一个微内核操作系统的设计与实现

王红玲, 吕 强, 褚亚铭
(苏州大学 计算机科学与技术学院, 江苏 苏州 215006)

摘 要: 设计并实现了一个运行在 Bochs 虚拟机上的微内核结构的操作系统, 详细描述了系统中进程管理、进程间通讯、基本内存管理、磁盘服务器以及文件服务器的设计和实现。并将实现的系统应用于安全路由器的开发中。本系统的实现将有利于从微观上观察操作系统的行为特征, 更好地学习、理解和实践微内核机制。同时, 也为面向特定应用开发嵌入式操作系统提供了实践经验。

关键词: 微内核; 操作系统; 虚拟机
中图分类号: TP31 **文献标识码**: A **文章编号**: 1000－7180(2008)04－0009－04

A Design and Implementation of a Microkernel Operating System

WANG Hong-ling, LV Qiang, ZHU Ya-ming
(School of Computer Science and Technology, Soochow University, Suzhou 215006, China)

Abstract: This paper designs and implements a microkernel operating system based on virtual machine Bochs. It describes the design and implementation of process management, IPC, basic memory management, file system server, disk server in detail. Then the system is applied on the development of security router. The process of the system development will be of benefit to those learners in learning operating system features and the mechanism of microkernel. Moreover, it provides practice experience in developing embedded operating system for specific application.

Key words: microkernel; operating system; virtual machine

1 引言

随着信息电器、移动设备、网络设备和工控仿真装置等嵌入式系统的蓬勃发展, 人们对嵌入式操作系统提出了新的要求. 与通用操作系统相比较, 嵌入式系统具有系统实时高效性、硬件的相关依赖性、应用的专用性等特点.

嵌入式操作系统由一个体积很小的内核及一些可以根据需要进行定制的系统模块组成, 目前绝大多数的嵌入式系统采用高性能的微内核设计. 因为微内核具有结构清晰、容易扩展等优点[1-2], 符合嵌入式系统的要求.

文中开发了一个微内核操作系统, 期望能够更好地应用微内核机制, 并利用虚拟机软件, 加快嵌入式系统开发的周期.

2 系统设计

2.1 总体设计

嵌入应用开发不可避免地与硬件高度耦合, 而操作系统的作用就是要尽可能地隔离应用与硬件. 那么作为微内核操作系统本身的设计, 也应该考虑到可移植性. 通常的设计是用一个硬件抽象层封装硬件特征, 选择 Bochs[3] 作为这种隔离层. 虚拟机的采用可以有效地提高开发效率, 缩短开发周期[4-5].

系统的逻辑结构如图 1 所示. 其中微内核直接运行于 Bochs 虚拟机之上, 它在完成对硬件的封装与抽象之后, 向应用层提供一个支撑环境, 而各个系统服务器与用户进程则运行于此支撑环境之上[6].

2.2 系统结构与模块划分

系统根据功能被划分为微内核与核外服务器这两大部分. 各个服务器与用户进程通过微内核这一

收稿日期: 2007－05－25

图 1 系统逻辑结构示意图

消息总线进行消息传递, 如图 2 所示. 整个系统采用了客户/服务器模式的微内核体系结构.

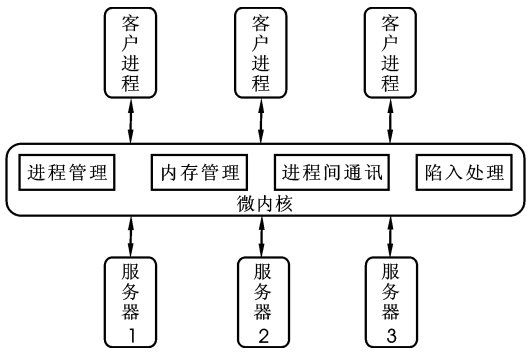

图 2 系统结构示意图

2.2.1 微内核部分

系统的微内核仅仅实现一些最为基本的服务, 它为整个系统的正常运作提供基础保障. 它被实现在核心级, 可以执行特权指令, 这部分实现高度依赖于底层的硬件. 微内核直接与底层硬件打交道, 并且通过少量的应用程序编程接口向上层提供一个内核的抽象. 微内核部分包括进程管理、进程间通讯、基本内存管理以及中断的响应框架. 主要功能如下:

(1) 进程管理. 负责实现进程的创建、进程的撤销以及最为核心的进程调度.

(2) 进程间通讯. 负责提供一种或多种进程间直接或间接通讯的方式.

(3) 基本内存管理. 负责对物理内存进行分配与回收, 以及对进程虚拟地址空间的管理.

(4) 陷入处理. 负责整个系统的系统调用、异常与中断响应框架的处理, 这是系统其余部分正常工作的基础.

2.2.2 服务器部分

服务器部分实现的是高层服务的提供者, 以服务器的形式存在于用户空间中, 并且采用客户机/服务器的方式与普通用户进程进行通讯. 服务器通过消息中指定的操作符与参数调用执行本部分实现的系统服务. 采用这种设计方案可以避免一个服务器的行为干扰同一系统中的其他模块, 每个服务器仅仅公开必须要让外界知道的接口即访问方式而将具体的实现细节隐藏了起来. 目前已 0 实现了以下两个服务器:

(1) 文件系统服务器. 将接收到的对于逻辑文件的处理转化成对存储于物理介质上的物理文件的处理, 为用户进程提供一个有效的文件访问方式.

(2) 磁盘服务器. 将接收到的磁盘读写消息转换成对磁盘控制器发送命令的操作, 磁盘服务器通过隐藏底层硬件的具体操作向用户进程提供一种有效的磁盘访问方式.

系统开发采用了 Linux 平台, 因为 Linux 环境下有着比较齐全的系统开发工具, 比较适合做系统层开发平台; 另一方面 Bochs 虚拟机也有相应的运行在 Linux 平台上的版本.

3 微内核关键机制的实现

3.1 基本内存管理

本系统中内核为每个进程分配的逻辑地址空间大小为 64M, 详细的布局如图 3 所示.

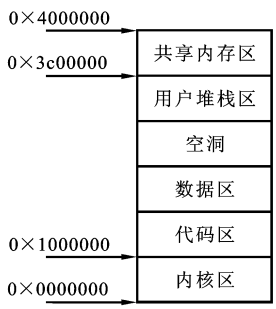

图 3 进程地址空间分配示意图

微内核中的地址映射部分的思路: 以 Bochs 模拟的 i386 处理器的分页存储管理为基础实现一个两层的页式存储管理模型. 逻辑地址转换成物理地址的过程如下: 根据 CR3 寄存器中的值取得页目录表, 并以线性地址中高 10 位的值为索引取得相应页表. 接着以线性地址中间 10 位的值为索引取得页面描述项, 将页面描述项中给出的页面基地址与线性地址中的最低 12 位相加获得对应的物理地址.

对于系统中主内存区的物理页面, 微内核使用了位图的方式来对其进行管理. 当可执行程序代码被加载到内存时, 内核会根据可执行代码的大小为其分配相应数量的内存页面以及相应的用于管理物理页面的页表与页目录表. 同时没有为用户堆栈分配任何的内存, 因此该进程运行以后当执行堆栈操作时, 由于用于映射堆栈的页表项尚未建立, 所以处理器会触发一次缺页异常. 微内核向内核申请获得一个空白物理页面, 对其地址作一番处理以后将其

写入相应的页表中.

3.2　进程管理

进程管理主要负责进程的创建、调度以及进程撤销等.

3.2.1　进程创建

每当一个新任务被创建时微内核会为其分配并且维护一个进程控制块(PCB)数据结构. 在设计进程控制块结构时, 强调并突出内核态与用户态的概念, 将内核态与用户态之间切换的纽带——系统堆栈放到了进程控制块(见图 4)中.

```
struct Task_Struct{
    unsigned int  *stackTop;
    unsigned int  *sysStackTop;
    enum TaskStatus status;
    int pageTable;
    struct port_s port;
    unsigned int stack[STACKSIZE];
    unsigned int sysStack[STACKSIZE];
    struct AddressSpace addrSpace;
    char name[MAXFILENAMELENGTH];
    int taskID;
};
```

图 4　进程控制块定义

在分配进程控制块后, 还需要对其中的结构作进一步初始化. 其中最为关键的就是对系统堆栈的初始化, 因为这与新创建的进程是怎样被调度执行的紧密关联.

3.2.2　进程的调度与切换

采用了软件来完成进程切换. 并采用了基于优先级的 FIFO 调度算法.

进程切换所涉及的最基本的动作就是保存上一个进程的 CPU 现场以及恢复下一个准备运行进程的CPU 现场, 这部分的实现是高度依赖底层硬件的. 由于本系统 PCB 的简化设计, 进程切换的实现较为简单.

当进程结束时微内核负责回收其占用的系统资源. 具体体现在, 根据进程任务控制块信息释放掉占用的相关资源, 主要是地址空间的数据结构以及存于端口上的尚未处理的消息. 微内核将进程占用的 PCB 置为可用并且重新调度新的任务投入运行.

3.3　进程间通讯

微内核的一个重要任务是负责并监督消息在各个进程之间的流动与传递. 本系统中, 设计了两种有代表性的进程间通讯机制即消息传递与共享内存.

在综合考虑简化性和有效性这两点以后本系统中接收被设计成阻塞式的, 即当进程执行接收原语时若发现端口为空, 则进入阻塞状态; 发送被设计成同步发送, 即当进程执行发送原语后立即进入阻塞状态, 只有当接收者收到该消息时, 发送进程才会再度进入就绪状态; 同时接收者可以不指定发送者, 而是接收来自任何一个想要与其通讯进程的消息, 但是发送者必须指定目标进程. 实现时, 内核中维护着一个消息池. 每当进程执行发送原语时内核首先作合法性检查, 通过合法性检查以后内核从消息池中分配一个消息控制块, 成功后将存在于用户空间的消息头连同消息正文一起复制到内核空间的消息控制块中, 并根据收发者调整消息控制块中其他一些管理信息, 最后将该内核消息块加入到目标进程端口所管理的消息队列中去, 并且更新端口的相应描述结构.

本系统实现共享内存时, 客户程序需要指定一块内存区域的地址与大小并通过系统调用来创建一个共享内存对象, 同时指定哪一个服务器获得相应的访问权限. 系统中维持着一个共享内存控制块数组, 微内核每次分配一个控制块给申请共享内存对象的相应进程.

3.4　陷入处理

中断、异常与系统调用是一个区分用户空间和内核空间的操作系统中三种不同的从用户空间进入内核空间的方式, 因而它们也被统称为"陷入". 陷入机制的实现跟底层硬件密切相关, 因而这部分是一个难点. 虽然它们各自的触发方式和作用截然不同, 但是微内核在处理这三类事件时的过程却是比较相似的.

本系统中当微内核收到中断信号时, 在完成底层处理后立即向相应服务器发送消息, 此后系统依然切回到用户空间中去. 例如, 当一个进程在用户空间运行时假设磁盘服务器先前发出的一个读扇区操作完成了, 此时磁盘驱动器将通过 8259A 芯片的第 15 号 IRQ 引脚向中断控制器发送一个中断信号, Bochs 将会模拟 i386 自动地完成以下三件事情:

(1) 切换到特权模式并从 TR 寄存器中取出内核堆栈 ss0: esp0 的值, 并将其加载到 ss: esp 寄存器中, 即将被中断进程的内核堆栈作为当前堆栈.

(2) 将 cs: ip: eflags: ss: esp 这五个寄存器中的值依次保存到当前系统堆栈.

(3) 根据中断向量号以及 IDTR 寄存器所指定的中断向量表取得对应的中断门数据结构, 从而获得底层中断处理函数的地址并且将其值与内核数据

段的段选择子加载到 CS:EIP 寄存器中去.

与中断机制的目的与触发方式不同,异常机制是处理器在硬件这个层次上提供的一种健壮性保证机制.其大体处理框架与中断响应的过程是一样的,只是有两个主要不同:

(1) 发生异常时,除了将几个关键寄存器压入系统堆栈以外,如果所发生的异常产生出错代码的话,处理器还会把这个出错代码也一起压入系统堆栈,因此异常处理程序必须对系统堆栈加以调整,以便在异常处理结束执行 iret 指令时,系统堆栈的栈顶存放的是返回地址.

(2) 与中断陷入不同,当发生异常的进程在用户空间中再次运行时,被执行的将是刚才引起异常的那条指令.

系统调用的实现被分成了两层:底层是具体系统调用的实现,由微内核负责在特权模式下完成;上层则是一个代理函数,静态地被连接到编译以后的用户进程代码中.

3.5 内核代理服务器

在微内核操作系统中,各部分之间的唯一通讯方式是标准化了的消息传递.因此,在内核中设计了一个内核代理服务器全权代理内核与外界的通讯.它与其他普通的进程没有任何区别,只是它没有用户空间堆栈并始终运行在内核空间中.主要作用是代表内核与核外各服务器及普通用户进程通讯.

4 典型服务器的实现

4.1 文件服务器

根据设计目标,系统设计了一个相对比较简单但同时又能解释清楚其工作原理的文件系统.为了体现文件系统中文件的具体内容与文件的管理信息分开存储的概念,系统设计了一个索引节点数据结构来描述文件的管理信息.该结构是实现逻辑文件到物理文件转换的核心数据结构之一,其定义如图 5(a)所示.

目录在系统中被设计成一个有着固定格式的特殊文件.它的内容是一个数组,每一项都具有"是否可用,索引节点号,文件名"三元组的格式,如图 5(b)所示.目录结构被实现为平板型的,即只有一层的层次文件系统,这样方便以后将文件系统扩展成多级目录.

本系统沿用了超级块的概念来描述文件系统在磁盘上的整体布局,尤其是结构和管理这两方面的信息.文件服务器在访问任何物理磁盘文件之前都必须先获得超级块,因而超级块也可以看成是访问文件系统的钥匙.另外,文件系统服务器还维护着一张记录每个进程有关文件操作信息的打开文件全局表,系统中每个普通用户进程所有打开文件的信息都被存储在这张表中.

```
struct inode {
    /* info on disk */
    int i_size;
    int i_sectors;
    int i_data[MAX_SECTOR];
    /* info only in the memory */
    int i_num;
    char i_dirt;
};
```

(a) 索引节点定义

```
struct dir_entry { /* is this entry usable */
    int active;
    /* inode for the file */
    int inode;
    char name[MAX_LENGTH];
};
struct directory {
    int entries;
    struct dir_entry table[MAX_ENTRIS];
};
```

(b) 文件目录定义

图 5 文件索引节点和目录的定义

最后,为了在虚拟机 Bochs 上配合使用本系统的文件系统服务器,还定制了一个磁盘映象文件,并且在虚拟机的配置文件 bochsrc.txt 中将其设置成 diskc 字段的值.制作文件系统映象的过程就是一个向格式化好了的空磁盘中逐个添加文件的过程.

4.2 设备管理服务器

由于磁盘是最为常见的随机存储外部设备,因此系统选择磁盘作为典型的输入输出设备,实现了磁盘服务器,并以之为代表实现了设备管理服务器.

磁盘服务器对磁盘的操作都是通过向设备控制器上的端口发送控制命令实现的,因此磁盘控制器可以被看作是磁盘服务器与磁盘硬件之间的接口.对磁盘的操作可以抽象为四类,即打开设备、关闭设备以及读设备和写设备,因此磁盘服务器启动以后将接收和处理这四类消息.

(下转第 17 页)

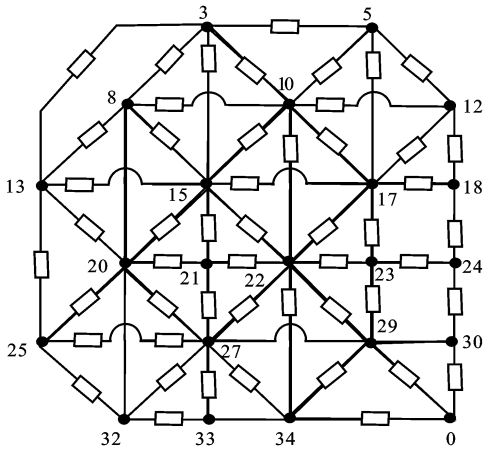

图 5　本征接地网

参考文献:

[1] 张丽萍, 袁建生, 李中新. 变电站接地网模拟计算[J]. 中国电机工程学报, 1999, 19(5): 76－79.

[2] 王东烨, 董刚. 大型变电所地网评估若干问题的探讨[J]. 高电压技术, 2001, 27(2): 64－65.

[3] 蔡崇积. 电力交流接地网损坏原因调研与分析[J]. 电力设备, 2005, 6(4): 20－22.

[4] 刘庆珍, 蔡金锭. 电力系统接地网故障的无伤检测方法[J]. 高电压技术, 2003, 29(6): 47－51.

[5] 刘健, 王建新, 王森. 测量误差对接地网故障诊断影响的分析[J]. 高电压技术, 2006, 32(4): 95－110.

[6] 栾超, 郭建胜. 基于分层序列法的搜索引擎系统设计[J]. 微电子学与计算机, 2007, 24(11): 80－82.

[7] 李朝鹏, 李肯立. 基于分层聚类的并行数据预处理算法[J]. 微电子学与计算机, 2007, 24(10): 187－189.

作者简介:

王树奇　男, (1974－), 博士研究生. 研究方向为模拟电路故障诊断.

(上接第 12 页)

5　结束语

文中设计并实现了一个基于 Bochs 虚拟机的微内核结构的操作系统. 该系统在保持简化性的前提下展现了一个真实操作系统的工作原理与机制, 并且保持了一定的可扩展性. 这个系统为可以很容易地被定制为面向嵌入应用的特定操作系统, 体现了微内核结构在嵌入式操作系统开发中的优越性.

参考文献:

[1] 付长冬, 孟庆余, 潘清. 基于微内核的操作系统综述[J]. 计算机工程与科学, 1997, 19(3): 72－75.

[2] 潘清, 张晓清. 操作系统微内核技术研究[J]. 软件学报, 1998, 9(8): 609－612.

[3] 张红光, 张健民, 李福才. 嵌入式系统虚拟运行平台的设计与研究[J]. 微电子学与计算机, 2005, 22(6): 101－103.

[4] Cisco System. Cisco Internetwork Design[M]. 美国: 美国 CISCO 公司, 1998.

[5] 刘瑞安. 基于嵌入式 Linux 的数字系统应用平台的设计与实现[J]. 微电子学与计算机, 2004, 21(6): 104－106.

[6] 徐恪, 吴建平, 喻中超, 等. 高性能路由器操作系统 HEROS 的设计与实现[J]. 小型微型计算机系统, 2002, 23(1): 18－24.

作者简介:

王红玲　女, (1975－), 博士研究生. 研究方向为信息处理、操作系统.

吕　强　男, (1965－), 教授. 研究方向为操作系统、智能计算.

诸亚铭　男, 硕士研究生.