基于 x86 平台的多任务操作系统的设计与实现

陈淑红，王　鸿，段　焰
（湖南工程学院 计算机与通信学院，湘潭 411104）

摘　要：操作系统是计算机用户和硬件的系统软件接口，为计算机程序提供通用服务. 简易操作系统的设计与实现有助于深入理解计算机相关技术，也可以为研究大型操作系统源代码提供参考. 基于 QEMU 设计并实现了一个基于 x86 平台的操作系统，运行在 32 位保护模式下. 实现了多任务管理、内存管理、设备管理、用户接口和应用程序接口等功能，有较好的可移植性.

关键词：操作系统；x86；多任务；32 位保护模式
中图分类号：TP316.7　　**文献标识码**：A　　**文章编号**：1671－119X(2015)04－0036－04

操作系统是计算机体系结构中最为基础和重要的软件系统，其将计算机提供的软、硬件资源进行管理和再组织，提供给用户和在其上运行的软件使用. 一个成型的操作系统往往非常复杂，代码量庞大，并且还涉及到很多硬件细节，虽然有很多开源的操作系统的源代码可以自由下载阅读，但是如果没有一些实际开发经验，往往无法理解其中的代码细节及其各功能模块之间的联系[1]. 开发一个简单的操作系统，可以积累操作系统的相关知识和开发经验，为阅读和研究大型操作系统源码打下基础.

本文从最基本的引导程序开始，介绍了一个具有多任务管理、内存管理、设备管理、中断处理、用户接口等操作系统核心功能的简易操作系统的设计与实现思路，为操作系统爱好者提供了设计参考.

1　系统总体结构

本文实现的操作系统麻雀虽小，五脏俱全. 该系统具有多任务管理、内存管理、设备管理、中断处理、用户接口等操作系统应具有的基本功能，总体结构如图 1 所示. 本系统的开发语言为 C 语言和 x86 汇编语言，开发平台主要为 Intel x86 平台的 PC 机和 QEMU 模拟处理器平台. 系统开发使用的文本编辑器为 CodeBlocks 内置的文本编辑器，使用的编译器为经过修改的 GCC 编译器.

2　系统内核功能模块的设计与实现

2.1　引导程序与模式转换程序

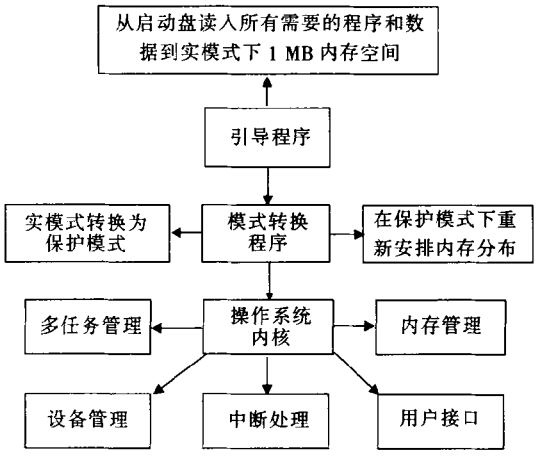

图 1　系统总体结构图

计算机启动后，固化在硬件上的 BIOS 程序开始检查启动盘的第一个扇区的内容，如果此扇区的最后两个字节的值为 55 AA，那么就可以确认这个扇区为引导扇区，因此在引导程序的最后加上语句 RESB 0x7dfe－$ 和语句 DB 0x55，0xaa，确保扇区最后两个字节的值为 55 AA. BIOS 会将该扇区的内容加载入内存的 0000:7c00 地址处并跳转到该地址执行扇区上的程序，计算机的控制器由 BIOS 转

收稿日期：2015－07－08
基金项目：国家自然科学基金青年基金项目(61502163)；湖南省教育厅高等学校科学研究项目(14C0286，13C0180).
作者简介：陈淑红(1975－)，女，博士，讲师，研究方向：社会网络分析，可信计算，算法优化.
通信作者：王　鸿(1992－)，男，本科，研究方向：计算机网络，操作系统.

移到了引导扇区程序[2,3]. 为此必须在引导程序中使用语句 ORG 0x7c00 指明程序装载地址. 引导扇区程序运行在实模式下，通过 BIOS 磁盘服务将启动盘中的包括模式转换程序在内的操作系统所有必要程序与数据读入实模式下 1 MB 内存中，并将计算机控制权交由模式转换程序[4].

2.2　内存管理

本文采用的内存分配方法为空闲分区列表法，优点为消耗内存小，分配迅速. 其基本思路是将空闲的内存区域的起始地址和大小记录在一个列表中，需要内存分配时，则遍历所有的表项找出符合条件的一块内存区域给它，其他操作类似. 内存变化后各内存块的边界问题尤为重要[6]. 为此定义了两个结构体 FREEINFO 和 MEMCTR. FREEINFO 结构体为内存可用信息条目结构体，包含两个数据项，其中 addr 存放可用内存块首地址，size 存放可用内存块大小. MEMCTR 结构体为内存管理结构体，包含 5 个数据项. 其中，frees 存放内存可用信息条目的数目，maxfrees 存放 frees 的最大值，lostsize 存放释放失败的内存大小的总和，losts 存放释放失败的次数，struct FREEINFO free[MEMMAN_FREES]存放内存可用信息条目数组.

内存管理实现的功能包括内存容量测试、内存分配和内存释放等功能，为了方便对于大内存的管理，减少磁盘碎片，本文额外设计了以 4KB 为单位的内存分配和内存释放功能，在操作系统中也主要使用这种方式进行内存管理.

2.3　中断处理

在 32 位保护模式下，CPU 使用中断描述符表寄存器(IDTR)存放中断描述符表的地址. 中断描述符表最多可以存放 256 个表项，因此最多可以注册 256 个中断处理程序. 门描述符存放着中断处理程序的地址. 由于中断处理程序在处理中断时必须要屏蔽其他可屏蔽中断信号，因此中断处理程序在设计时应尽快结束. 为了实现这一目的，本文采用的方法是利用缓冲区保存中断的必要信息然后迅速结束中断程序，其他程序根据缓冲区内的信息执行相应的处理操作.

中断处理模式如图 2 所示. 外部设备产生中断信号并发送给 CPU，CPU 通过 IDTR 找到中断描述符表(IDT)，并根据中断号找到对应的中断门描述符，通过中断门描述符中记录的中断处理程序的入口地址调用相应的中断处理程序. 中断处理程序通过端口读写指令从外设中读出需要用到的数据，并将数据写入缓冲区，然后尽快结束并通知 CPU 取消中断屏蔽. 内核主程序持续对缓冲区状态进行检测，当缓冲区有数据写入时，内核主程序读出这些数据并根据这些数据调用相应的功能程序，如绘图程序、控制台程序、任务切换程序等，从而执行相应的操作如重绘鼠标图层、在控制台显示相应的键盘输入、自动切换任务等，这样就完成了中断的处理过程.

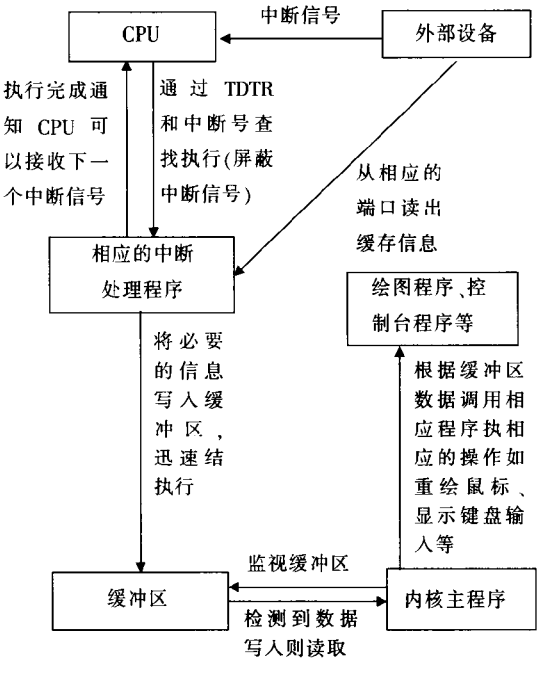

图 2　中断处理模式图

2.4　设备管理

本文实现的操作系统要管理的设备主要是鼠标、键盘、显示器和定时器等，其中定时器是一种特殊的芯片，其每隔一段设定好的时间就发送给 CPU 一个中断信号，这样 CPU 只要根据定时器发送的中断信号就可以计算时间，定时器的功能相当重要，在多任务的自动切换和其它涉及到时间计量的地方都需要使用定时器的功能.

设备管理功能的实现主要是通过为每个设备编写一个对应的驱动程序来检测他们发生的中断信号并执行相关的操作，要对这些设备的中断进行检测，那么必须要首先对 PIC(programmable interrupt controller)进行设定. PIC 和中断的关系非常密切. 由于 CPU 在当初的设计上只能单独处理一个中断信号，而电脑的外设很多，很可能同一时间要发出很多的中断，所以一般要增加几个辅助芯片帮助 CPU 去处理中断信号. PIC 就是这样的一种设备. 设置 PIC 的主要操作是为 PIC 引脚连接的设备发出的中断信号注册起始中断编号，正确设置 PIC 的连接方式、中断方式和触发方式等参数，PIC 设置完成后就可以根据中断号来编写相应的中断处理程序，当中断信号经过 PIC 传入 CPU 时，CPU 就可以正确的

调用已经编写好的中断处理程序，完成对于各种设备的管理.

2.5 缓冲区设计

如上文所述，为了能够让正在执行的中断处理程序可以和内核主程序以及其他程序进行信息传递，本文采用了缓冲区方式. 缓冲区采用先进先出模式，在前述设备中断处理程序中已经用到了它. 缓冲区的设计大大提高了中断程序执行的速度. 缓冲区不仅仅要存放数据，还要有记录读取和写入数据的位置指针等信息. 缓冲区结构体包含 7 个数据项，其中 buf 为缓冲区指针，记录着存放数据的内存地址，p 和 q 是数据写入和读出的位置，size 为缓冲区的大小，free 记录缓冲区有多少空闲，flags 记录缓冲区是否溢出，task 是与特定缓冲区相关联的任务，如果此缓冲区有数据写入，就唤醒对应的任务进行相关的处理过程. 通过定义缓冲区结构体可以很方便的实现缓冲区的读、写和状态检测等操作.

2.6 进程管理

本文设计和实现的是一个多任务操作系统，处理机管理尤其是进程的管理是实现多任务的手段. 另外，多任务系统还需要实现任务间的自动切换，为此涉及到定时器的相关功能.

由于处理机管理功能涉及到进程的调度算法以及任务运行时环境的改变，情况较为复杂，因此其所使用的数据结构也比较复杂. 对于一个任务来说，其需要保存的信息很多，为此在保护模式下设计任务状态段(TSS)来保存任务的各种信息，方便在任务切换等场合进行信息的保存和读取. 处理器中的任务寄存器 TR 指向当前获得处理机资源的进程对应的 TSS，在进行进程切换时，将当前的寄存器值等信息存入当前 TR 指向的 TSS 中，再将 TR 指向即将被执行的任务的 TSS，然后根据此 TSS 中的内容设置执行环境.

为了实现任务管理的功能，除了 TSS 中保存的信息外，还有很多其他的信息需要保存，例如任务的状态，优先级，时隙等. 由于任务切换需要在定时器中断处理中被调用，因此还需要有相关的缓冲区来传递信息. 为了保存这些信息，本文定义了一个任务结构体，包含 8 个数据项. 其中 select，flags，priolevel，exetime 分别为任务的段选择子、运行状态、优先级别和每次执行时间. fifo 为任务的缓冲区，tss 为任务的任务状态段，其结构体定义为固定格式. ldt 为任务的数据和代码段的段描述符. cons 为任务对应的控制台，在本文的系统中每个任务都以一个控制台窗口标识.

本文的任务调度算法为多优先级队列调度算法，设置了 10 个优先级队列，每个任务都属于其中的一个优先级队列，属于特定优先队列的所有任务按照一定的规则进行调度.

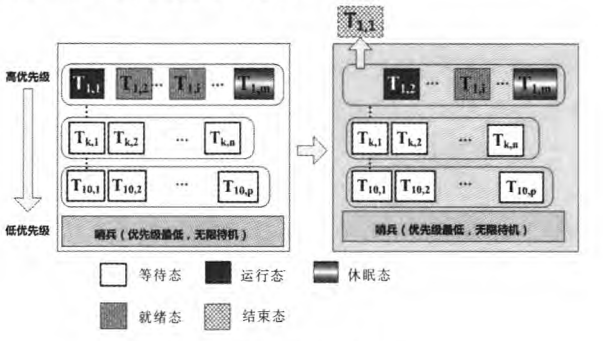

图 3 任务调度模型图

任务的调度过程如下：

(1)按优先级从高到低依次检查优先队列，找到第一个非空队列.

(2)根据公式 index2=(index1+1)%sum 从该队列当前任务的位置找到下一个任务的位置. 其中 index1 为当前任务在任务数组中的位置，index2 为下一个要被切换的任务的位置，sum 为任务队列中的任务总数.

(3)为该任务设置定时器和运行时间(当运行时间结束后该定时器产生中断信号，中断处理程序调用任务调度程序).

(4)利用远跳转使 CPU 自动切换到下一个任务.

如图 3 所示，优先级按从上往下的顺序由高至低分布. 上一优先级队列中的任务没有全部执行完之前，下一优先级的任务将没有机会得到执行. 在每个优先级队列中，任务调度按顺序进行，但当有任务因为等待输入数据等原因不能继续执行时，系统会将其设置为休眠，此任务在唤醒之前不会参与任务调度，当该任务等待的数据到来时系统会将其唤醒，该任务又可以和其它任务一起参与任务调度. 为了系统在没有其它任务时不会频繁地执行调度程序，浪费资源，在最低优先队列中加入了一个“哨兵”任务，该任务用来进行待机操作，将其运行时间设置为最大值，这样如果多任务队列中没有其它任务执行，系统会自动进行待机等待.

2.7 用户接口

用户接口包括控制台程序和图形用户界面，控制台程序类似于 DOS 操作系统的模式，通过用户输入指定的命令执行相应的操作，此外，本文实现的操作系统的每一个任务都会关联一个控制台，方便用户输入指令. 图形用户界面的实现本质上就是对于显存空间的写入操作，在设计上为了避免各图形组件之间互相干扰，提高图像刷新效果，为每个可以显

示和拖动的图形组件均设置了一个图层,如果只是上层图层发生变化,则无需刷新比其层级低的下层图层.另外又设计了一个图形内存映射缓冲区,其相当于是显存空间的映射,如果图形界面有变化需要刷新,先在图形内存映射缓冲区内绘制完成后再写入显存中去,这些设计极大的提高了图形绘制程序的效率和灵活性.

2.8　应用程序接口

API 设计是利用中断调用系统代码以实现 API 功能.由于中断号是有限的,如果一个中断号对应于一个系统调用,则可能因为系统调用数量过多造成中断号不足.因此,本文采用的方式为只用一个中断号注册一个 API 调用中断程序,在一个寄存器中放入功能号,这样根据功能号的不同就可以达到通过一个中断 8 号实现大量的 API 调用.由于用户程序运行在用户态,操作系统内核功能函数运行在核心态,其使用的堆栈和内存区域不同,因此在 API 调用过程中使用寄存器和公共内存区域传递功能号和函数参数.API 调用过程如图 4 所示.

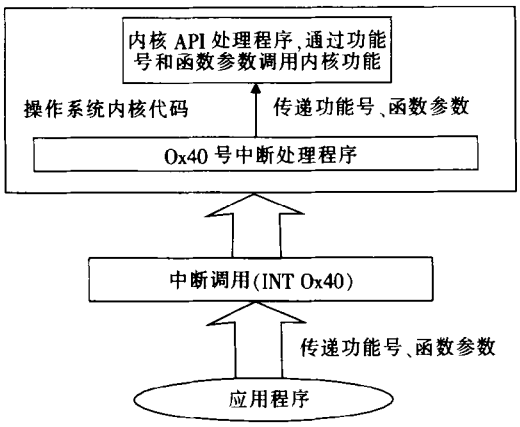

图 4　API 调用过程图

3　结束语

本文采用 C 语言和 x86 汇编语言开发了一个基于 Intel x86 平台的操作系统,具有较好的可移植性.实现了操作系统所应有的基本功能,用户界面较友好.由于时间有限,本文设计的操作系统较为简单,在后期研究中将考虑实现磁盘读写功能以减少内存消耗,并可以考虑实现更多的 API 函数,以编写出更多更好的应用程序.

参　考　文　献

[1] Cao Bo, Yang Shan, Liang Xinjian. Design and Implementation of Operating System Security Monitoring of Mobile Intelligence Terminal[J]. Microelectronics & Computer, 2014,31(10):146－152.

[2] 苑　勋,王　琰,黄利萍.一个微型操作系统的设计和实现[J].沈阳工业学院学报,2001,20(2):30－34.

[3] Andreas Hanssona, Marcus Ekerhultb, Anca Molnosc. Design and Implementation of An Operating System for Composable Processor Sharing[J]. Microprocessors and Microsystems, 2011,35(2): 246－260.

[4] 吴兆芝.X86 平台操作系统引导技术研究与实现[J].南京晓庄学院学报,2011(6):94－97.

[5] 杨学俊,冯荣荣,杨　燕.基于 X86 平台的嵌入式 Linux 操作系统[J].软件开发与设计,2011(14):35－40.

Design and Implementation of Multi-task Operating System Based on Intel X86 Platform

CHEN Shu-hong,　WANG Hong,　DUAN Yan

(College of Computer and Communication, Hunan Institute of Engineering, Xiangtan 411104, China)

Abstract: An operating system is the system software which acts as an interface between user and computer hardware and provides common services for computer programs. Implementation of the operating system helps us to understand computer-related technologies. It can also provide a reference for studying source code of large-scale operating system. This paper designs and implements an operating system based on Intel X86 using QEMU which runs on 32-bitprotected mode. It can realize multi-tasking management, memory management, device management, user interfaces and application programming interfaces, etc. All these functions are portable.

Keywords: operating system; X86; multi-task; 32-bit protected mode