基于.COM文件的一种动态加载机制的研究

王 铮,杨志祥
(重庆大学 计算机学院,重庆 400044)

摘 要:提出并设计了应用于嵌入式环境下的一种基于.COM文件格式的一种动态加载机制,其基本思想是系统只需要一个该机制,可以动态加载应用模块配置各种不同应用需要嵌入式系统。其优点是灵活性好,如可以动态地修改或者升级等等,这在嵌入式环境中应用是非常重要的。文中就.COM文件模块的动态加载进行了阐述,并提出设计了一套实验性的原型方案。

关键词:动态加载; 嵌入式环境; 解析;目标模块
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2007)04-0045-04

Research of a Dynamic Loading Mechanism Based on .COM Files

WANG Zheng, YANG Zhi-xiang
(College of Computer Science and Technology, Chongqing University, Chongqing 400044, China)

Abstract: A dynamic loading mechanism based on .COM files is presented and designed for application of embedded environment. So system only needs a dynamic loading mechanism, which can load dynamically the module and dispose the embedded systems which suit different uses. The advantage of this technique is the flexibility, which can be used in updating or modifying system dynamically, and it is very suitable in embedded environment. The dynamic loading mechanism based on .COM files is described in detail in this thesis, and a simple prototype was provided.

Key words: dynamic loading; embedded environment; resolve; object module

0 引 言

软件技术的不断发展,使得开发出来的系统越来越大和复杂。但是,很多软件都存在下列问题:有不可忽视的错误;或者不能完全满足用户的需求;或者使用过程中需要不断地修改和升级等等。在通常情况下,所有的这些修改和升级会导致停机。当用户越来越依赖一个系统的时候,这种中断对他们来说是不能忍受的,如金融处理系统、电信交换机系统、交通控制系统,以及一些关键的军事应用上的卫星系统等等。针对这些应用,能够动态地升级和修改是必须的。随着嵌入式应用的广泛普及到生活的各个方面,动态加载升级的要求在嵌入式环境下也显得越发重要。比如在恶劣且遥远的环境下的嵌入式应用,有什么修改和升级,如果要工作人员到现场才能完成的话,显得十分不经济和现实。但是,若这些嵌入式应用拥有动态修改和加载升级的功能,则这些繁琐的动作可以通过网络或者无线接口轻松完成。

收稿日期:2006-06-21

作者简介:王 铮(1953-),男,重庆人,副教授,研究方向为软件自动化、嵌入式系统。

1 动态加载机制研究背景

嵌入式系统正在被广泛应用,因此对嵌入式系统的研究也越来越成为人们关注的焦点。因为嵌入式环境的本身的特殊性,如其存储器极其有限,没有明显的输入输出,其工作地点可能是处在遥远甚至恶劣的环境之中,更特殊的如航天器、卫星上的嵌入式应用等。现有的嵌入式系统,在灵活性方面有很多不足,如系统运行起来之后是无法改变的,并且对某些具体应用,根本就不需要的功能,也要一次装载,这样既不经济,也浪费了本来就很宝贵的系统资源。

文中就是针对嵌入式软件应用的需求,提出并设计的一种基于.COM文件动态加载机制,整个系统只是一个动态加载机制,可以把应用模块从零开始动态地加载来配置各种不同应用需要的嵌入式系统。这样的系统与现今的嵌入式系统,在修改升级和实用方面,灵活性、方便性和经济性均有极大的提高。

针对嵌入式系统的特点,文中的研究着眼于提出一套实用性及实验性的原型方案,为了避免开发新的编译器或解释器的负担,选择已有的成熟语言作为工具。毕竟,新的语言不仅对用户来说是个负担,而且,

开发新的语言需要长时间的开发流程作为代价,不如选择成熟的语言。其次,选择微软公司的 MS-DOS 的.COM 文件格式作为动态加载模块的格式,这是因为,.COM 文件里面只包含可执行代码,是程序的一个绝对映象,MS-DOS 仅仅是将文件的内容载入一块从 0x100 开始的可用的内存空间。加载模块的格式选择.COM 文件,这样就不需要在加载的过程中进行重定位了。而一般的文件进行重定位的话,都是比较复杂麻烦的。尽管.COM 文件大小一般要求为 64k,但是嵌入式环境下的 RAM 是非常有限的,而且,因为大多数关于动态加载的文献和技术都是有专利的,.COM 文件小一点、简单点也便于操作和应用。

2 动态加载机制的原理分析

文中研究讨论的动态加载机制有两个特点:首先是动态地加载,也就是这个运行的模块在需要的时候才被映射入运行模块的虚拟内存空间中;其次是动态地解析,也就是当要调用的函数模块被调用的时候,才会去把这个函数模块在虚拟内存空间的起始地址解析出来,再写到专门在调用模块中的存储地址内。

文中所研究的动态加载机制模块格式采用的是.COM 文件,它只含有可执行的二进制代码,不存在浮动地址信息,当对其进行加载时,不需要进行重定位工作,这就为设计和实现该动态加载机制省了不少工作。这样的话,由于模块格式本身是可执行代码,则可以把该机制看作由加载和分解二个基本的操作构成。当系统检测有新的需求或有新的模块需要加载时,动态加载机制先在系统已加载模块链中搜索。如果该模块已经在系统里,就不需要下载该模块;如果不在系统里,就需要从远程下载需要的目标模块到本地。下载到本地的模块,加载器就给其分配内存,然后将其注册到模块链中。目标模块不再被系统需要的时候为了节省存储空间就要把模块从系统中删除掉,这就是分解操作,也就是模块的卸载,被说明的模块从内存中移出,分配给它的存储空间被收回,此外,必须取消外部引用的解释[1,2]。

3 动态加载过程模型

动态加载机制是个实施起来非常困难的问题,在动态加载的过程中需要考虑很多的外部和内部因素,而且系统平台的不同动态加载的具体实现也会有很大的不同。针对文中研究的动态加载机制原理分析,提出了一个动态加载机制的过程模型(见图 1)[3]。该模型以描述一个目标模块的生命周期为主线,在主线上侧重表现了动态加载机制对其作用的阶段,体现了动态加载机制中加载、处理、执行和卸载模块的过程,也反映了动态加载机制本身的功能结构。显然,在这个模型中可以清晰地看到目标模块动态加载的整个过程以及每个阶段需要解决的问题。当然,由于文中采用的目标加载模块是.COM 文件格式,其拥有独特的性质(在 4.1 节详细介绍),使得图中虚线部分的重定位以及重定位表在实际实施过程中就不需要了。

4 动态加载机制的设计

4.1 .COM 文件及其模块名字空间管理

微软公司.COM 文件被称为“空目标文件”,文件里面只存在二进制代码,没有像.EXE 文件所具有的包括有关文件信息的标题区(header)。选择.COM 文件格式的模块的话,因为其代码中没有浮动的地址信息,当把模块下载到系统时,就不需要再对其进行重定位操作了,只需要装载器分配内存空间即可。但是,由于.COM 文件里没有标题信息,就出现了一个问题,那就是.COM 文件模块的相互调用的问题。加载到系统里的模块,必须在系统模块的管理机制中进行注册,

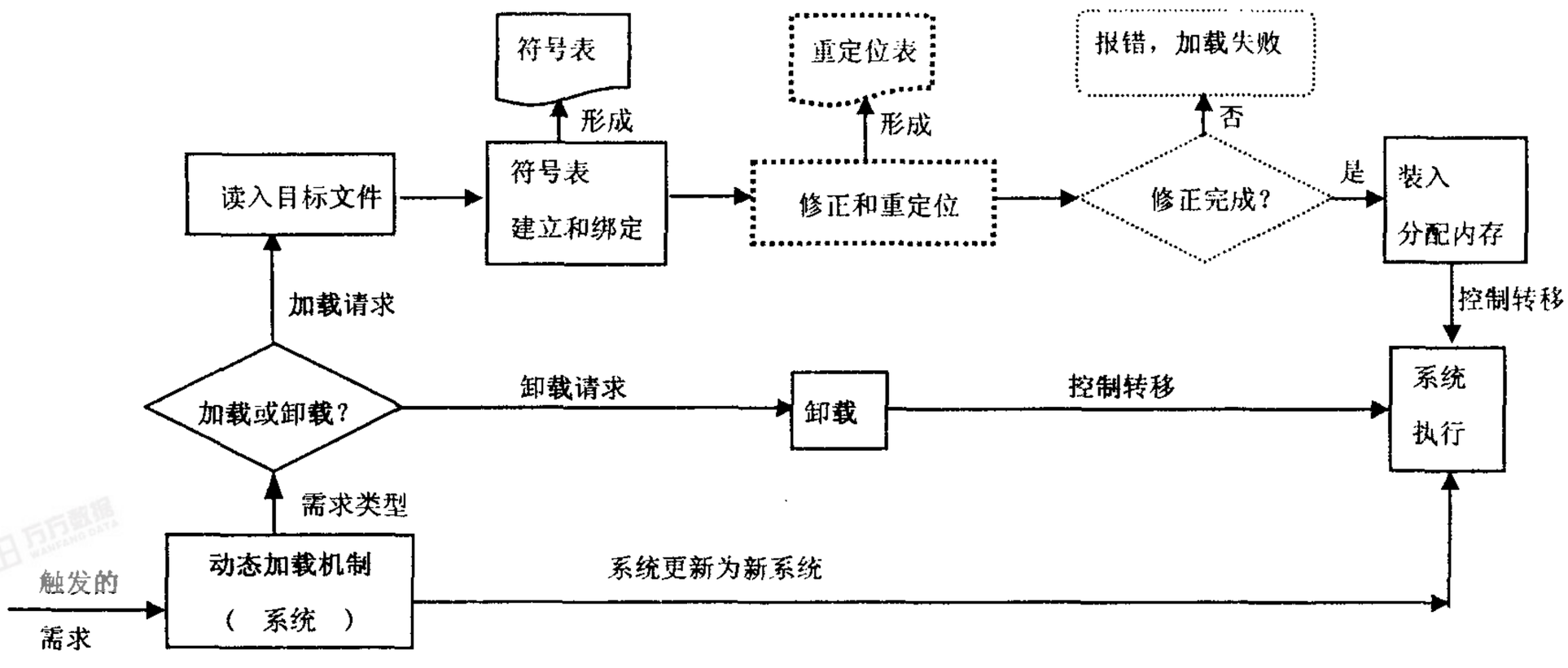

图 1 动态加载过程图

这里,采用链表作为数据结构,把各个模块的入口地址记录在模块链表中。由于.COM 文件模块具有其独特性,采用预编译的.COM 源文件留空,然后与一个名为 Main 的 C 函数 OBJ 文件进行连接,最终生成.COM 模块文件。C 函数部分代码如下:

```
void main(void)
{
asm string db 'dm_**.com','$';/* 汇编语句内嵌到 C 函数中,"dm**.com"字符串作为注册到模块管理器中的入口地址.*/
……
}
```

应用程序.COM 文件模块的.ASM 源文件结构及其调用该 C 函数框架如下:

```
extrn_main: near ;声明外部函数调用
code segment para 'code'
    assume
cs:code,ds:code,ss:code,es:code
org 100h
begin:
    call_main
    ……
    ;应用程序主体
    main endp
code ends
end begin
```

简要解释一下以上函数功能:main 函数里面的内嵌汇编语句是为了能够让动态加载机制里的模块管理器作为模块入口地址,当作名字链接到模块链表中,也即通过该字符串可以得到模块的入口地址,相当于该模块的名字,保存在名字表中。在这里作一个约定,每个加载模块源码中只包含一个汇编程序的子过程,当然了,子过程规模可大可小,这样做是为了不给动态加载机制里的模块管理器设计带来复杂的数据结构来查找各个模块名字表。

由于对每个模块建立一个名字表,并向模块管理器注册这个模块的定位地址表,在模块的初始化代码中,模块通过系统调用向加载器注册,这时需提供模块名和模块的入口地址,模块管理器中管理着一个这样的名字表格,并同时维护一张本模块提供的函数接口地址表(见图 2),由于模块文件约定了只有一个子过程,也即一个入口,那么该模块注册的名字即是其入口地址。具体方法是把名字表用链表链接起来,然后保存在一个文件之中,以后使用时再查询它们。名字空间管理工作是:

①模块名字链表中连接各个模块名字对应的符号串,建立名字表;

②加载时,查询表格链表,找到匹配的名字串字符表及其对应其他信息,并完成冲突查询;

③卸载时,卸掉名字表中相应的名字及相关数据结构。

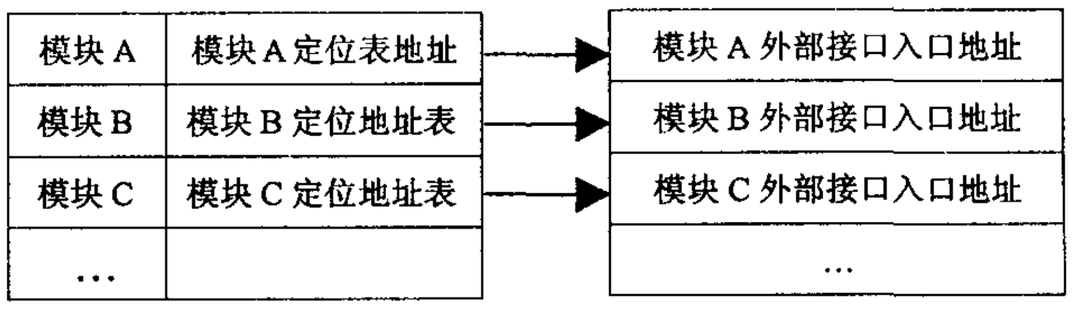

图 2　模块名字表及其外部接口地址表

4.2　动态加载机制的设计

动态加载机制的设计引入了以下几个概念:可动态加载的模块(DM, Dynamic Module),伪内核(PseudoKernel), DMServer, 模块管理器(DMM, Dynamic Module Manager),调度器(Scheduler),动态加载机制。

①动态加载的模块(DM)。主要是指加载器加载的一个单位,并且这里模块的概念主要就是.COM 文件格式的动态可加载模块的封装, 里面涉及到其模块的一些数据变结构变量,如模块的大小、模块装入的地址、模块的引用记数、模块间的依赖关系等。这样,模块格式保证了动态加载机制载能够识别并把模块登记在该内核任务管理链中,同时在运行的时候,DM 能够安全正确地被加载或从系统中安全地卸载。

②伪内核。可以响应外部的加载模块的事件中断,其主要功能是能够完成控制功能的转移。

③DMServer 的主要任务是完成加载和卸载 DM。

④模块管理器(DMM)。指对加载进系统的模块进行名字表管理和定位。

⑤调度器(Scheduler)。它是伪内核中重要的一个模块部分,它管理各个目标程序模进行实时调度。

⑥动态加载机制。最终所有功能模块组合起来相当于一个系统,只是这系统不同于其他系统,它只是一个动态加载机制,但是它可以从零开始动态地加载模块来配置各种满足不同应用需要的嵌入式系统。

系统中要实现动态加载,首先需要一种模块定位机制,使得调用者能够在系统中动态定位需要的模块;其次是要能让模块与目标程序动态地关联在一起,协调工作。为了解决这些问题,需要一系列相关的设计:模块的存储管理;简化目标机端模块地址空间定位的工作等等。当模块加载到系统中,首先对各个模块进行初始化,同时有一个模块的注册函数(该函数主要工作是向系统注册模块的名字和接口函数地址表地址)。当模块被加载时,初始化函数会被系统统一调用,向系统注册模块信息,此后模块交由加载器统一管理。基于这样的设计,系统可以比较顺利地实现动态加载。

整个动态加载系统见图 3。

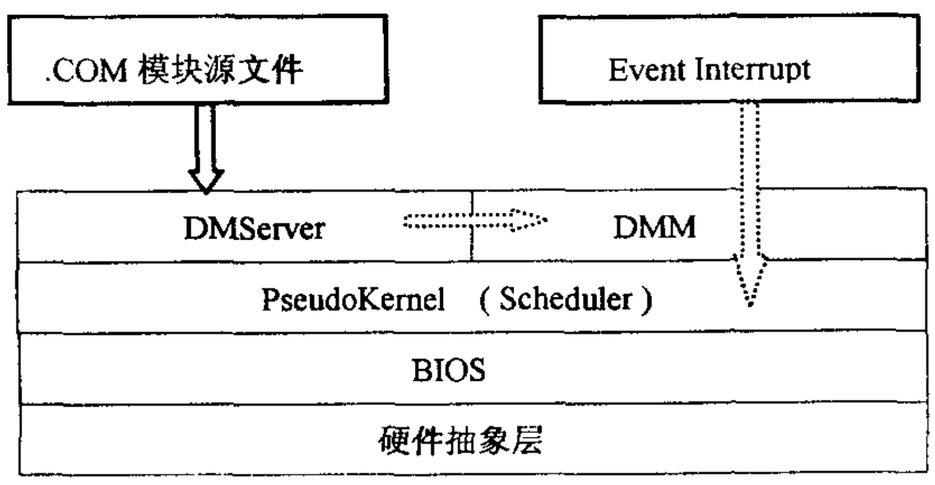

图 3 动态加载系统

PseudoKernel 主要用于完成控制转移的功能,即相当于一个运行管理核心,所以称之为伪内核。我们所设计的动态加载机制,整个系统只有一个动态加载机制,通过该机制可以把应用模块从零开始动态地加载来配置各种不同应用需要的嵌入式系统。而且当系统运行起来后,可以方便地对系统进行添加或者删除模块,从而达到系统的灵活性要求。当然了,在整个系统运行起来后,控制权仍然交还给伪内核。伪内核里面有一个很重要的模块——调度器(Scheduler)。该调度器采用优先级可抢占调度策略,并配以同优先级时间片轮转调度的调度方式,其主要作用是根据模块任务的优先级进行实时调度。文中研究实现的动态加载机制里的调度器总是运行进入就绪态任务中优先级最高的任务。最终实现该模块时,对任务级的调度是由函数模块 OSSched( )完成,而中断级的调度是由另一个函数模块 OSIntExt( )完成的[4]。

DMServer 是一个常驻 RAM 模块,它反复检测是否有模块文件需要加载,如果有的话,就会发送模块加载中断通知伪内核,随之,转入中断服务子程序,并对加载模块进行检测,是否满足约定的模块格式,并到模块管理器模块注册表中去搜索是否已经有与之相同的模块名字,如果没有找到,就对.COM 文件模块进行初始化,给任务模块分配内存空间,把模块加载到系统中,并建立模块名字表。模块管理器(DMM)采用模块链表对模块进行管理并完成对模块接口的定位和查找,当把该模块加载注册到模块管理器中,就把控制权交给伪内核。

4.3 动态加载系统实现演示实验

为了实现并演示该动态加载机制工作过程,选用两台桌面 PC 机作为工具环境,并约定,选用串口 COM1 进行通信,选择通信标准 RS-232-C 作为串口通信标准,采用零 Model 的最简连线[5](3 线制,见图 4)。在设计调度器时置该串口中断优先级为最高优先级别,当有模块需要加载时,系统就会立即执行。

当有模块需要加载时(主机 B 装载有动态加载机制),主机 A 向主机 B 发送信号 XON,B 检测到有 XON 信号后,就产生一个串口中断,由于该中断被定为最高优先级,当该中断到来后,PseudoKernel 就会停下当前运行的任务,跳到中断服务子程序,随后把控制权交给 DMServer,这时 DMServer 所作的动作有:

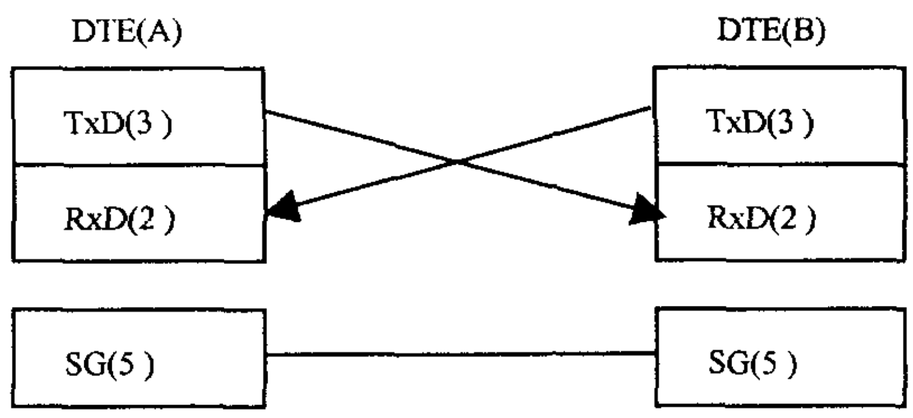

图 4 零 Model 的最简连线图

①初始化串口 COM1;

②检查是否有.COM 模块加载标志;

③有,跳入中断服务子程序,在内存中开辟空间,并加载之;

④无,继续执行当前程序。

当 DMServer 完成系列动作后,把模块加载到系统中后,就会对 A 发送一个加载完毕的信号 XOFF。随后系统控制权交还给 PseudoKernel,伪内核调度执行当前任务链中优先级最高级别的任务。

5 结 语

文中就基于.COM 文件模块格式的动态加载技术进行了探讨和设计,对其中涉计到的关键问题给予了解决方案,并提出和设计了一个实验性的原型,在 DOS 环境下的实验显示,能够初步实现动态加载的效果,并能够演示文中提出的思想:整个系统只有一个动态加载机制,可以把应用从零开始动态加载。但是,此原型还有需要改进和完善的地方,如 PseudoKernel 和 DMServer 间控制转移部分,可以改用多线程实现,还要考虑这种控制转移的快速和高效,以及加载模块本身格式要较严格地限制,需要改进等等。同时这些也是下一步的研究方向。

参考文献:

[1] Lyu Janghoon, Kim Youngin, Kim Yongsub, et al. A Procedure-Based Dynamic Software Update[J]. IEEE Computer, 2001,11(7):271-280.

[2] FRANZ M. Dynamic Linking of Software Components[J]. IEEE Computer,1997(3):74-81.

[3] Levine J R. linkers&loaders[M]. San Francisco: Morgan-Kauffman,1999.

[4] 邵贝贝. 嵌入式实时操作系统 uc/os-Ⅱ[M]. 第 2 版. 北京:北京航空航天大学出版社,2003:88-103.

[5] 串口通讯-RS-232-C 详解[EB/OL]. 2003. http://www.eebyte.com.