小型单内核嵌入式操作系统的设计与实现

曲 波

(南京晓庄学院 数学与信息技术学院, 江苏 南京 210017)

摘 要: 文章阐述了小型单内核嵌入式操作系统的设计与实现. 该系统以流行的 PC机做研发平台, 以 VMWare虚拟机做实验环境, 以 Linux操作系统及开放软件做开发工具, 整个系统的源码约为 4200行. 该系统可实现多进程优先级调度、信号处理、段页式存储管理、写时复制、虚拟内存等功能. 结构简洁、易于移植, 适用于嵌入式系统应用开发及操作系统教学实验.

关键词: 嵌入式系统; 单内核操作系统; 进程调度; 写时复制; 虚拟内存

中图分类号: TP316 文献标识码: A 文章编号: 1009-7902(2009)06-0079-04

0 引言

随着计算机技术的不断发展, 嵌入式系统的应用也越来越普及. 如何有效设计实现嵌入式系统, 是当前业界研究的一个重要课题. 另一方面, 如何加强计算机专业操作系统课程的实践性教学, 也是多年来操作系统课程教学研究的重要课题. 本文阐述的小型单内核嵌入式操作系统设计与实现, 就是在这方面的一个深入研究与探讨.

本文实现的小型单内核嵌入式操作系统 (以下简称 EMK)以流行的 PC机做研发平台, 以 VMWare虚拟机做实验环境, 以 Linux操作系统及开放软件做开发工具, 整个系统的源码约为 4200行, 其中汇编代码约 1100行, C代码 3100行. 该系统可实现多进程优先级调度、信号处理、段页式存储管理、通过写时复制功能实现简单的虚拟内存.

EMK由引导程序和系统内核两大部分组成. 为了便于开发、便于移植、便于理解, 整个程序结构简洁, 使用 GCC编译器, 采用标准的 GCC汇编语言 (AT&T语法)及 C语言语法, 程序设计风格类似于 Linux, 但不使用嵌入式汇编 (embedded-assembly)、内联编译 (inline)等特殊方法. 由于该系统规模较小、功能实用, 不仅适用于实际的嵌入式应用开发, 同时也适用于操作系统及嵌入式操作系统的教学与实验.

EMK以 PC机作为研发平台. PC机可比一般嵌入式硬件环境提供更多的扩展性和硬件部件, 如监示器、键盘等, 可使操作系统的设计涉及更多的硬件设备. 然而, 如果将所设计的操作系统直接用 PC机执行, 显然不便于系统的开发与调试. 因此, 笔者采用 VMWare作为实验环境, 由 VMWare虚拟机运行 EMK.

1 引导程序

引导程序负责将内核从外存介质 (如软盘、U盘等)引导到内存, 启动内核运行.[1-2]

为简化系统、便于移植, EMK的引导程序采用直接读写软盘扇区的形式实现. 为此, 笔者专门设计了一个系统生成工具, 将引导扇区、装载程序和内核镜像在软盘 0面 0道 1扇区依次写入. 引导扇区与装载程序由一个汇编程序实现, 前半部分为引导部分, 写入 0面 0道 1扇区, 以引导扇区标志 0xAA55结束; 后半部分为装载部分, 负责将内核从软盘读入内存后, 将 CPU交给内核, 启动并运行操作系统. 由于引导扇区容量有限 (不超过 510字节), 不可能完成复杂的启动工作, 所以只负责将装载程序及系统内核一次性读入内存, 然后由装载程序完成后续工作.

主机启动后, BIOS将软盘 0面 0道 1扇区的引

收稿日期: 2009-09-21

作者简介: 曲波 (1953—), 男, 辽宁大连人, 南京晓庄学院数学与信息技术学院教授. 研究方向: 操作系统、嵌入式系统、软件工程.

导扇区装入内存 0x7C00处执行,引导扇区程序首先将自身移动到 0x90000处,然后跳转到 0x90000处继续执行:

首先,从软盘 0面 0道 2扇区开始,依次读入装载程序和内核镜像.然后,跳转动 0x90200处,执行装载程序,完成启动内核前的准备工作,包括检测主机配置、移动内核、开启 A20地址线、设置全局描述符、进入保护模式、跳转到内核启动地址等.

PC机开启 A20地址线有几种方法, EMK采用 PC机最初使用的方法,通过设置键盘控制器的端口值实现对 A20地址线的控制.开启 A20地址线后, EMK通过"加载机器状态字"的方法,设置进入 32位保护模式运行.最后,通过一条长跳转指令,跳转到内核首地址,启动内核运行.

2 内核

内核完成操作系统的各项功能,由内核主框架、中断处理(硬件中断、软件陷阱、系统调用)、TTY终端、进程调度、信号处理、内存管理、库函数等部分组成.[1,2,3]

2.1 内核主框架

内核主框架完成保护模式下启动内核的初始化工作,包括:初始化中断描述符表(设置默认中断)、初始化全局描述符表、检测 A20地址线、初始化页表、创建初始用户进程等.

由于系统进入了保护模式,所以必须为系统重新设置中断,并设置相应的中断描述符. EMK使用了一个非常简单的中断程序作为默认中断:

```
dummy_int:
    incb 0xb8000+160 # put something on the screen
    movb $2, 0xb8000+161 # so that we know something happened
    iret
```

页表初始化程序根据装载程序测得的系统内存容量,初始化页表目录及页表项. EMK使用段页式内存管理模式,最多可使用 64M物理内存.

为保证第一个用户进程堆栈的"纯洁性", EMK使用汇编代码直接调用"fork"系统调用,生成子进程,避免由于 C语言程序调用所引起的堆栈操作.

2.2 中断处理

EMK的中断处理由硬件中断(Hardware Interrupts)、软件陷阱(Software Traps)和系统调用(System Calls)三大部分组成.由于本质上都是中断,所以采用统一的中断接口.

2.2.1 硬件中断

EMK的硬件中断包括时钟中断及键盘中断,全部由 C程序实现.

时钟中断由初始化程序及中断处理两部分组成.初始化程序完成对 8253定时器的初始化、设置时钟中断门、修改中断控制器屏蔽码(允许时钟中断).中断处理包括修改系统时钟计数(ticks++)、修改当前进程时间片计数、调用进程调度程序(schedule)等.

键盘中断实现对键盘信息的接收、向 TTY终端传送键盘信息等.考虑到系统的简洁性, EMK没有实现标准终端的复杂功能,不进行 ESC代码系列转换,因此大大简化了键盘中断及 TTY程序. EMK的键盘中断处理程序根据当前键盘状态产生相应的双字节代码,低字节为按键的 ASCII码,高字节为状态码(包括 CapsLock, Shift, Ctrl, Alt, NumLock等).

2.2.2 软件陷阱

EMK的软件陷阱设计得非常简洁,除缺页中断(14号)外,只是简单地将当前 CPU寄存器状态显示出来.为得到当前 CPU寄存器状态,中断接口在调用软件陷阱处理程序之前,先将全部 CPU寄存器入栈,然后将 ESP寄存器入栈,作为软件陷阱处理程序的一个参数.软件陷阱处理程序可以通过此参数依次访问堆栈上保存的全部寄存器.

由于 EMK使用"写时复制(copy_on_write)"技术实现简单的虚拟内存,所以在内核运行过程中势必会产生"缺页中断".所以,对缺页中断的处理不再是简单地显示中断信息,而需要对缺页中断做相应的处理. EMK的"缺页中断"处理由 C程序实现.具体实现参见后面"内存管理"部分的说明.

2.2.3 系统调用

EMK的系统调用包括 sys_exit, sys_fork, sys_waitpid, sys_getpid, sys_alarm, sys_pause, sys_nice, sys_kill, sys_signal, sys_getppid等调用函数,全部由 C程序实现.

由于系统调用本质上是中断调用,所以系统调用处理本质上是中断处理. EMK的系统调用处理器程序如下:

```
void syscall_handler(struct intr_state * state)
{
    state->eax=
    (*syscall_table[state->eax])(state);
    if(curtask->state!=TASK_RUNNING||
curtask->count==0)
    schedule();
    if(curtask==task[0]||state->cs!=0x0f
||state->ss!=0x17)
```

```
        return;
    do_signal(sigmap(), state);
}
```

其中,指针 state的值实质上是堆栈寄存器 ESP的值,所以通过 state变量,就可以访问堆栈上保存的各寄存器. EAX为系统调用功能码, sycall_table为系统调用函数表,以 EAX为索引值,即可调用该功能码所对应的系统调用处理函数.

系统调用处理函数完成系统调用功能处理后,返回值存入 EAX,如果当前进程不是就绪态进程,或当前进程时钟计数器归零,则调用调度程序重新调度.然后判断当前进程是否任务 0,如果是任务 0则返回.否则,判断是否在用户数据段或用户堆栈段,如果不是,则返回.

如果同时处于用户代码段及用户堆栈段,则还需要进行信号方面的处理,由 do_signal函数实现.

2.3 TTY终端

EMK的 TTY终端由键盘中断处理、控制台(屏幕)显示处理、TTY处理三部分组成.

控制台显示处理实现屏幕的上卷(Screen RollUp)、下卷(Screen RollDown)、普通可显示字符的显示、回车、换行、制表符等. TTY处理程序设置三个缓冲队列,一个用来暂存键盘信息、一个用来作为输入缓冲区、一个用来作为输出缓冲区.

由于不需实现标准终端功能,不实现 ESC代码序列,极大简化了 TTY程序. EMK的 TTY部分(键盘、显示、TTY三个程序)只有不足 600行 C代码.

2.4 进程调度

EMK的进程调度根据进程的时间片和优先权调度机制进行调度.它首先循环检查进程数组中的所有进程,根据每个就绪态进程的剩余时间值,选取该值最大的进程.若所有就绪态进程的时间片都用完,则根据各进程的优先权值重置每个进程的时间片值,然后再次循环检查所有进程的时间片值,选取该值最大的进程.

PC机的进程切换有三种方法:通过任务门、通过任务状态段、或在进入或退出中断/异常处理时实现. EMK采用通过任务状态段 TSS进行进程切换的方法.当 JMP或 CALL指令中包含的选择符指向一个可用任务状态段描述符时,就可以发生从当前进程到 TSS指向进程的转移.转移的目标地址由进程的 TSS中的 CS和 EIP决定,而 JMP或 CALL指令中的偏移则被丢弃.

EMK实现进程切换的汇编子程序如下:

```
switch_task_to:
    ljmp *(%esp,1)
    ret
```

在进程调度 C程序中将前述选中进程的 TSS描述符地址作为参数调用汇编子程序 switch_task_to(),即可通过 JMP指令实现进程的切换.在 Linux操作系统中,这部分是由嵌入式汇编实现的.

2.5 信号处理

EMK可实现简单的信号处理功能,由三个函数构成: sigmap()、do_signal()和 sys_signal().

do_signal()函数是内核系统调用中断处理程序中信号的预处理程序,主要作用是将信号的处理句柄插入到用户程序堆栈中,从而在当前系统调用结束返回后就会立即执行信号句柄程序,然后再继续执行用户程序. sigmap()函数由 do_signal()函数调用,检测当前任务的信号量.

sys_signal()函数为指定的信号安装新的信号句柄(信号处理程序).信号句柄可以是用户指定的函数,也可以是 SIG_DFL(默认句柄)或 SIG_IGN(忽略).

2.6 内存管理

EMK的内存管理由生成子进程(fork)、中止子进程(exit)、内存段页式管理等功能组成. EMK采用段页式内存管理、写时复制机制,实现简单的虚拟内存功能.

2.6.1 段页式内存管理

EMK采用段页式内存管理机制,最多支持 64M物理内存,最多可实现 64个进程,每个进程在线性地址中都是从 nr*64M的地址位置开始(n是任务号),占用地址空间的范围是 64M,因此,所有进程空间的总和为 64M*64=4G.

为了使用分页机制,一个 32位的线性地址被分成三个部分,分别用来指定一个页目录项、一个页表项和对应物理内存页上的偏移地址,从而间接寻址到线性地址指定的物理内存位置.在 EMK中,线性地址的位 31—22共 10位用来确定页目录中的目录项、位 21—12共 10位用来寻址目录项指定的页表中的页表项,最后 12位用作页表项指定的一页物理内存中的偏移地址.

一个系统中可以同时存在多个页目录表,而在某个时刻只有一个页目录表可用.当前的页目录表是用 CPU的寄存器 CR3来确定的,它保存当前页目录表的物理内存地址. EMK只使用一个页目录表.

2.6.2 写时复制

为节约内存,实现简单的虚拟内存功能, EMK采用“写时复制”机制.在调用 fork()函数生成新进

程时，将新进程设置为与父进程以只读方式共享同一内存区.只有当其中一个进程需要进行写操作时，由于共享内存区域为只读方式，从而产生缺页中断；然后，由缺页中断处理程序为其另外分配内存页面，并将共享区域内容复制到新页面，使父进程与子进程各自拥有一块内容相同的物理页面；最后，将父、子进程相对应的页面都设置为可以写访问.

2.7 库函数

为便于移植，同时也便于教学与实验，在 EMK 的程序设计中，尽量使用 C语言编程.原则上，只要可用 C语言实现的，就不用汇编程序.整个 EMK内核只包括三个汇编程序：主框架程序（main.s）、中断接口程序（intr.s）和汇编语言库函数程序（liba.s）.

汇编语言库函数主要有：端口输入输出函数、允许/禁止中断函数、直接写屏函数、进程切换函数、读/写任务寄存器函数、读/写页表寄存器函数、读/写 FS数据段函数等.

C语言库函数主要有：系统调用函数、串处理函数等.

2.8 宏

为使程序尽可能简洁，并提高可读性，对于一些保护方式下的描述符操作使用 C语言的“宏”而不是函数来实现.例如：

```
#define set_gate_desc(addr,type,dpl,off) {\
*((unsigned short *)(addr)+0)=(unsigned)(off)&0xffff;\
*((unsigned short *)(addr)+1)=0x08;\
*((unsigned short *)(addr)+2)=((type)<<8)|((dpl)<<13);\
*((unsigned short *)(addr)+3)=(unsigned)(off)>>16;}
```

该宏定义实现对保护方式下门描述符的设置，比用函数方式简洁.在此基础上，即可定义本系统特定的设置中断门和陷阱门的宏：

```
#define set_intr_gate(n,off) set_gate_desc(&idt[n],DA_386IGate,PRIV_KERN,off)
#define set_trap_gate(n,off) set_gate_desc(&idt[n],DA_386TGate,PRIV_KERN,off)
#define set_user_gate(n,off) set_gate_desc(&idt[n],DA_386IGate,PRIV_USER,off)
```

3 结束语

操作系统课程是计算机专业重要的核心课程，如何加强操作系统课程的实践性教学，是多年来操作系统教学研究的一个重要课题.另一方面，随着嵌入式技术的发展，社会对于嵌入式操作系统技术的需求以及对嵌入式人才的需求都越来越多.鉴于国内外操作系统课程教学的现状，笔者设计并实现了这个小型单内核嵌入式操作系统，采用开放软件工具和编程环境，结构简洁，只有 4200行源码，便于学生学习与理解；易于移植，学生还可以在本系统的基础上进一步扩充改进；非常适合于嵌入式操作系统的教学与实验，很有实用价值.深入进行这方面的研究，也无疑具有重要意义.

参考文献：

[1]卢军. Linux0.01内核分析与操作系统设计[M].北京：清华大学出版社，2005.

[2]赵炯. Linux内核完全注释[M].北京：机械工业出版社，2007.

（责任编辑：王海军）

Design and Implementation of a Small Embedded Operating System of Monolithic Kernel

QU Bo

（School of Information & Technology, Nanjing Xiaozhuang University, Nanjing 210017, Jiangsu）

Abstract: The paper describes the design and implementation of a small embedded operating system of monolithic kernel. The system consists of about 4200 lines of source code, developed with PC as the developing platform, VMWare as the testing environment, Linux and the open source software as the developing tools. The system can perform such functions as the multi-process priority scheduling, signal processing, segment-paging memory management, copy-on-write virtual memory. With the simple structure and easy portability, the system is suitable for the application in the development of embedded system, and in teaching and testing on operating system.

Key words: embedded system; operating system of monolithic kernel; process scheduling; copy-on-write; virtual memory