(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 105224370 B
(45)授权公告日 2019.03.08

(21)申请号 201510688388.3

(22)申请日 2015.10.21

(65)同一申请的已公布的文献号
申请公布号 CN 105224370 A

(43)申请公布日 2016.01.06

(73)专利权人 安一恒通(北京)科技有限公司
地址 100193 北京市海淀区东北旺西路8号4号楼软件广场C座1-01、1-03、1-04室

(72)发明人 秦松 陈鑫 位广军 马家智 桂敬文 王博通 王晓卿 张洪卫 常磊 张治 李新开

(74)专利代理机构 北京鸿德海业知识产权代理事务所(普通合伙) 11412
代理人 袁媛

(51)Int.Cl.
G06F 9/445(2018.01)

(56)对比文件
CN 101697131 A,2010.04.21,

审查员 吴朝烨

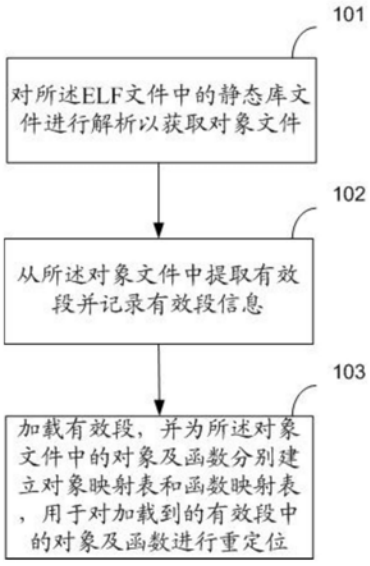
权利要求书2页 说明书10页 附图2页

(54)发明名称

一种加载ELF文件的方法和装置

(57)摘要

本发明提供了一种对具有可执行和链接格式(ELF)文件动态加载的方法和装置,其中方法包括:对所述ELF文件中的静态库文件进行解析以获取对象文件;从所述对象文件中提取有效段;加载有效段,并为所述对象文件中的对象及函数分别建立对象映射表和函数映射表,用于对加载的有效段中的对象及函数进行重定位。通过本发明所提供的ELF文件动态加载的方法和装置,可以摆脱现有技术中嵌入式系统在解析和加载ELF文件时需要依赖于特定的指令集,无法适用于全部的嵌入式处理器架构的困扰,实现了在特定处理器体系结构下能够同时运行多个ELF文件的应用,并且能够实现动态链接,大大减小了之前采用静态链接产生的文件体积,减小了功耗。



CN 105224370 B

1.一种加载具有可执行和链接格式ELF文件的方法,其特征在于,所述方法包括:

对所述ELF文件中的静态库文件进行解析以获取对象文件;

从所述对象文件中提取有效段;

不同类型的有效段分别加载到不同的区域,并为所述对象文件中的对象及函数分别建立对象映射表和函数映射表,用于对加载的有效段中的对象及函数进行重定位;其中,

将不同类型的有效段分别加载到不同的区域包括依照可执行代码段及相应重定位段的段信息,对可执行代码段进行重定位,将重定位后的可执行代码段写入Flash;或者依照应用数据段及相应重定位段的段信息,将应用数据段直接加载到RAM。

2.根据权利要求1所述的方法,其特征在于,提取的有效段包括:可执行代码段、应用数据段或重定位段。

3.根据权利要求1或2所述的方法,其特征在于,根据所述对象文件的ELF文件头和段文件头表来提取有效段。

4.根据权利要求1所述的方法,其特征在于,所述段信息包括段序号、段偏移量或段大小。

5.根据权利要求4所述的方法,其特征在于,所述可执行代码段包括.text段,所述应用数据段包括.data段、.bss段或.rodata段,所述重定位段包括.rel.data段、.rel.bss段或.rel.rodata段。

6.根据权利要求1或2所述的方法,其特征在于,在所述加载有效段之后,对所述对象文件的有效段进行以下至少一个的标记:段的起始地址、段偏移量、段大小。

7.根据权利要求1所述的方法,其特征在于:

所述对象映射表的映射表项包括对象名称和对象地址,所述对象地址是通过查找有效段中的导出对象,根据导出对象所在地址而得到的;

所述函数映射表的映射表项包括函数名称和函数地址,所述函数地址是通过查找有效段中的导出函数,根据导出函数所在的地址而得到的。

8.根据权利要求1所述的方法,其特征在于,在对有效段中的对象进行重定位时,利用当前对象在所述对象映射表中的地址查找所述对象的跳转对象在文字池中的地址,并将文字池中所述跳转对象的地址修改为当前对象的地址。

9.根据权利要求1所述的方法,其特征在于,在对有效段中的函数进行重定位时,利用当前函数运行时的地址以及从函数映射表中查找到的需要跳转到的函数地址,经过计算得到跳转指令,利用计算得到的跳转指令对之前的跳转指令进行更新,以便于根据所述更新的跳转指令跳转到即将要执行的函数地址。

10.一种加载ELF文件的装置,其特征在于,所述装置包括:

解析装置,用于对所述ELF文件中的静态库文件进行解析以获取对象文件;

提取装置,用于从所述对象文件中提取有效段;

加载装置,用于将不同类型的有效段分别加载到不同的区域,为所述对象文件中的对象及函数分别建立对象映射表和函数映射表,并且用于对加载的有效段中的对象及函数进行重定位;其中,

所述加载装置在将不同类型的有效段分别加载到不同的区域时,具体执行依照可执行代码段及相应重定位段的段信息,对可执行代码段进行重定位,将重定位后的可执行代码

段写入Flash;或者依照应用数据段及相应重定位段的段信息,将应用数据段直接加载到RAM。

11.根据权利要求10所述的加载ELF文件的装置,其特征在于,所述提取装置中提取的有效段包括:可执行代码段、应用数据段或重定位段。

12.根据权利要求10或11所述的加载ELF文件的装置,其特征在于,所述提取装置用于根据所述对象文件的ELF文件头和段文件头表来提取有效段。

13.根据权利要求10所述的加载ELF文件的装置,其特征在于,所述段信息包括段序号、段偏移量或段大小。

14.根据权利要求13所述的加载ELF文件的装置,其特征在于,所述可执行代码段包括.text段,所述应用数据段包括.data段、.bss段或.rodata段,所述重定位段包括.rel.data段、.rel.bss段或.rel.rodata段。

15.根据权利要求10或11所述的加载ELF文件的装置,其特征在于,所述加载装置在执行所述加载有效段之后,进一步用于对所述对象文件的有效段进行以下至少一个的标记:段的起始地址、段偏移量、段大小。

16.根据权利要求10所述的加载ELF文件的装置,其特征在于:

所述对象映射表的映射表项包括对象名称和对象地址,所述对象地址是通过查找有效段中的导出对象,根据导出对象所在地址而得到的;

所述函数映射表的映射表项包括函数名称和函数地址,所述函数地址是通过查找有效段中的导出函数,根据导出函数所在的地址而得到的。

17.根据权利要求10所述的加载ELF文件的装置,其特征在于,所述加载装置在对有效段中的对象进行重定位时,具体执行:

利用当前对象在所述对象映射表中的地址查找所述对象的跳转对象在文字池中的地址,并将文字池中所述跳转对象的地址修改为当前对象的地址。

18.根据权利要求10所述的加载ELF文件的装置,其特征在于,所述加载装置在对有效段中的函数进行重定位时,具体执行:利用当前函数运行时的地址以及从函数映射表中查找到的需要跳转到的函数地址,经过计算得到跳转指令,利用计算得到的跳转指令对之前的跳转指令进行更新,以便于根据所述更新的跳转指令跳转到即将要执行的函数地址。

一种加载ELF文件的方法和装置

【技术领域】

[0001] 本发明涉及计算机领域，尤其涉及一种加载ELF文件的方法和装置。

【背景技术】

[0002] 具有可执行和链接格式的ELF(Executable and Linking Format)是一种对象文件的格式，用于定义不同类型的对象文件(Object files)中的内容及其格式。由于现有的嵌入式操作系统对于ELF文件的加载采用的是静态链接的方式生成可执行文件，并采用烧录的方式将可执行文件直接烧录至内存(Flash)中，这些处理过程都必须在ARM指令集下完成的，而有些处理器采用的并非ARM指令集，且不支持其所采用的指令集到ARM指令集的切换，因此现有技术无法为此类处理器提供ELF文件的加载。

【发明内容】

[0003] 本发明提出了一种加载ELF文件的方法和装置，以便于解决现有技术在一些嵌入式系统中无法记载ELF文件的问题。

[0004] 具体技术方案如下：

[0005] 一种加载具有可执行和链接格式ELF文件的方法，所述方法包括：

[0006] 对所述ELF文件中的静态库文件进行解析以获取对象文件；

[0007] 从所述对象文件中提取有效段；

[0008] 加载有效段，并为所述对象文件中的对象及函数分别建立对象映射表和函数映射表，用于对加载的有效段中的对象及函数进行重定位。

[0009] 根据本发明一优选实施例，提取的有效段包括：可执行代码段、应用数据段或重定位段。

[0010] 根据本发明一优选实施例，根据所述对象文件的ELF文件头和段文件头表来提取有效段。

[0011] 根据本发明一优选实施例，所述加载有效段包括：将不同类型的有效段分别加载到不同的区域。

[0012] 根据本发明一优选实施例，所述将不同类型的有效段分别加载到不同的区域包括：

[0013] 依照可执行代码段及相应重定位段的段信息，对可执行代码段进行重定位，将重定位后的可执行代码段写入Flash；或者

[0014] 依照应用数据段及相应重定位段的段信息，将应用数据段直接加载到RAM。

[0015] 根据本发明一优选实施例，所述段信息包括段序号、段偏移量或段大小。

[0016] 根据本发明一优选实施例，所述可执行代码段包括.text段，所述应用数据段包括.data段、.bss段、或.rodata段，所述重定位段包括.rel.data段、.rel.bss段、或.rel.rodata段。

[0017] 根据本发明一优选实施例，在所述加载有效段之后，对所述对象文件的有效段进

行以下至少一个的标记:段的起始地址、段偏移量、段大小。

[0018] 根据本发明一优选实施例,所述对象映射表的映射表项包括对象名称和对象地址,所述对象地址是通过查找有效段中的导出对象,根据导出对象所在地址而得到的;所述函数映射表的映射表项包括函数名称和函数地址,所述函数地址是通过查找有效段中的导出函数,根据导出函数所在的地址而得到的。

[0019] 根据本发明一优选实施例,在对有效段中的对象进行重定位时,利用当前对象在所述对象映射表中的地址查找所述对象的跳转对象在文字池中的地址,并将文字池中所述跳转对象的地址修改为当前对象的地址。

[0020] 根据本发明一优选实施例,在对有效段中的函数进行重定位时,利用当前函数运行时的地址以及从函数映射表中查找到的需要跳转到的函数地址,经过计算得到跳转指令,利用计算得到的跳转指令对之前的跳转指令进行更新,以便于根据所述更新的跳转指令跳转到即将要执行的函数地址。

[0021] 一种加载ELF文件的装置,所述装置包括:

[0022] 解析装置,用于对所述ELF文件中的静态库文件进行解析以获取对象文件;

[0023] 提取装置,用于从所述对象文件中提取有效段;

[0024] 加载装置,用于加载有效段,为所述对象文件中的对象及函数分别建立对象映射表和函数映射表,并且用于对加载的有效段中的对象及函数进行重定位。

[0025] 根据本发明一优选实施例,所述提取装置中提取的有效段包括:可执行代码段、应用数据段或重定位段。

[0026] 根据本发明一优选实施例,所述提取装置用于根据所述对象文件的ELF文件头和段文件头表来提取有效段。

[0027] 根据本发明一优选实施例,所述加载装置加载有效段包括:将不同类型的有效段分别加载到不同的区域。

[0028] 根据本发明一优选实施例,所述加载装置在将不同类型的有效段分别加载到不同的区域时,具体执行:

[0029] 依照可执行代码段及相应重定位段的段信息,对可执行代码段进行重定位,将重定位后的可执行代码段写入Flash;或者

[0030] 依照应用数据段及相应重定位段的段信息,将应用数据段直接加载到RAM。

[0031] 根据本发明一优选实施例,所述段信息包括段序号、段偏移量或段大小。

[0032] 根据本发明一优选实施例,所述可执行代码段包括.text段,所述应用数据段包括.data段、.bss段、或.rodata段,所述重定位段包括.rel.data段、.rel.bss段、或.rel.rodata段。

[0033] 根据本发明一优选实施例,所述加载装置在执行所述加载有效段之后,进一步用于对所述对象文件的有效段进行以下至少一个的标记:段的起始地址、段偏移量、段大小。

[0034] 根据本发明一优选实施例,所述对象映射表的映射表项包括对象名称和对象地址,所述对象地址是通过查找有效段中的导出对象,根据导出对象所在地址而得到的;所述函数映射表的映射表项包括函数名称和函数地址,所述函数地址是通过查找有效段中的导出函数,根据导出函数所在的地址而得到的。

[0035] 根据本发明一优选实施例,所述加载装置在对有效段中的对象进行重定位时,具

体执行：

[0036] 利用当前对象在所述对象映射表中的地址查找所述对象的跳转对象在文字池中的地址，并将文字池中所述跳转对象的地址修改为当前对象的地址。

[0037] 根据本发明一优选实施例，所述加载装置在对有效段中的函数进行重定位时，具体执行：利用当前函数运行时的地址以及从函数映射表中查找到的需要跳转到的函数地址，经过计算得到跳转指令，利用计算得到的跳转指令对之前的跳转指令进行更新，以便于根据所述更新的跳转指令跳转到即将要执行的函数地址。

[0038] 由以上技术方案可以看出，本发明采用了解析静态库(.a)文件并且提取有效段，对有效段进行加载，建立对象及函数映射表用于对可重定位段中的对象和函数分别进行重定位的方式，这些处理方式并不受限于特定指令集，因此解决了现有技术在一些嵌入式系统中无法加载ELF文件的问题。

【附图说明】

[0039] 图1为本发明实施例提供的一种对ELF文件动态加载方法的流程示意图；

[0040] 图2为本发明实施例提供的ELF Header的格式示意图；

[0041] 图3为本发明实施例提供的一种加载ELF文件的装置的结构示意图。

【具体实施方式】

[0042] 为了使本发明的目的、技术方案和优点更加清楚，下面结合附图和具体实施例对本发明进行详细描述。

[0043] 请参考图1，图1为本发明提供的一种加载具有可执行和链接格式的ELF文件的方法实施例的流程示意图。如图1所示，该流程包括：

[0044] 101：对ELF文件中的静态库(.a)文件进行解析以获取对象文件(.o)。

[0045] 本发明的对象文件(.o)是以ELF格式保存的可重定向文件，对象文件中的内容可以包含对各个函数的入口标记，描述，以形成机器可执行的指令。当程序要执行时还需要进行链接(link)，链接就是把多个.o文件链成一个可执行文件。

[0046] ELF文件除了可以包含可重定位的对象文件外，还可以包含可执行的对象文件和可被共享的对象文件，上述三种对象文件由于都可以以对象文件.o的形式表示，而多个.o对象文件可以归档(archive)成.a静态库文件，从而最终组成ELF文件，因此本发明中解析静态库(.a)文件的目的就是为了获取组成ELF文件的对象文件(.o)。

[0047] 具体地，解析静态库文件的过程与windows操作系统中一个压缩包rar的解压缩(还原)过程类似，也就是分析由对象文件(.o)打包生成的静态库(.a)，解析其中每个对象文件(.o)。

[0048] 102：从获取的对象文件中提取有效段(section)。

[0049] 当获取对象文件(.o)后，可以针对每一个对象文件，从中提取有效段并且记录段信息。

[0050] 有效段(sections)在ELF文件里面，用以装载内容数据的最小容器。

[0051] 优选地，可以针对每个获取的对象文件，根据所述对象文件的ELF文件头(ELF Header)和段文件头表(SHT，Section Header Table)来提取有效段并记录各段信息。

[0052] 该有效段的具体类型可以包括:可执行代码段,应用数据段,或重定位段。其中该可执行代码段又可以具体包括.text段等,该应用数据段可以包括.data段、.bss段、或.rodata段等,该重定位段可以包括.rel.data段、.rel.bss段、或rel.rodata段等。

[0053] 该记录的有效段信息包括但不限于:各段序号、段偏移量、或段大小等。

[0054] 其中

[0055] 1).text段里装载了可执行代码;

[0056] 2).data段里装载了被初始化的数据;

[0057] 3).bss段里装载了未被初始化的数据;

[0058] 4).rodata段里装载了相应字符串和定义的常量

[0059] 5)以.rel打头的重定位段rel.text段、.rel.data段、.rel.bss段、.rel.rodata段里面装载了重定位条目,该重定位段是用于帮助可执行代码段和应用数据段进行重定位的段;

[0060] 例如:.rel.text中示出了.text段中哪些地方需要做重定位。

[0061] 以下举个具体实例以说明从对象文件的ELF Header和SHT中提取有效段及各段信息的具体方式:

[0062] 本实施例中的ELF Header可以以图2所示的格式表示。

[0063] 如图2所示,该ELF Header表示了可重定位对象文件(.o)的格式。可以从该ELF Header中得知相应的sections及序号,而一个ELF文件中到底有哪些具体的sections,则反映在这个ELF文件中的SHT中。

[0064] SHT表中条目的结构定义可以为:

[0065]
```
typedef struct {
        Elf32_Word      sh_name;
        Elf32_Word      sh_type;
        Elf32_Word      sh_flags;
        Elf32_Addr      sh_addr;
        Elf32_Off       sh_offset;
        Elf32_Word      sh_size;
        Elf32_Word      sh_link;
        Elf32_Word      sh_info;
        Elf32_Word      sh_addralign;
        Elf32_Word      sh_entsize;
} Elf32_Shdr;
```

[0066] 在SHT中,针对每一个section,都设置有一个条目,用来描述对应的这个section,

其内容主要包括该section的名称、类型、大小以及在整个ELF文件中的字节偏移位置等等。

[0067] 上述SHT表中条目的结构中字段Off(sh_offset)表示了该section离开文件头部位置的距离,即偏移量;Size(sh_size)表示section的段大小。

[0068] 从而根据ELF Header和SHT表示的相关section从而能够提取各有效段的信息。

[0069] 以上的101和102是在动态加载ELF文件之前对加载的准备过程,其步骤可以同时或者分先后执行,本发明对其执行的顺序不做硬性规定,当上述准备过程执行后,则通过103进一步实现对对象文件的有效段的加载。

[0070] 103:加载有效段,并为所述对象文件中的对象及函数分别建立对象映射表和函数映射表,用于对加载到的有效段中的对象及函数进行重定位。

[0071] 对于103,还可以将其进一步划分为加载有效段、建立映射表以及重定位三个步骤,其中前两个步骤可以同时或者分先后执行,本发明对其执行的顺序不做硬性规定,只要能完成相应的功能,则均在本发明保护范围之内。

[0072] 以下对三个步骤进行详细描述:

[0073] 第一步:加载有效段。

[0074] 对于在102所提取的有效段,包括可执行代码段、应用数据段以及重定位段三种类型,其中可执行代码段和应用数据段属于需要加载的有效段,而重定位段不需要加载,针对该需要加载的有效段,可以依据其类型的不同可以对应的采用两种方式将其加载到不同的区域。

[0075] 第一种方式,对于可执行代码段(.text段)加载:

[0076] 由于针对于不同的产品其flash的读写不一致,因此为了便于重定位修改.text段内容,可以先对.text段进行重定位,然后再将其写入Flash。

[0077] 由于提取的与.text段相应的重定位段(.rel.text段)中包含.text段需要重定位的信息,因此在写入flash之前的重定位方式可以为:将识别出的.rel.text段中记录的.text段需要重定位的部分记录下来创建重定位信息,对.text做重定位,然后再将.text段拷贝并写入到Flash中。

[0078] 第二种方式,对于应用数据段(.data、.bss及.rodata)的加载:

[0079] 对于.data、.bss及.rodata的加载方式与对.text段的加载方式略有不同,相比于对.text段的加载在写入到Flash之前还包括对.text段的重定位过程而言,对.data、.bss及.rodata的加载在将各段加载到随机存取存储器(Random Access Memory,RAM)之前并不需要进行重定位,而是依照.data、.bss及.rodata及相应重定位段(..rel.data、.rel.bss及.rel.rodata)的有效段信息,将.data、.bss及.rodata段直接加载到RAM。

[0080] 在通过第一种和第二种方式加载有效段后,可以对各个对象文件(.o)的各个段做好相应的标记,该标记可以包括但不限于段的起始地址、段偏移量、段大小等信息。

[0081] 其中段的起始地址是根据对加载后的有效段初始化获得的,段偏移量、段大小是在提取有效段时基于该提取的有效段获得的。

[0082] 在第一步中,通过将装载了可执行代码的.text段与其他的应用数据段的应用数据分离,并通过对可执行代码的目标文件创建重定位信息并载入内存,从而生成内存映射过程,避免了在加载ELF文件时多次拷贝代码数据的过程,提高了运行效率。

[0083] 第二步,为对象文件中的对象及函数分别建立对象映射表和函数映射表。

[0084] 在本步骤中，按照对象及函数的区别分别建立对象映射表(obj_map)及函数映射表(func_map)，所建立的映射表供后续重定位及外部调用使用。

[0085] 其中对象映射表的映射表项包括对象名称和对象地址，该映射表中的对象地址是通过查找对象文件的有效段中的导出对象，根据导出对象所在地址而得到的；函数映射表的映射表项包括函数名称和函数地址，该函数地址是通过查找对象文件的有效段中的导出函数，根据导出函数所在的地址而得到的。

[0086] 关于对象和函数，举个例子：其中.text段中包含的大部分为函数，而.text段、.data段、.bss段、.rodata段中文件变量、函数变量等全局变量、函数元素等都可以以对象表示。由于对象和函数的划分为现有技术，在此不再赘述。

[0087] 第三步，对加载到的有效段中的对象及函数进行重定位。

[0088] 在本步骤中，使用对象映射表对有效段中的对象进行重定位，或使用所述函数映射表对有效段中的函数进行重定位。

[0089] 具体地，在链接时，需要对所有目标文件的可重定位段进行重定位，建立符号引用规则，同时为对象，函数等分配地址。程序执行时，把代码加载到链接时指定的地址空间，以保证程序在执行过程中对对象，函数等符号的正确引用，使程序正常运行。

[0090] 基于对象文件中提取出的对象及函数类型的不同，可以采用两种不同的重定位方式。

[0091] 第一种方式，对对象(obj)进行重定位：

[0092] 对象在.o文件中的使用方式是，直接将对象的真实地址写入文字池。在动态加载此对象时，利用当前对象在所述对象映射表中的地址查找所述对象的跳转对象在文字池中的地址，并将文字池中所述跳转对象的地址修改为当前对象的地址。

[0093] 具体地，编译器会把对象的地址都放到文字池(Literal Pool)区域。其中Literal Pool里面只存放绝对地址或常量。当需要进行重定位时，根据.rel.text、.rel.data、.rel.bss及.rel.rodata所包含的对象重定位信息，通过对象映射表中所记录的对象的当前地址获取文字池中所记录的跳转对象的绝对地址，从Literal Pool中查找绝对地址，在动态加载时，将对象所在的文字池中的绝对地址修改为当前对象所在的地址，从而对.text、.data、.bss及.rodata段包含的对象做重定位。

[0094] 第二种方式，对函数(func)进行重定位：

[0095] 对于函数的重定位，可以利用当前函数运行时的地址以及从函数映射表中查找到的需要跳转到的函数地址，经过计算得到跳转指令，利用计算得到的跳转指令对之前的跳转指令进行更新，以便于根据所述更新的跳转指令跳转到即将要执行的函数地址。

[0096] 具体地，函数调用跳转采用相应的跳转指令，例如B/BL等，对函数进行重新定位时，基于.rel.text等所包含的函数重定位信息，从函数映射表中查找需要跳转到的函数的真实地址，根据函数运行时所在的当前地址结合从函数映射表中查找到的需要跳转到的函数真实地址，经过一系列计算得到跳转指令，获取正确的跳转指令对之前的跳转指令进行更新，从而根据更新后的跳转指令加载重定位的函数。

[0097] 当然，并非只能依据对象映射表和函数映射表来查找地址，由于动态加载时对象可能引用的是外部对象或者函数使用的是动态库函数，因此当在对象或函数映射表中无法找到相应的地址信息时，则表明对象或函数引用的是外部对象或者动态库函数，此时可以

去动态的查找对象或函数的地址以便重定位,从而实现动态加载。

[0098] 请参考图3,图3为本发明提供的一种加载ELF文件的装置实施例的结构示意图。如图3所示,该装置包括:解析装置301,提取装置302和加载装置303,以下对各装置进行详细描述:

[0099] 解析装置301,用于对ELF文件中的静态库(.a)文件进行解析以获取对象文件(.o)。

[0100] 本发明的对象文件(.o)是以ELF格式保存的可重定向文件,对象文件中的内容可以包含对各个函数的入口标记,描述,以形成机器可执行的指令。当程序要执行时还需要进行链接(link),链接就是把多个.o文件链成一个可执行文件。多个.o对象文件可以归档(archive)成.a静态库文件,而静态库文件行成了ELF文件。

[0101] 其中解析装置解析静态库(.a)文件的目的是为了获取其中的对象文件(.o)。

[0102] 具体地,解析静态库文件的过程与windows操作系统中一个压缩包rar的解压缩(还原)过程类似,也就是分析由对象文件(.o)打包生成的静态库(.a),解析其中每个对象文件(.o)。

[0103] 提取装置302,用于从获取的对象文件中提取有效段(section)。

[0104] 当获取对象文件(.o)后,可以针对每一个对象文件,从中提取有效段并且记录段信息。

[0105] 有效段(sections)在ELF文件里面,用以装载内容数据的最小容器。在ELF文件里,每一个sections内都装载了性质属性相同的内容。,

[0106] 优选地,可以针对每个获取的对象文件,根据所述对象文件的ELF文件头(ELF Header)和段文件头表(SHT,Section Header Table)来提取有效段并记录各段信息。

[0107] 该有效段的具体类型可以包括:可执行代码段,应用数据段,或重定位段。其中该可执行代码段又可以具体包括.text段等,该应用数据段可以包括.data段、.bss段、或.rodata段等,该重定位段可以包括.rel.data段、.rel.bss段、或rel.rodata段等。

[0108] 该记录的有效段信息包括但不限于:各段序号、段偏移量、或段大小等。

[0109] 其中

[0110] 1).text段里装载了可执行代码;

[0111] 2).data段里装载了被初始化的数据;

[0112] 3).bss段里装载了未被初始化的数据;

[0113] 4).rodata段里装载了相应字符串和定义的常量

[0114] 5)以.rel打头的重定位段rel.text段、.rel.data段、.rel.bss段、.rel.rodata段里面装载了重定位条目,该重定位段是用于帮助可执行代码段和应用数据段进行重定位的段;

[0115] 例如:.rel.text中示出了.text段中哪些地方需要做重定位。

[0116] 从对象文件的ELF Header和SHT中提取有效段及各段信息的具体方式可以为:

[0117] 从ELF Header中得知相应的sections及序号,从SHT的内容中,根据SHT中的字段Off(sh_offset)、Size(sh_size)来获取段偏移量、段大小等信息。

[0118] 装置301和302用于在动态加载ELF文件之前对预加载的ELF文件执行加载前的准备,其准备过程可以同时或者分先后执行,本发明对其执行的顺序不做硬性规定,当上述准

备过程执行后，则通过装置303进一步实现对对象文件的有效段的加载。

[0119] 加载装置303，用于加载有效段，为所述对象文件中的对象及函数分别建立对象映射表和函数映射表，并且用于对加载到的有效段中的对象及函数进行重定位。

[0120] 还可以将加载装置303进一步划分为有效段加载单元3031、映射表建立单元3032以及重定位单元3033三个单元，其中前两个单元可以同时或者分先后执行相应操作，本发明对其执行的顺序不做硬性规定，只要能完成相应的功能，则均在本发明保护范围之内。

[0121] 以下对三个单元进行详细描述：

[0122] 加载有效段单元3031。

[0123] 对于由提取单元302所提取的有效段，包括可执行代码段、应用数据段以及重定位段三种类型，其中可执行代码段和应用数据段属于需要加载的有效段，而重定位段不需要加载，针对该需要加载的有效段，可以依据其类型的不同可以对应的采用两种方式将其加载到不同的区域。

[0124] 第一种方式，对于可执行代码段(.text段)加载：

[0125] 由于针对于不同的产品其flash的读写不一致，因此为了便于重定位修改.text段内容，可以先对.text段进行重定位，然后再将其写入Flash。

[0126] 由于提取的与.text段相应的重定位段(.rel.text段)中包含.text段需要重定位的信息，因此在写入flash之前的重定位方式可以为：将识别出的.rel.text段中记录的.text段需要重定位的部分记录下来创建重定位信息，对.text做重定位，然后再将.text段拷贝并写入到Flash中。

[0127] 第二种方式，对于应用数据段(.data、.bss及.rodata)的加载：

[0128] 对于.data、.bss及.rodata的加载方式与对.text段的加载方式略有不同，相比于对.text段的加载在写入到Flash之前还包括对.text段的重定位过程而言，对.data、.bss及.rodata的加载在将各段加载到随机存取存储器(Random Access Memory，RAM)之前并不需要进行重定位，而是依照.data、.bss及.rodata及相应重定位段(.rel.data、.rel.bss及.rel.rodata)的有效段信息，将.data、.bss及.rodata段直接加载到RAM。

[0129] 在通过第一种和第二种方式加载有效段后，可以对各个对象文件(.o)的各个段做好相应的标记，该标记可以包括但不限于段的起始地址、段偏移量、段大小等信息。

[0130] 其中段的起始地址是根据对加载后的有效段初始化获得的，段偏移量、段大小是在提取有效段时基于该提取的有效段获得的。

[0131] 本实施例通过加载有效段单元的操作，将装载了可执行代码的.text段与其他的应用数据段的应用数据分离，并通过对可执行代码的目标文件创建重定位信息并载入内存，从而生成内存映射过程，避免了在加载ELF文件时多次拷贝代码数据的过程，提高了运行效率。

[0132] 映射表建立单元3032，用于为对象文件中的对象及函数分别建立对象映射表和函数映射表。

[0133] 本单元用于按照对象及函数的区别分别建立对象映射表(obj_map)及函数映射表(func_map)，所建立的映射表供后续重定位及外部调用使用。

[0134] 其中对象映射表的映射表项包括对象名称和对象地址，该映射表中的对象地址是通过查找对象文件的有效段中的导出对象，根据导出对象所在地址而得到的；函数映射表

的映射表项包括函数名称和函数地址，该函数地址是通过查找对象文件的有效段中的导出函数，根据导出函数所在的地址而得到的。

[0135] 关于对象和函数，举个例子：其中.text段中包含的大部分为函数，而.text段、.data段、.bss段、.rodata段中文件变量、函数变量等全局变量、函数元素等都可以以对象表示。由于对象和函数的划分为现有技术，在此不再赘述。

[0136] 重定位单元3033，用于对加载到的有效段中的对象及函数进行重定位。

[0137] 本单元用于使用对象映射表对有效段中的对象进行重定位，或使用所述函数映射表对有效段中的函数进行重定位。

[0138] 具体地，在链接时，需要对所有目标文件的可重定位段进行重定位，建立符号引用规则，同时为对象，函数等分配地址。程序执行时，把代码加载到链接时指定的地址空间，以保证程序在执行过程中对对象，函数等符号的正确引用，使程序正常运行。

[0139] 基于对象文件中提取出的对象及函数类型的不同，可以采用两种不同的重定位方式。

[0140] 第一种方式，对对象(obj)进行重定位：

[0141] 对象在.o文件中的使用方式是，直接将对象的真实地址写入文字池。在动态加载此对象时，利用当前对象在所述对象映射表中的地址查找所述对象的跳转对象在文字池中的地址，并将文字池中所述跳转对象的地址修改为当前对象的地址。

[0142] 第二种方式，对函数(func)进行重定位：

[0143] 对于函数的重定位，可以利用当前函数运行时的地址以及从函数映射表中查找到的需要跳转到的函数地址，经过计算得到跳转指令，利用计算得到的跳转指令对之前的跳转指令进行更新，以便于根据所述更新的跳转指令跳转到即将要执行的函数地址。

[0144] 当然，并非只能依据对象映射表和函数映射表来查找地址，由于动态加载时对象可能引用的是外部对象或者函数使用的是动态库函数，因此当在对象或函数映射表中无法找到相应的地址信息时，则表明对象或函数引用的是外部对象或者动态库函数，此时可以查找对象或函数的地址以便重定位，从而实现动态加载。

[0145] 本发明的ELF文件动态记载的方式和装置，可以应用于机顶盒、智能可穿戴式设备、智能家居云等多种场景中。以机顶盒为例，当用户使用机顶盒启动或者安装ELF文件时，可以使用本发明所提供的方法和装置，通过对静态库文件(.a)进行解析并进行加载和重定位，从而实现ELF文件的动态加载。

[0146] 本发明提供的方法和装置可以具备下优点：

[0147] 1)本发明采用了解析静态库(.a)文件并且提取有效段，对有效段进行加载，建立对象及函数映射表用于对可重定位段中的对象和函数分别进行重定位的方式，这些处理方式并不受限于特定指令集，因此解决了现有技术在一些嵌入式系统中无法加载ELF文件的问题。

[0148] 举一个例子，本发明尤其适用于支持采用Thumb-2指令集的Cortex-M系列处理器嵌入式系统。由于嵌入式Mbed OS是基于ARM Cortex-M处理器所设计的操作系统，Mbed os作为专门为嵌入式开发定制的系统具有独特的优势。目前Cortex-M系列处理器已经能够支持Mbed os，但目前Mbed os还无法提供适用于Cortex-M处理器的ELF文件加载器(Loader)，因此限制了Mbed os在Cortex-M系列处理器上的使用。这是因为现有的嵌入式操作系统对

于ELF文件的加载采用静态链接的方式生成可执行文件，并将可执行文件烧录至Flash，这些操作都必须在ARM指令集下完成，但是由于Cortex-M系列处理器指令集采用的是Thumb-2指令集，该指令集不支持切换到ARM指令集的状态，因此现有技术无法为此类处理器提供动态解析加载ELF文件的加载器。

[0149] 而本发明提供的方法和装置转换了一种思路，不是通过静态链接的方式生成可执行文件并直接载入内存的方式，也不需要通过特定的ARM指令集来解析和加载ELF文件。而是对ELF文件的静态库文件进行解析以获取ELF可重定位的对象文件的有效段并进行重定位，以便使ELF文件中的对象及函数能够以可重定位的方式被动态加载。本发明的这些处理过程并不受限于特定的ARM指令集，因此尤其适用于支持采用Thumb-2指令集的Cortex-M系列处理器嵌入式系统。

[0150] 2)现有技术中采用静态链接的方式生成可执行文件，并将可执行文件烧录至Flash，因此其运行的应用数量受限于Flash中烧录的可执行文件数量。而本发明采用动态加载方式，基于此能够实现动态链接，运行的应用数量不再受限于Flash中烧录可执行文件数量。

[0151] 3)静态链接的方式需要生成可执行文件并载入Flash，这就造成了文件体积较大，功耗较大。本申请的动态加载方式相比较现有技术大大较小了文件体积和功耗。

[0152] 在本发明所提供的几个实施例中，应该理解到，所揭露的系统，装置和方法，可以通过其它的方式实现。例如，以上所描述的装置实施例仅仅是示意性的，例如，所述单元的划分，仅仅为一种逻辑功能划分，实际实现时可以有另外的划分方式。

[0153] 所述作为分离部件说明的单元可以是或者也可以不是物理上分开的，作为单元显示的部件可以是或者也可以不是物理单元，即可以位于一个地方，或者也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部单元来实现本实施例方案的目的。

[0154] 另外，在本发明各个实施例中的各功能单元可以集成在一个处理单元中，也可以是各个单元单独物理存在，也可以两个或两个以上单元集成在一个单元中。上述集成的单元既可以采用硬件的形式实现，也可以采用硬件加软件功能单元的形式实现。

[0155] 以上所述仅为本发明的较佳实施例而已，并不用以限制本发明，凡在本发明的精神和原则之内，所做的任何修改、等同替换、改进等，均应包含在本发明保护的范围之内。

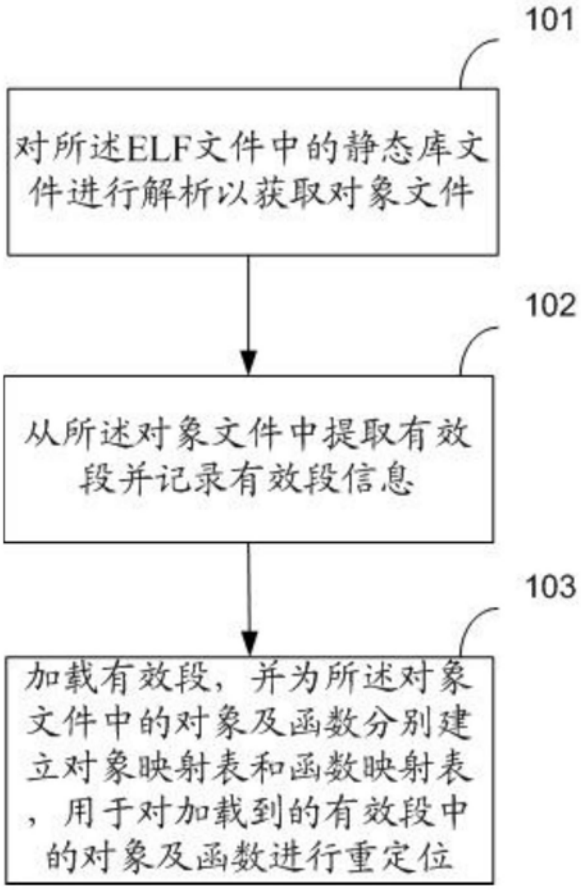

101
对所述ELF文件中的静态库文件进行解析以获取对象文件
102
从所述对象文件中提取有效段并记录有效段信息
103
加载有效段，并为所述对象文件中的对象及函数分别建立对象映射表和函数映射表，用于对加载到的有效段中的对象及函数进行重定位

图1

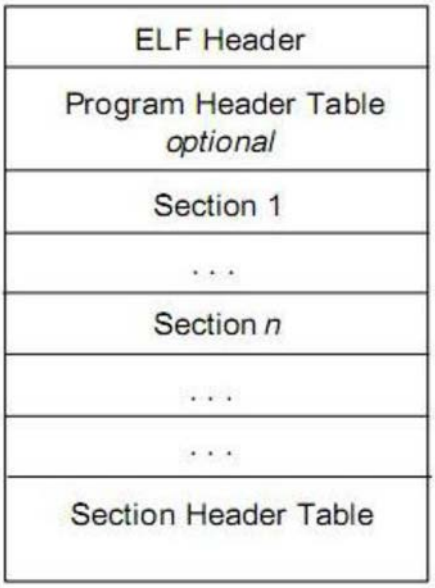

ELF Header
Program Header Table
optional
Section 1
. . .
Section n
. . .
. . .
Section Header Table

图2

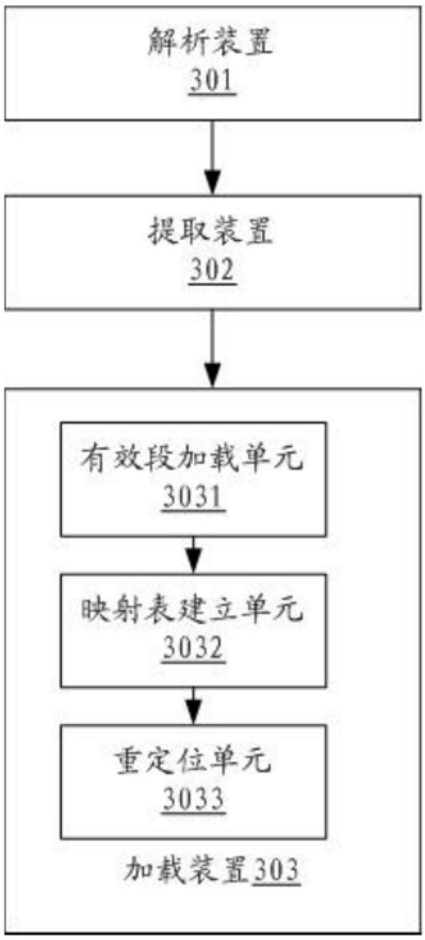

解析装置
301
提取装置
302
有效段加载单元
3031
映射表建立单元
3032
重定位单元
3033
加载装置303

图3