



普通高等教育“十一五”国家级规划教材

嵌入式操作系统原理与应用

吴旭光 何军红 编著



化学工业出版社

本书以嵌入式系统的开发为核心,重点讲述了嵌入式技术中的嵌入式操作系统,并且考虑到嵌入式系统开发的特点,也介绍了嵌入式系统的硬件平台和系统开发技术。这样读者既能够深入地学习实时多任务嵌入式操作系统,也能对嵌入式系统和嵌入式系统开发有一个比较全面的学习和理解。

本书的主要内容有:嵌入式操作系统的概念、嵌入式系统的硬件构成、ARM 体系结构、嵌入式实时操作系统内核、任务间通信、中断处理与定时机制、I/O 子系统、VxWorks 文件系统、网络系统、内存管理、系统引导程序及调试环境的建立和嵌入式系统的开发等。

本书语言流畅、条理清晰,内容全面且深入浅出,并结合作者多年的实际开发经验编写而成。本书可作为高等理工院校自动化、电气工程及其自动化、计算机应用、电子信息工程、测控技术与仪器、电子科学与技术等相关专业的本科生和研究生教材,也可供从事嵌入式系统开发、系统控制的科研人员和工程师参考。

图书在版编目(CIP)数据

嵌入式操作系统原理与应用 / 吴旭光,何军红编著. 北京:
化学工业出版社, 2007.7

普通高等教育“十一五”国家级规划教材

ISBN 978-7-122-00819-0

I. 嵌… II. ①吴…②何… III. 实时操作系统-高等
学校-教材 IV. TP316.2

中国版本图书馆 CIP 数据核字 (2007) 第 101226 号

责任编辑: 唐旭华

文字编辑: 钱 诚

责任校对: 徐贞珍

装帧设计: 韩 飞

出版发行: 化学工业出版社 (北京市东城区青年湖南街 13 号 邮政编码 100011)

印 刷: 北京永鑫印刷有限责任公司

装 订: 三河市万龙印装有限公司

787mm×1092mm 1/16 印张 18½ 字数 489 千字 2007 年 8 月北京第 1 版第 1 次印刷

购书咨询: 010-64518888 (传真: 010-64519686) 售后服务: 010-64518899

网 址: <http://www.cip.com.cn>

凡购买本书, 如有缺损质量问题, 本社销售中心负责调换。

定 价: 29.80 元

版权所有 违者必究

前 言

当前,随着计算机科学与技术的发展,计算机和网络已经全面渗透到人们的日常生活、工业过程以及许多领域。对于每个人来说,后 PC 时代的需要不再仅仅是那种放在桌面上处理文档、进行科学计算、工程管理和生产控制的台式计算机。而是包括各种 PDA、MP3、网络家电、智能家电,也包括诸如工业机器人、智能仪表、监测监控和复杂的武器系统等设备,所有这些设备都将拥有至少一颗“芯”。例如,现代豪华汽车,德国 BMW 的 7 系列车,其内部安装有近 70 个控制器,主要控制软件大于 60MB。而在这些控制器、设备和仪器中负责管理 CPU 和用户应用程序的就是操作系统。但是台式计算机的操作系统和人们说到的这些设备中的操作系统有非常大的区别,常将后一种操作系统称为嵌入式操作系统,也是本书所要讲解的操作系统。

嵌入式系统技术是建立在操作系统、计算机结构和组成、计算机接口技术、计算机语言、电子线路、数字电路、控制理论、计算方法和计算机技术等学科上的一门综合性很强的技术科学。目前,大学中许多专业都已经将有关嵌入式系统的课程作为其专业基础课程开设。

随着嵌入式技术的普及和应用范围的拓宽,嵌入式系统的复杂性和规模都在提高。尤其是近 10 年来,嵌入式实时操作系统得到了飞速的发展,从支持 8 位处理器到 16 位、32 位甚至 64 位。从支持单一品种的微处理器芯片到支持多种微处理器芯片,从只有实时内核到除了内核外,还提供其他功能模块,如高速文件系统、TCP/IP 网络系统、GUI 系统等。另外又因为嵌入式系统具备硬件平台和软件平台的多样性和应用个性化的特点,因此包括嵌入式系统硬件和软件产品都呈现出一种高度细分的市场格局。即不可能像台式机的硬件结构和 Windows 操作系统那样,出现一种产品能够垄断整个市场。嵌入式系统的这个特点不但为我国的嵌入式产品的研制和开发提供了一个难得的发展机遇,也决定了市场对嵌入式系统技术开发人员的大量需求。为此,目前全国各高校大多已经开设嵌入式系统类的课程。

这次在西北工业大学教务处和化学工业出版社的大力支持下,也参照了国内外的许多教材和专著,结合西北工业大学自动化类专业的教学特点,以及作者对嵌入式系统的开发经验,编写这部教材。考虑到本书的读者大多数是嵌入式系统的应用者,非计算机专业毕业,没有学习过计算机操作系统的课程。因此,本书不但深入浅出地介绍计算机操作系统有关的基础知识,为帮助读者建立嵌入式操作系统打下基础,并建立设计嵌入式操作系统的基本概念,而且还针对嵌入式系统开发的特点,向读者介绍了嵌入式硬件平台的基本结构。通过本书的学习,不但使读者能全面地学习到嵌入式操作系统的概念,也能够使读者对嵌入式系统有更全面的学习和理解,因此本书更加适合目前的教学大纲要求。

全书共分 11 章。主要内容如下。

第 1 章嵌入式操作系统的概念。本章介绍计算机操作系统有关的基础知识,为帮助读者建立嵌入式操作系统打下基础,并建立设计嵌入式操作系统的基本概念。本章的内容包括操作系统的概念和结构、嵌入式系统的原理和结构、嵌入式系统软件开发设计、商用嵌入式操作系统等。

第 2 章嵌入式系统的硬件构成。本章将向读者介绍嵌入式实时操作系统硬件的基本原理和

结构,包括系统的基本结构和开发环境、嵌入式实时操作系统的任务、任务编程和调度、任务间通信、中断处理机制等内容。

第3章 ARM 体系结构。本章主要介绍嵌入式硬件系统的组成、嵌入式处理器的特点和选型。本章以目前使用最多的嵌入式微处理器 ARM 系列为主,讲解嵌入式微处理器的结构和指令、嵌入式系统的存储器结构、输入/输出接口和设备等内容。

第4章嵌入式实时操作系统内核。本章介绍 RTOS 的结构和功能,包括内核、任务、任务的调度等。另外以 VxWorks 的实时内核 Wind 和任务编程接口为实例。本章也是各章的重要基础。

第5章任务间通信。实时内核的主要部分是任务,内核中多个任务之间不可避免地存在相互协同的关系,来完成一定的内核功能,这种协同最直观的就是任务间相互通信。嵌入式操作系统一般都会提供许多任务间通信的方法。本章对于每一种通信机制,均从主要数据结构、功能和有关的资源配置等方面讲解。希望通过本章的学习,读者能够了解实时多任务操作系统的同步、互斥与通信机制的基本原理,以便在做具体应用开发时能够正确、合理地使用。

对于嵌入式系统开发者来说,中断、异常和定时都是必须要面对的工作,它们不但是 RTOS 中非常重要的概念,也是利用 RTOS 开发正确、高效应用程序的重要基础。因此正确理解中断和定时的机制,对于用户开发操作系统和设计应用程序都是十分有意义的。因此在第6章向读者介绍操作系统内核的两个主要功能:中断和定时,在此基础上,进一步结合 VxWorks 来阐述这些概念。

第7章介绍嵌入式系统中的 I/O 子系统。一般的操作系统提供 I/O 系统、文件系统和网络系统的主要目的,就是为应用层提供标准的设备驱动接口,以最大程度地屏蔽应用层对硬件的相关性,来减轻开发者对底层硬件和操作系统的依赖性。本章是先介绍嵌入式系统的基本 I/O 操作机制,最后还是以 VxWorks 的 I/O 系统为对象介绍实时内核的 I/O 系统的开发和应用。

第8章是 VxWorks 文件系统。所谓文件系统,是指操作系统中用来组织、存储和命名文件的结构。本章将介绍 dosFs 文件系统、rawFs 原始文件系统、tapeFs 磁带文件系统、CD_ROM 文件系统和目标服务器文件系统。

第9章内存管理。嵌入式系统开发者通常在基本 RTOS 提供的基础上制定内存管理功能。因此,了解操作系统的内存管理成为开发嵌入式系统的一个重要方面,对嵌入式系统的内存管理有充分的了解,对系统开发者也是有利的。这也是本章编排的初衷。

系统引导程序及调试环境的建立是第10章的内容。本章将从嵌入式系统的软件结构、系统启动流程、引导装载程序的概念和主要任务以及引导装载程序与板级支持包的对比等四个方面来讨论嵌入式系统的引导机制。最后将给出笔者在 s3c44b0x 开发板上开发 VxWorks 的板级支持包过程,并详细分析了开发流程,对于初次涉足嵌入式操作系统开发的读者提供很好的实例。

第11章涉及嵌入式系统的开发方法。本章首先介绍嵌入式系统的开发模式,重点讨论如何选择处理器和硬件开发平台、操作系统和开发环境等问题。这些对后续的嵌入式系统软件开发有很大的影响。第3小节介绍实时软件设计方法。第4小节介绍嵌入式系统的软件/硬件协同设计问题。最后,还要向读者介绍目前嵌入式系统的最新开发方法,即快速控制原型和硬件在回路的仿真。以便读者在今后的嵌入式系统开发中能够应用最新的技术和最新的方法来完成自己的设计。

嵌入式系统是一门涉及面较广的学科,就嵌入式系统本身来看,可分硬件平台和软件平台,

而软件平台又分为底层的操作系统内核和 I/O 驱动以及上层的应用程序。本书仅介绍嵌入式系统中软件平台的底层，即嵌入式操作系统的内核和 I/O 驱动。其他部分本书仅仅作了扼要的介
绍，在实际嵌入式系统开发中，读者还需要进一步学习嵌入式系统的硬件平台结构和上层应用程序的开发技术。

在本书的编写和使用过程中，西北工业大学自动化教研室和自动控制理论教研室的许多老师都曾给予了极大的帮助，许多与作者有过合作的研究所的工程师也提出过许多具体和中肯的意见。尤其是作者的研究生王慧、王欢、赵勋峰、王艳、张朝、苏娟、陈兴隆、张竞凯、刘鲁玉、张佐经等也都参与了本书编写的前期准备工作，在此表示衷心的感谢。许多使用过本书讲义的学生也曾经提出过许多宝贵的意见，也向他们表示深深的谢意。

本书的编写还参阅了大量的文献，这些文献给我们不少启迪，在此向引用这些文献的作者表示感谢。

最后，第一作者还要感谢他的夫人和女儿长期对他的教学和科研工作给予的支持和鼓励，在生活中给予的关怀，也正是这些支持、鼓励和关怀，不断激励作者的向上精神。

由于编者水平有限，不妥之处在所难免，敬请读者给予批评指正。

编 者

2007 年 6 月

目 录

1 嵌入式操作系统的概念	1
1.1 操作系统的概念和结构.....	1
1.2 操作系统的功能和主要特征.....	2
1.2.1 操作系统的功能.....	2
1.2.2 操作系统的主要特征.....	5
1.3 操作系统的结构.....	6
1.3.1 操作系统的内核.....	6
1.3.2 传统的操作系统结构设计模式.....	8
1.3.3 现代的操作系统结构设计模式.....	9
1.4 嵌入式实时操作系统.....	11
1.4.1 嵌入式实时操作系统的特点.....	11
1.4.2 嵌入式应用.....	12
1.4.3 嵌入式实时操作系统的发展.....	13
1.5 嵌入式实时系统的分类.....	15
1.5.1 按速度分类.....	15
1.5.2 按确定性分类.....	15
1.5.3 按软件结构分类.....	15
1.6 商用嵌入式实时操作系统.....	18
1.7 小结.....	19
思考题.....	19
2 嵌入式系统的硬件构成	20
2.1 嵌入式系统的总体结构.....	20
2.2 嵌入式系统硬件.....	22
2.2.1 嵌入式处理器.....	22
2.2.2 存储器.....	24
2.2.3 输入输出设备.....	25
2.3 嵌入式系统硬件开发相关技术.....	27
2.3.1 接口技术.....	27
2.3.2 总线.....	30
2.3.3 嵌入式系统开发常用的硬件 调试和编程技术.....	34
2.4 小结.....	37
思考题.....	38
3 ARM 体系结构	39
3.1 ARM 的结构和特性.....	39
3.1.1 ARM 处理器系列.....	39
3.1.2 ARM 处理器的特点.....	42
3.1.3 ARM 处理器模式.....	43
3.2 ARM 寄存器介绍.....	43
3.3 ARM 体系的异常中断.....	45
3.4 ARM 的存储系统.....	47
3.5 ARM 指令系统.....	48
3.6 ARM 指令基本寻址方式.....	50
3.7 ARM 指令集.....	52
3.8 ARM 汇编程序设计.....	55
3.8.1 ARM 汇编器所支持的伪指令.....	55
3.8.2 汇编语言的语句格式.....	56
3.8.3 汇编语言的程序结构.....	60
3.8.4 ARM 和 Thumb 混合编程简介.....	60
3.9 小结.....	61
思考题.....	62
4 嵌入式实时操作系统内核	63
4.1 嵌入式实时内核的结构和功能.....	63
4.1.1 嵌入式实时内核的结构.....	63
4.1.2 嵌入式实时内核的功能.....	65
4.2 嵌入式实时内核的任务管理.....	72
4.2.1 多任务机制.....	72
4.2.2 任务状态和状态转变.....	73
4.2.3 任务调度策略.....	74
4.2.4 任务异常处理.....	76

4.3	嵌入式实时内核的共享代码和重入	76	4.6.1	POSIX 和 Wind 调度方法的差异	88
4.4	VxWorks 操作系统结构	79	4.6.2	获得和设置 POSIX 任务优先级	89
4.4.1	VxWorks 操作系统	79	4.6.3	获得和显示当前的调度策略	90
4.4.2	VxWorks 的内核 Wind	81	4.6.4	获得调度参数: 优先级限制和时间片	91
4.5	VxWorks 任务编程接口	82	4.7	小结	91
4.5.1	任务控制函数	82	思考题		92
4.5.2	任务扩展函数	86			
4.5.3	VxWorks 系统任务	87			
4.6	POSIX 调度接口	88			
5	任务间通信	93			
5.1	实时内核任务间通信机制	93	5.5.2	普通消息队列	119
5.1.1	任务之间的关系	93	5.5.3	消息队列使用例子	121
5.1.2	共享内存	94	5.6	管道	124
5.1.3	互斥	95	5.6.1	概述	124
5.1.4	信号量	96	5.6.2	使用管道	125
5.1.5	内部数据结构	96	5.6.3	管道用于任务间通信	126
5.2	二进制信号量	97	5.7	POSIX 信号量	129
5.2.1	互斥	98	5.7.1	POSIX 信号量接口	129
5.2.2	同步	99	5.7.2	POSIX 和 Wind 信号量比较	130
5.2.3	二进制信号量用于任务间同步	100	5.7.3	未命名信号量使用	130
5.3	互斥信号量	104	5.7.4	命名信号量的使用	132
5.3.1	优先级倒置	105	5.8	POSIX 消息队列	134
5.3.2	删除安全	106	5.8.1	POSIX 和 Wind 消息队列比较	135
5.3.3	递归资源访问	106	5.8.2	POSIX 消息队列属性	135
5.3.4	互斥信号量用于任务间同步	107	5.8.3	显示消息队列属性	137
5.4	计数器信号量	112	5.8.4	用消息队列通信	137
5.4.1	计数器信号量	112	5.8.5	通知任务有消息在等待	140
5.4.2	计数器信号量用于任务间同步	113	5.8.6	POSIX 队列信号	143
5.5	消息队列	116	5.9	小结	144
5.5.1	概述	116	思考题		144
6	中断处理与定时机制	145			
6.1	中断管理	145	6.2.4	中断级异常	155
6.1.1	异常的定义	145	6.2.5	保留的最高中断级	156
6.1.2	中断的类型	146	6.2.6	最高中断级 ISRs 的约束	156
6.1.3	中断处理的过程	148	6.2.7	中断与任务的通信	156
6.1.4	实时内核的中断管理	150	6.3	时间管理	156
6.2	中断服务程序	152	6.3.1	硬件时钟设备	157
6.2.1	设置中断处理程序	152	6.3.2	时间管理	158
6.2.2	中断堆栈	154	6.4	看门狗机制和应用	161
6.2.3	ISR 的特殊限制	154	6.4.1	看门狗	161

6.4.2 利用看门狗处理任务时限	162	6.5.2 POSIX 内存上锁接口	167
6.5 POSIX 计时器和内存上锁	166	6.6 小结	167
6.5.1 POSIX 计时器	166	思考题	168
7 I/O 子系统	169		
7.1 基本 I/O 概念	169	7.3.3 打开和关闭	180
7.1.1 I/O 子系统	169	7.3.4 新建和删除	181
7.1.2 端口映射与内存映射 I/O 和 DMA	170	7.3.5 读写操作	181
7.1.3 文件名称	171	7.3.6 I/O 系统的其他操作	182
7.2 I/O 系统的内部结构	172	7.3.7 基于多文件描述符的挂起操作: select 功能	183
7.2.1 标准的 I/O 函数	173	7.4 缓冲 I/O	184
7.2.2 标准 I/O 函数到驱动程序函数的映射	174	7.5 其他的格式化 I/O 操作	185
7.2.3 设备	175	7.5.1 printf()、sprintf()、sscanf() 函数	185
7.2.4 文件描述符	176	7.5.2 printErr()和 fdprintf()	185
7.2.5 基本 I/O 的操作流程	177	7.5.3 信息记录	186
7.3 VxWorks 的 I/O 接口	178	7.6 基本 I/O 应用示例	186
7.3.1 文件描述符	178	7.7 小结	191
7.3.2 标准输入输出设备和标准错误输出设备	179	思考题	191
8 文件系统	192		
8.1 与 MS-DOS 系统兼容的文件系统	192	8.1.9 灾难恢复和磁盘卷的一致性	200
8.1.1 dosFs 文件系统的磁盘组织	193	问题	200
8.1.2 包含必要的 dosFs 文件系统组件	194	8.1.10 dosFs 文件系统支持的 I/O 控制参数	200
8.1.3 初始化 dosFs 文件系统	195	8.2 原始文件系统(rawFs)	201
8.1.4 初始化及使用支持 dosFs 的设备	195	8.3 磁带文件系统(tapeFs)	204
8.1.5 对磁盘和磁盘卷进行操作	195	8.4 CD-ROM 文件系统(cdromFs)	206
8.1.6 目录操作	196	8.5 目标机服务器文件系统(TSFS)	207
8.1.7 文件操作	196	8.6 文件系统应用示例	208
8.1.8 分配磁盘空间	199	8.7 小结	213
		思考题	213
9 内存管理	214		
9.1 引言	214	9.3 内存保护	220
9.2 内存管理机制	215	9.3.1 存储器管理单元 MMU	220
9.2.1 固定大小存储区管理	215	9.3.2 VxWorks 的虚拟内存接口	224
9.2.2 可变大小存储区管理	217	9.4 小结	227
9.2.3 VxWorks 动态内存管理机制	218	思考题	227
9.2.4 VxWorks 动态内存管理函数	219		

10 系统引导程序及调试环境的建立	228
10.1 嵌入式系统的软件结构.....	228
10.1.1 嵌入式操作系统的运行条件.....	229
10.1.2 系统的启动流程.....	230
10.2 嵌入式系统的引导装载程序	
bootloader.....	230
10.2.1 bootloader 作用分析及 一般实现.....	231
10.2.2 bootloader 的阶段 1.....	232
10.2.3 bootloader 的阶段 2.....	234
10.2.4 u-boot 简介.....	236
10.3 嵌入式系统板级支持包 BSP.....	237
10.3.1 BSP 概念.....	237
10.3.2 BSP 在嵌入式开发中的位置 和作用.....	237
10.3.3 BSP 以及 bootloader 与 PC 机主 板上的 BIOS 的关系与区别.....	238
10.4 VxWorks 的 BSP.....	238
10.4.1 VxWorks 的 BSP 开发.....	240
10.4.2 VxWorks 的 BSP 启动流程.....	242
10.5 基于 S3C44B0X 开发板的 VxWorks 的 BSP 定制.....	244
10.5.1 S3C44B0X 开发板板上资源.....	244
10.5.2 BSP 包的初步定制.....	245
10.5.3 在 BSP 包中进行串行通信 的建立.....	248
10.6 小结.....	258
思考题.....	258
11 嵌入式系统的开发	259
11.1 嵌入式系统开发模式.....	259
11.1.1 嵌入式系统开发模式概述.....	259
11.1.2 处理器及硬件开发平台的 选择.....	261
11.1.3 操作系统的选择.....	263
11.1.4 开发环境的选择.....	266
11.2 嵌入式系统的硬件/软件协同设计 技术.....	267
11.3 集成开发环境 Tornado.....	269
11.3.1 Tornado 集成开发环境简述.....	269
11.3.2 Tornado 集成开发工具.....	270
11.3.3 VxWorks 的交叉编译开发 环境.....	271
11.4 MATLAB/RTW 系统开发环境.....	273
11.4.1 MATLAB/RTW 系统概述.....	273
11.4.2 RTW 生成代码结构与外部 模式.....	275
11.4.3 RTW Tornado 目标环境.....	276
11.4.4 RTW 程序创建过程.....	277
11.5 飞行器控制系统仿真及性能分析.....	278
11.5.1 RTW 嵌入式代码在 VxWorks/ Tornado 环境下的实现过程.....	278
11.5.2 控制系统仿真.....	282
11.6 小结.....	282
思考题.....	283
参考文献	284

1 嵌入式操作系统的概念

考虑到本书的读者大多数是嵌入式系统的应用者，非计算机专业毕业，没有学习过计算机操作系统的课程。但是嵌入式操作系统也是属于计算机操作系统的一种，它是在台式计算机操作系统的基础上发展起来的。因此，本章先介绍计算机操作系统有关的基础知识，使读者能够初步了解操作系统的基本结构和原理，进而了解并建立设计嵌入式操作系统的基本概念。

本章的内容包括操作系统的概念和结构、嵌入式系统的原理和结构、嵌入式系统软件开发、设计、商用嵌入式操作系统等。

1.1 操作系统的概念和结构

今天，随着计算机科学与技术的发展，计算机和网络已经全面渗透到人们的日常生活、工业过程以及许多领域。对于每个人来说，后 PC 时代或普适计算时代（pervasive computing）的需要不再单单是那种放在桌面上处理文档、进行科学计算、工程管理和生产管理的台式计算机，而是包括各种 PDA、MP3、网络家电、智能家电，也包括能够处理和控制诸如工业机器人、智能仪表、监测监控和复杂的武器系统等设备的“计算机”，所有这些“计算机”也都必须至少有一颗 CPU “芯”和各种不同的软件程序。而在这类计算机、设备和仪器中负责管理 CPU 和用户应用程序的就是操作系统。但是台式计算机的操作系统和说到的这些设备中的“计算机”的操作系统有非常大的区别，人们将后一种操作系统称为嵌入式操作系统，也是本教材主要的内容。

台式机操作系统和嵌入式操作系统虽然是两种不同的操作系统，但也有许多相同之处，另外嵌入式操作系统也是由台式机操作系统发展起来的。

无论是台式机，还是嵌入式系统，它们都是由两部分构成：系统硬件和系统软件。系统硬件是指构成系统所必须配置的全部设备。现代计算机系统硬件一般都包含有处理器、内存、磁盘、光盘、打印机、时钟、鼠标、键盘、显示器、网络接口以及其他输入输出设备。计算机系统硬件构成了计算机本身和用户执行任务的物质基础。只有系统硬件而无系统软件的计算机称为裸机，用户直接在裸机上应用是非常复杂的。

系统软件是一个计算机系统必须配置的程序，它包括汇编程序、编译程序、编辑程序、链接装配程序、系统应用程序以及系统维护程序等。按层次的概念，可将系统软件分成两个层次，如图 1-1 所示，即操作系统和应用软件。最为基本的系统软件是操作系统，因此有时也称操作系统为系统软件。

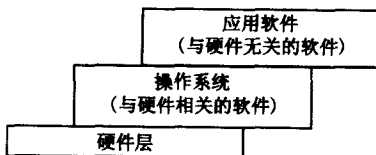


图 1-1 计算机系统的层次结构

从图 1-1 可以看出,整个计算机系统由三层构成,其中每一层不但代表一组功能,并为相邻的两层提供相应的接口。所谓接口,就是掩盖或封装低一层内部功能的实现细节,向上一层提供某些使用的约定。

操作系统层是硬件层的第一次扩充,在这一层上要实现操作系统的全部功能,并为应用软件层提供了相应的接口。显见,操作系统程序的开发是与系统的硬件紧密相关的,因此,也称操作系统是与硬件相关的软件。应用程序层是操作系统层的进一步扩充。也正是由于操作系统层的隔离,用户可以使用各种程序设计语言,在操作系统的支持下,编写并运行满足用户需要的各种应用程序,而不必关心系统的硬件结构和 I/O 口的驱动。

从图 1-1 可以看到,操作系统与上层应用程序的区别在于程序的内容和程序任务的不同。操作系统的任务就是在相互竞争的作业或程序之间有序地控制计算机系统资源的分配和管理,使计算机系统资源得到充分有效的利用,从而实现对计算机系统工作流程的控制。

显然,操作系统实际上是计算机系统的资源管理器,其首要任务是要跟踪资源的使用状况,满足用户程序对资源的请求,提高系统资源利用率,协调各程序对资源的使用冲突。

① 跟踪资源状态。时刻保持系统资源分配的全局信息,掌握系统资源的种类和数量、已分配和未分配的情况。

② 分配资源。处理对资源的使用请求,协调请求中的冲突,确定资源分配算法。有多个用户争用某个资源时,进行裁决。同时根据资源分配的条件、原则和环境决定是立即分配还是暂缓分配。

③ 回收资源。用户程序在资源使用完了之后要释放资源。此时,资源管理器应及时回收资源以便下次重新分配。

④ 保护资源。资源管理器负责对资源进行保护,防止资源被有意或无意地破坏。

系统资源的使用方法和策略决定了操作系统的规模、类型、功能与实现。

一般将计算机系统资源划分为四类:处理机、存储器、I/O 设备和信息(程序和数据)。针对这四类资源,可以为操作系统建立相应的四类管理器:处理机管理、存储管理、设备管理和信息管理(通常指文件系统)。

通过以上讨论,知道操作系统是计算机系统中最重要、最基本的软件。它管理计算机系统中的所有软、硬件资源,为用户使用计算机提供了方便。于是,可以这样给操作系统下一个定义:操作系统是控制和管理计算机系统的硬件和应用软件资源,合理地组织计算机工作流程和方便用户使用的程序和数据集合。

1.2 操作系统的功能和主要特征

1.2.1 操作系统的功能

无论是台式机的操作系统还是嵌入式操作系统的主要任务都是对系统中的硬件、软件实施有效的管理,以提高系统资源的利用率。计算机硬件资源主要是指 CPU、主存储器和外部设备。软件资源主要是指 I/O 驱动程序、文件系统和应用程序。相应操作系统的主要功能就有 CPU 管理、存储器管理、设备管理和信息管理。另外操作系统又要为上一层应用程序提供用户接口。本小节就从以上几个方面介绍操作系统的功能。

(1) CPU 管理

在早期的计算机系统中,或 8 位和 16 位嵌入式系统中,多采用单道作业或单用户的环境,CPU 为一个作业或一个用户所独占,这种方式的操作系统对 CPU 的管理十分简单。但对于 32

位计算机基本都采用多道程序或多用户的环境，操作系统要组织多个作业同时运行，这时 CPU 的管理就比较复杂。在多道程序环境下，CPU 的分配和运行都是以进程为基本单位的，因而对 CPU 的管理可归结为对进程的管理。

进程由三部分组成：一个可执行的程序；该程序所需的相关数据（变量、工作空间、缓冲区等）；该程序的执行上下文。操作系统对进程的管理包括以下几个方面。

① 进程控制。进程控制的主要任务是为作业创建进程、撤销已结束的进程和控制进程在运行过程中的状态转换。对于嵌入式系统来说，由于内存较小，因此及时撤销已结束的进程是非常重要的。

② 进程调度。进程调度的任务是从进程的就绪队列中，根据一定的优先级算法选择一个进程，把 CPU 分配给它，并为它设置运行现场，使之投入运行。

③ 进程同步。为使系统中的所有进程有条不紊地运行，操作系统必须设置进程同步机制，以协调系统中各进程的运行。

④ 进程通信。系统中的各进程一般都不是独立的，进程之间往往要交换信息，因此，在操作系统中一般都设置多种进程间的通信方式。

（2）存储器管理

存储器管理的主要任务是为多道程序的运行提供良好的环境，方便用户和进程可靠地使用存储器，提高主存的利用率。嵌入式系统使用的 CPU 基本都是精简指令集系统，有些 CPU 甚至没有内存管理单元（MMU），因此存储器管理难度更大。一般的存储器管理包括以下几个方面。

① 地址重定位。在多道程序设计环境下，每个作业是动态装入主存的，作业的逻辑地址必须转换为主存的物理地址，这一转换称为地址重定位。

② 存储器分配。存储器管理的主要任务是为每道程序分配内存空间，在作业结束时要收回它所占用的空间。

③ 存储器保护。保证每道程序都在自己的主存空间运行，各道程序互不侵犯，尤其是不能侵犯操作系统。

④ 存储器扩充。一般来说，主存的容量是有限的。在多道程序设计环境下往往感到主存容量不能满足用户作业的需要。为此操作系统存储器管理的任务是要扩充主存容量，但这种扩充是逻辑上的扩充，可以通过建立虚拟存储器系统来实现。虚拟存储器允许程序以逻辑方法寻址，而不用考虑物理上可获得的内存大小。当一个程序执行时，只有一部分程序和数据保存在内存中，其他部分则存储在磁盘上。

（3）设备管理

一个计算机系统的硬件，除了 CPU 和主存，其余都属于外部设备。外部设备种类繁多，物理特性相差甚大。因此，操作系统的设备管理往往很复杂，也是操作系统的关键。嵌入式操作系统由于中央处理机（MCU）一般都集成了较多的 I/O 口，因此 I/O 的驱动通常都是将大量的 I/O 设备精简成一种统一的 I/O 系统来进行管理，即对内核和应用开发者隐藏设备特定的信息，并且对系统外围 I/O 设备提供一个统一的访问方法。例如，在 VxWorks 操作系统中，对于应用程序，操作系统提供 7 个基本的与设备无关的 I/O 标准函数。设备管理主要包括以下几个方面。

① 缓冲管理。由于 CPU 和 I/O 设备的速度相差很大，为缓和这一矛盾，通常在设备管理中建立 I/O 缓冲区，而对缓冲区的有效管理便是设备管理的一项任务。例如，有的 CPU 具有直接内存访问 DMA 功能。

② 设备分配。当用户程序提出 I/O 请求后，设备管理程序要依据一定的策略和系统中设备情况将所需设备分配给它，设备用完后还要及时收回。

③ 设备处理。设备处理程序又称设备驱动程序，对于未设置通道的计算机系统，其基本任务通常是实现 CPU 和设备控制器之间的通信。即由 CPU 向设备控制器发 I/O 操作，并能接收由设备控制器的中断请求，给予及时的响应和相应的处理。对于设置了通道的计算机系统，设备处理程序还应能根据用户的 I/O 请求，自动构造通道程序。

④ 设备独立性和虚拟设备。设备独立性是指应用程序独立于物理设备，使用户编程与实际使用的物理设备无关。

虚拟设备的功能是将低速的独占设备改造为高速的共享设备。

(4) 文件管理

在现代计算机应用中，文件管理是基本功能之一。在操作系统中，实现这一基本功能的程序称为文件系统。它主要是进行信息的组织、管理、存取和保护。处理机管理、存储管理和设备管理都属于硬件资源的管理。软件资源的管理称为信息管理，即文件管理系统。

现代计算机系统或嵌入式系统中，总是把程序和数据以文件的形式存储在文件存储器中（如磁盘、光盘、磁带等）供所有用户或指定用户使用。为此，操作系统必须具备文件管理功能。文件管理的主要任务是对用户文件和系统文件进行管理，并保证文件的安全性。文件管理包括以下内容。

① 目录管理。为方便用户在文件存储器中找到所需文件，通常由系统为每一文件建立一个目录项，包括文件名、属性以及存放位置等，由若干目录项又可构成一个目录文件。目录管理的任务是为每一文件建立其目录项，并对目录项施以有效的组织，以方便用户按名存取。

② 文件读、写管理。文件读、写是文件管理的最基本功能。文件系统根据用户给出的文件去查找文件目录，从中得到文件在文件存储器上的位置，然后利用文件读、写指针，对文件进行读、写。

③ 文件存取控制。为了防止系统中断时文件被非法窃取或破坏，在文件系统中应建立有效的保护机制，以保证文件系统的安全性。

④ 文件存储空间的管理。所有的系统文件和用户文件都存放在文件存储器上，文件存储空间管理的任务是为新建文件分配存储空间，在一个文件被删除后及时释放所占用的空间，文件存储空间管理的目标是提高文件存储空间的利用率，并提高文件系统的工作速度。

(5) 用户接口

操作系统的另一个作用是掩蔽了计算机硬件的操作细节，它使用户或程序员与系统硬件隔离开来。要做到这一点，操作系统必须为用户或程序员提供相应的接口，通过使用这些接口达到方便地使用计算机硬件层的目的。

一般操作系统为用户提供了以下接口。

① 命令接口。命令接口也称作业控制级接口，分为联机命令接口和脱机命令接口。

联机命令接口是为联机用户提供的，它由一组键盘命令及其解释程序所组成。当用户在终端或控制台上打入一条命令后系统便自动转入命令解释程序，对该命令进行解释并执行。在完成指定操作后，控制又返回到终端后控制台，等待接收用户打入的下一条命令。这样，用户可以通过不断键入不同的命令达到自己作业的目的。

脱机命令接口是为批处理系统的用户提供的。在批处理系统中，用户不直接与自己的作业进行交互，而是使用作业控制语言（JCL）的语句，将用户对其作业控制的意图写出作业说明书，然后将作业说明书连同作业一起，以卡片的形式提交给系统。当系统调度到该作业时，通过解释程序对作业说明书进行逐条解释并执行。这样，作业一直在作业说明书的控制下运行，直到遇到作业结束语句时，系统停止该作业的执行。这种工作方式已成为历史，至今再也看不到了。

② 程序接口。程序接口是用户获取操作系统服务的惟一途径。程序接口由一组系统调用

组成。每一个系统调用都是一个完成特定功能的子程序。早期的操作系统（如 UNIX、MS-DOS 等），系统调用都是用汇编语言写成的，因而只有在用汇编语言写的应用程序中可以直接调用，而在高级语言及 C 语言中，往往提供与系统调用一一对应的库函数，应用程序通过调用库函数来使用系统调用。近年来推出的操作系统中，如 UNIX System V、OS/2 2.x 版本中，系统调用是用 C 语言编写的，并以函数的形式提供，从而可在用 C 语言编写的程序中直接调用。

③ 图形接口。以终端命令和命令语言方式来控制程序的运行固然有效，但给用户增加了不少负担，即用户必须记住各种命令，并从键盘输入这些命令以及所需数据来控制程序的运行。随着大屏幕高分辨率图形显示和多种交互式输入输出设备（如鼠标、光笔、触摸屏等）的出现，使得改变“记忆并键入”操作为图形接口形式成为可能。图形用户接口的目标是通过出现在屏幕上的对象直接进行操作，以控制和操作程序的运行。这种图形用户减轻或免除了用户记忆的工作量，其操作方式也从原来的“记忆并键入”改变为“选择并点取”，极大地方便了用户，并受到用户的普遍欢迎。

国际上为了促进图形用户接口（GUI）的发展，1988 年制定了 GUI 标准。到了 20 世纪 90 年代各种操作系统的图形用户接口普遍出现，如 Microsoft 公司的 Windows95、Windows98、Windows NT 等。

图形用户接口的主要构件是：窗口、菜单和对话框。

1.2.2 操作系统的主要特征

设置操作系统的目的在于提高计算机系统的效率，增强计算机系统的处理能力，充分发挥系统资源的利用率，方便用户的使用。为此，操作系统广泛采用并行操作技术，使多种硬件设备能并行工作。例如 I/O 操作和 CPU 计算同时进行，在内存中同时存在存取执行多道程序等。以多道程序设计为基础的现代操作系统具有以下主要特征。

（1）并发性

并发指的是在操作系统中存在着许多同时的或并行的活动。例如，在多道程序设计的环境下，各道程序同时在处理机上交替、穿插地执行。由并发而产生的一些问题是：如何从一个活动转到另一个活动、如何保护一个活动不受另一个活动的影响以及如何实现相互制约活动之间的同步。

（2）共享性

系统中存在的各种并发活动，要求共享系统的硬、软件资源。这样做的理由如下。

① 向各个用户分别提供足够的资源是浪费的，有时也是不可能的。

② 多个用户共享同一程序比向各个用户分别提供程序副本能节省存储空间，提高工作效率。

③ 几个用户或程序员在开发软件过程中为避免重复，应允许相互使用他人拥有的软件资源。

与共享有关的问题是资源的分配、对资源的同时存取以及保护程序免遭破坏等。

（3）虚拟性

虚拟是指将一个物理实体映射为若干个逻辑实体。前者是客观存在的，后者是虚拟的，是一种感觉性的存在，即主观上的一种想象。例如，在多道程序系统中，虽然只有一个 CPU，每一时刻只能执行一道程序，但采用多道程序技术后，在一段时间间隔内，宏观上是将一个物理的 CPU 虚拟为多个逻辑上的 CPU，并各自运行自己的程序。这种情况就是将一个物理的 CPU 虚拟为多个逻辑上的 CPU，逻辑上的 CPU 称为虚拟处理机。类似的还有虚拟存储器、虚拟设备等。

（4）不确定性

在操作系统中，不确定性有以下两种表现。

多道程序设计环境下, 程序按异步方式运行。也就是说每道程序在何时执行, 各个程序执行的顺序以及每道程序所需的时间可能是不确定的。操作系统必须能处理随时可能发生的事件。例如从外部设备来的中断、输入输出请求等, 程序运行发生的故障和发生的时间都是不可预测的。但是操作系统必须能处理这些不确定事件。然而, 这些不确定事件序列的数量可能相当大, 这给操作系统的设计带来了很大的复杂性。如果考虑不周、设计不当, 就会发生操作系统在处理这一事件序列时正确, 而在处理另一事件序列时出错。而且, 这种错误很难发现, 同样也会带来不确定性。

上述 4 点是现代操作系统的主要特征, 在进一步学习嵌入式操作系统时, 要引起注意。

1.3 操作系统的结构

前一节讲述了操作系统的外部特性, 对操作系统的功能、特性有了初步了解。本节研究操作系统的内部结构以及操作系统结构设计模型。

操作系统从内部结构来分析, 通常包括两部分: 一是内核; 二是核外部。核外部通常由一些实用程序组成, 例如 I/O 驱动、文件系统等。

1.3.1 操作系统的内核

(1) 内核在操作系统中的地位

嵌入式操作系统与台式机操作系统的最大区别是: 台式机的操作系统是不需要用户修改的, 在安装时仅仅是些参数的设置。而嵌入式操作系统用户在使用的时候必须要根据使用的目的和硬件的配置对操作系统进行裁减。因此, 嵌入式操作系统在设计的时候, 往往将那些最基本的操作、用户不需要修改的部分提出来。即操作系统将一些与任务密切相关的模块、运行频率较高的模块、关键性的数据结构以及公共的基本操作模块等组成操作系统的内核, 这部分一般不需要用户修改。而将用户需要经常裁减的部分, 例如文件系统、I/O 驱动、内存管理、网络协议等组成核外部。

以 VxWorks 为例, VxWorks 操作系统是一种功能最全的可以获得独立于处理器的实时系统。然而, VxWorks 带有一个相当小的微内核, 该内核仅提供多任务环境、进程间通信和同步功能。这些功能模块足够支持 VxWorks 在较高层次所提供的丰富的性能的要求。通常内核操作对于用户是不可见的。应用程序为了实现需要内核参与的任务管理和同步使用一些系统调用函数, 但这些函数的处理对于调用任务是不可见的。这种调用不像有些系统需要一个笨拙的跳转表接口, 用户需要通过一个整数来指定一个内核功能调用。图 1-2 给出了 VxWorks 的内核结构。

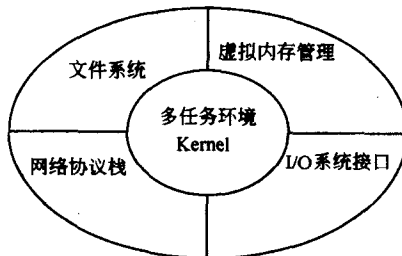


图 1-2 VxWorks 系统的内核结构

再例如, 开放式源码操作系统 $\mu\text{C}/\text{OS}$ 也是一个微内核结构, 它仅仅给用户提供了一个内核所必备的任务管理等基本功能。而大量的核外结构, 如 I/O 驱动则需要用户自己开发。

一般操作系统内核, 通常包括以下功能模块。

- ① 进程、线程及管理。
- ② 存储管理。
- ③ I/O 管理。
- ④ 文件系统。

对操作系统内核的定义和功能范围的划分, 各个操作系统有所不同。有的操作系统将全部系统功能都包括在内核中, 也有的操作系统将最基本的系统功能放入内核, 而将其余部分放在核外。

(2) 强内核与微内核

操作系统有两种内核组织形式: 一种是强内核, 另一种是微内核。一般台式机的操作系统大多采用的是强内核, 而目前大多数嵌入式操作系统系统多采用微内核。

操作系统采用强内核结构, 是基于传统的集中式操作系统的内核结构。在这种强内核的操作系统中, 系统调用是通过陷入内核实现的, 在内核完成所需要的服务, 最后返回结果给用户程序。许多基于 UNIX 的操作系统都是采用这种内核结构, 因为 UNIX 本身就具有一个大的强内核。

微内核结构体现了操作系统结构设计的新思想。微内核的设计目标是使操作系统的内核尽可能小, 除了最基本的服务外, 尽量将其他服务都放在核外。这样微内核作为一个不可少的核心, 为用户提供最基本、最必要的服务, 其他操作都以服务器的形式建立在微内核之上。一般微内核仅提供任务间通信机制、部分存储管理、有限的低级进程管理和调度、低级 I/O。

图 1-3 给出了强内核和微内核的示意。

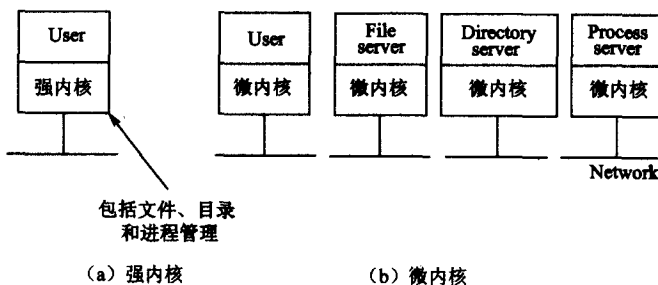


图 1-3 强内核和微内核

微内核与强内核相比具有以下优点。

① 灵活性。微内核短小精干, 它并不需要提供文件系统、目录系统、所有的进程管理以及许多系统调用的处理。

② 开放性。除了微内核以外, 其他操作系统服务都可运行在核外, 用户级服务都以服务器的形式建立在内核之上。这样所建立的操作系统称为服务器化的操作系统。系统的开发者基于这种结构框架, 可以方便地设计、开发、集成自己的新系统, 并易于保证其正确性, 从而使系统的设计能适应硬件平台发展的需要。

③ 可扩充性。采用微内核的操作系统, 对于实现、安装和调试一个新系统是很容易的, 因为加入或改变一个服务不需要停止系统和重新引导内核。利用这一点, 用户可以重写已存在的不满意的服务。

微内核技术与客户/服务器模式的结合是网络操作系统、分布式操作系统结构的新形式。

1.3.2 传统的操作系统结构设计模式

从操作系统结构设计观点来看,早期的操作系统,主要是单处理机操作系统,在结构设计方法上采用三种设计模式,即整体式结构设计模式、层次式结构设计模式、以管程为工具的结构设计模式。

本节仅介绍前两种设计模式。

(1) 整体式结构设计模式

简单地说,操作系统结构设计模式就是将系统所提供的特性、服务以及系统所执行的任务统一成体的一个概括性的框架。

早期的操作系统,常采用最简单的整体式结构模式,也叫单块式模式,其方法称为模块组合法。

整体式模式是把操作系统组织成一个过程(模块)集合。任一个过程可以调用其他过程。这种整体式结构的特点是不强调信息的隐蔽,即每个过程对其他过程是可见的。系统中每个过程都有一个定义完好的接口,包括入口参数和返回值,而且过程之间的调用不受任何约束。

在整体式系统中,为了构造最终的目标操作系统程序,开发人员首先将一些独立的过程进行编译,然后用链接程序将它们链接成一个整体,就成了一个操作系统。

在所有的最简单的整体式系统中,应用程序与操作系统本身还是分开的。操作系统代码运行在核心态,应用程序运行在用户态。把用户态与核心态分开的整体式操作系统如图 1-4 所示。

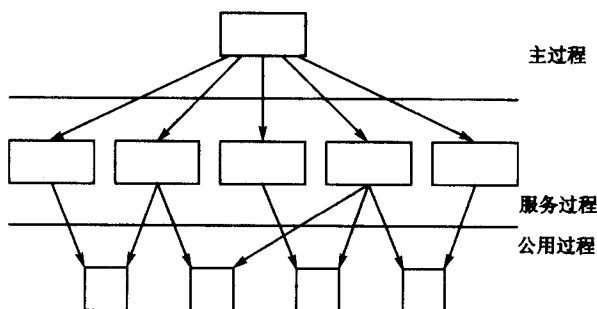


图 1-4 整体式操作系统结构模式

采用整体式结构模式具有明显的缺点:扩充这类系统非常困难,因为修改一个过程可能导致系统若干过程的修改,会产生看起来与系统无关的错误,操作系统的正确性难以保证。显见,这种结构是不适合嵌入式操作系统的。

(2) 层次式结构设计模式

层次式方法是把操作系统划分为若干层,每一层有若干模块。每个模块提供一组可被其他模块调用的功能。在任一特定层次上的代码只能调用较低层次上的代码。按此种模式构造的第一个操作系统是 E.W.Dijkstra 开发的 THE 系统(1968)。

该系统是一个批处理系统,它分为六层,见表 1-1。

表 1-1 一个层次式操作系统

层 次	功 能	层 次	功 能
5	操作员	2	操作员-进程通信
4	用户程序	1	内存和外存管理
3	输入输出管理	0	处理器分配和多道程序