VxWorks 实时操作系统及其在 PC104 下以太网编程的应用
31. 实时操作系统任务调度策略的研究与设计
32. 军事指挥系统中 VxWorks 下汉字显示技术
33. 基于 VxWorks 实时控制系统中文交互界面开发平台
34. 基于 VxWorks 操作系统的 WindML 图形操控界面实现方法
35. 基于 GPU FPGA 芯片原型的 VxWorks 下驱动软件开发
36. VxWorks 下的多串口卡设计

Linux：

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核

15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）
17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究
22. Linux TCP IP 协议详解
23. Linux 桌面环境下内存去重技术的研究与实现
24. 掌握 Android 7.0 新增特性 Quick Settings
25. Android 应用逆向分析方法研究
26. Android 操作系统的课程教学
27. Android 智能手机操作系统的研究
28. Android 英文朗读功能的实现
29. 基于 Yocto 订制嵌入式 Linux 发行版
30. 基于嵌入式 Linux 的网络设备驱动设计与实现
31. 如何高效学习嵌入式
32. 基于 Android 平台的 GPS 定位系统的设计与实现
33. LINUX ARM 下的 USB 驱动开发
34. Linux 下基于 I2C 协议的 RTC 驱动开发
35. 嵌入式下 Linux 系统设备驱动程序的开发
36. 基于嵌入式 Linux 的 SD 卡驱动程序的设计与实现
37. Linux 系统中进程调度策略
38. 嵌入式 Linux 实时性方法
39. 基于实时 Linux 计算机联锁系统实时性分析与改进
40. 基于嵌入式 Linux 下的 USB30 驱动程序开发方法研究
41. Android 手机应用开发之音乐资源播放器
42. Linux 下以太网的 IPv6 隧道技术的实现
43. Research and design of mobile learning platform based on Android

Windows CE:

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发

8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究
17. Windows CE 环境下无线网卡的自动安装
18. 基于 Windows CE 的可视电话的研究与实现
19. 基于 WinCE 的嵌入式图像采集系统设计
20. 基于 ARM 与 WinCE 的掌纹鉴别系统
21. DCOM 协议在网络冗余环境下的应用
22. Windows XP Embedded 在变电站通信管理机中的应用

PowerPC：

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现
9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发
10. 基于 MPC8313E 嵌入式系统 UBoot 的移植
11. 基于 PowerPC 处理器 SMP 系统的 UBoot 移植
12. 基于 PowerPC 双核处理器嵌入式系统 UBoot 移植
13. 基于 PowerPC 的雷达通用处理机设计
14. PowerPC 平台引导加载程序的移植
15. 基于 PowerPC 嵌入式内核的多串口通信扩展设计
16. 基于 PowerPC 的多网口系统抗干扰设计
17. 基于 MPC860T 与 VxWorks 的图形界面设计
18. 基于 MPC8260 处理器的 PPMC 系统
19. 基于 PowerPC 的控制器研究与设计
20. 基于 PowerPC 的模拟量输入接口扩展

21. 基于 PowerPC 的车载通信系统设计

ARM：

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现
14. 基于 S3C6410 处理器的嵌入式 Linux 系统移植
15. CortexA8 平台的μC-OSⅡ及 LwIP 协议栈的移植与实现
16. 基于 ARM 的嵌入式 Linux 无线网卡设备驱动设计
17. ARM S3C2440 Linux ADC 驱动
18. ARM S3C2440 Linux 触摸屏驱动
19. Linux 和 Cortex-A8 的视频处理及数字微波传输系统设计
20. Nand Flash 启动模式下的 Uboot 移植
21. 基于 ARM 处理器的 UART 设计
22. ARM CortexM3 处理器故障的分析与处理
23. ARM 微处理器启动和调试浅析
24. 基于 ARM 系统下映像文件的执行与中断运行机制的实现
25. 中断调用方式的 ARM 二次开发接口设计
26. ARM11 嵌入式系统 Linux 下 LCD 的驱动设计
27. Uboot 在 S3C2440 上的移植
28. 基于 ARM11 的嵌入式无线视频终端的设计
29. 基于 S3C6410 的 Uboot 分析与移植
30. 基于 ARM 嵌入式系统的高保真无损音乐播放器设计

Hardware：

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计
10. 基于 COM Express 的回波预处理模块设计
11. 基于 X86 平台的简单多任务内核的分析与实现
12. 基于 UEFI Shell 的 PreOS Application 的开发与研究
13. 基于 UEFI 固件的恶意代码防范技术研究
14. MIPS 架构计算机平台的支持固件研究
15. 基于 UEFI 固件的攻击验证技术研究
16. 基于 UEFI 的 Application 和 Driver 的分析与开发
17. 基于 UEFI 的可信 BIOS 研究与实现
18. 基于 UEFI 的国产计算机平台 BIOS 研究
19. 基于 UEFI 的安全模块设计分析
20. 基于 FPGA NiosⅡ的等精度频率计设计
21. 基于 FPGA 的 SOPC 设计
22. 基于 SOPC 基本信号产生器的设计与实现
23. 基于龙芯平台的 PMON 研究与开发
24. 基于 X86 平台的嵌入式 BIOS 可配置设计
25. 基于龙芯 2F 架构的 PMON 分析与优化
26. CPU 与 GPU 之间接口电路的设计与实现
27. 基于龙芯 1A 平台的 PMON 源码编译和启动分析

Programming:

1. 计算机软件基础数据结构 - 算法
2. 高级数据结构对算法的优化
3. 零基础学算法
4. Linux 环境下基于 TCP 的 Socket 编程浅析
5. Linux 环境下基于 UDP 的 socket 编程浅析
6. 基于 Socket 的网络编程技术及其实现