Linux 裁减技术研究

彭 丹
(贵州大学电子科学与信息技术学院，贵州 贵阳 550003)

摘 要:介绍了嵌入式 Linux 系统中的裁减方法,分析了共享库裁剪技术的原理,最后提出了 Linux 裁减的一般顺序和步骤。
关键词:Linux; 共享库; 裁减

引言

嵌入式 Linux 裁减技术简介:

现今,Linux 已经越来越广泛地应用于手机、摄像机、PDA、数控机床等各种各样的嵌入式设备当中。但是一般的应用于个人 PC 上的 Linux 所需要的空间都很庞大,很难用于只有有限存储空间的嵌入式设备，所以我们必须对 Linux 系统进行一些必要的裁减。

Linux 系统大致有以下 4 种主要的裁剪技术,使用这些技术可以有效地减小系统的尺寸且不会影响系统的性能。

① 删除冗余文件。包括很多的帮助文档、辅助程序、配置文件和数据模板,以及大量的注释。

② 共享库裁剪。嵌入式系统的应用程序是有限的,共享库中就可能有很多永远不会被用到的冗余代码，这些代码就可以被删除。

③ 采用具有同样功能的替代软件包。Linux 上有许多具有相似功能的软件包，可以选择其中占存储空间较小的软件包并其移植到嵌入式设备上，用来代替原来占空间较大那些的软件包。

④ 修改源码。包括重新配置、编译软件包,去掉不需要的功能;增加软件的模块性,从而有利于提高裁剪效率;重新配置内核,去掉不需要的驱动和模块。

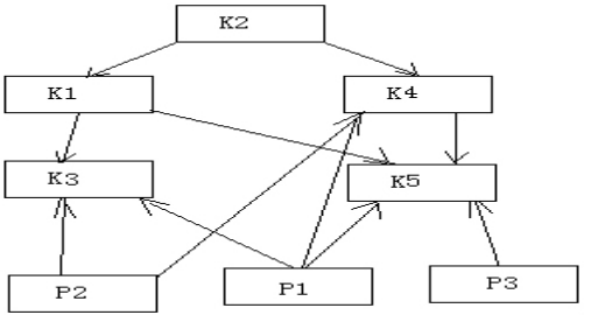
1 共享库裁减技术

1.1 共享库文件裁减

共享库文件裁减是一个比较易于用软件实现裁减技术,而且裁减后系统空间可以比较明显的降低。下面介绍一些它的裁减原理. 共享库的裁减基本原理是通过提取和解析系统库文件的依赖关系,通过对这些依赖构造关系模型进行关系演算,根据应用程序中的函数中的调用信息，能够在共享库中实现函数一级的裁减.它的一般流程如下:

1.1.1 首先获得共享库中所有目标文件集合，并建立库文件与应用程序,库文件与库文件之间的多对多关系.在此处,我们认为应用程序是确定的。

其中 K1,K2,K3,K4,K5 为共享库文件,P1,P2,P3 为应用程序,箭头表示调用关系。



根据库文件和应用程序,库文件和库文件之间的依赖关系,我们可以建立一个与它们相关的图，图重记录了它们之间的关系,记为 R。其中 K1->K3 表示 K1 库文件调用或包含了库文件 K3,P1->K5 表示应用程序 P1 调用了库文件 K5。

1.1.2 可以建立新库的生成规则,其中 T 为新的共享库。

(1) $\forall P_i \to K_j \in R, K_j \notin T, T = T \cup K_j$

(2) $\forall K_i \in T, K_i \to K_j \in R, K_j \notin T, T = T \cup K_j$

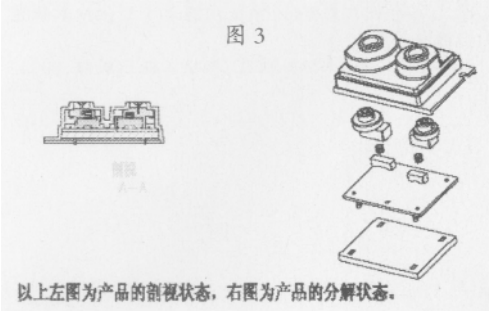
显然,我们很容易得到它们的新共享库 T={K3,K4,K5},那么可以被裁减的就是 K1 和 K2。

但是以上只是做到了文件级的裁减，事实上在很多的库文件里,还保留着大量的冗余的代码。例:库文件 K3 中含有 3 个函数,KP1,KP2,KP3 调用 K3 的应用程序有两个 P1,P2，但是 P1->KP1,P1->KP2,P2->KP2，所以在库文件中 KP3 是永远不会被使用了,要使共享库达到真正意义上的裁减,做到函数级的裁减,就必须删除这部分的冗余代码,而实际中存在着大量这样的代码。可以通过下面的图 1 看出:

1.2 共享库文件函数裁减

1.2.1.使用数据库技术

由于不同的嵌入式系统的性能要求不同，它的共享库一般是不同的,而且可能大小差别很大。由此导致了一个问题,当共享库文件增多的时候，其所包含的函数将相对于文件数量成倍数的增加，如果当我们使用 1.1 中的步骤建立调用关系的时候,那



4 产品试制及老化试验

由于产品工作于高压状态下并且产品使用中的损坏可能造成高压包的损坏。所以在产品设计完成后根据产品设计图投入小批量试产,约 100 只。主要用于可靠性试验。试验项目主要有:耐候性试验、冷热冲击试验、蒸煮试验、耐冲击试验、过电压试验等,一般完成这些试验需要 45 天左右。

只有通过这些试验而保证性能稳定才能进入中批量试用性生产。

么将可能出现一张十分庞大的二维表。所以在此我们考虑使用数据库技术对其进行裁减。使用数据库技术有以下几方面考虑:(1)查找速度快;(2)易于精确定位;(3)可以存储海量数据。

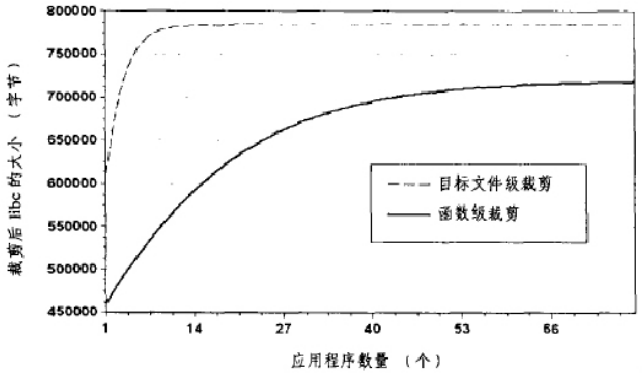


图 1

1.2.2 建立数据表

首先我们先确定应用程序集,记为 F。根据裁减的具体要求,我们建立一个具有四个项的表 Lib(id,Func_name,Call_state,Is_exe,Call_func)。其中主键是 idFunc_name 表示函数名称,Call_state 表示此函数是否被调用 (1 表示调用,0 表示未调用)。Is_exe 表示函数是否是应用程序集 F 中所包含的函数(1 表示属于,0 表示不属于)。Call_func 表示此函数调用其它函数的情况。

1.2.3 添加表项入库

(1)我们可以将 F 的代码段反汇编并对其进行分析。以 x86 体系结构为例,我们可以很容易地找出如下的每个函数块:

```
00000000: (symbol):
0: 55 pushl ebp
1: 89 e5 movl esp, ebp
……
d: e9 leave
e: C3 ret
```

其中 symbol 是该函数块的函数名. 可能是内部符号或外部符号。每个函数块就是这样以符号(symbol)开始。以指令 ret 结束。我们将函数入库,例如(1,symbol,0,1,NULL)。这样我们就确定了应用程序的函数集合,我们记为 F1。

(2)调用函数在汇编中的体现将由 Call 指令体现出来,我们可以根据反汇编代码进行分析。

伪代码如下,symbol 是被调用函数名称,Lib (Func_name)表示 Lib 表中的 Func_name 项。

```
While(Lib(Is_exe)==1)
{
if (Lib(Func_name)!=symbol)
{add_to_database(id,symbol,1,0,NULL);
}
Movenext();// 移动到下一项
}
Delete Lib(Is_exe)==1;// 删除 F1
```

到此,剩下的就是共享库的初始函数库 T。

3)将共享库代码进行反汇编,原理同 1)步。

```
While(所有共享库文件遍历完成)
{
    根据反汇编文件找到此函数调用情况.call_function=(symbol1(), symbol2(), symbol3()……)
if (Lib(Func_name)==symbol)
{set Call_func= call_function;
}
else
{add_to_database(id,symbol,0,0, call_function);
}
    检查下一个函数;
}
    4)根据 Lib 进行最后裁减;
While(i<MAX)
{
取出 id=i 数据项)Lib(i);
If(Lib(i)中 Call_func!=NULL)
{
找到 Lib(Func_name)中与之匹配的项;
Set Lib(Call_state)=1;
}
Movenext();
}
Delete Lib(Call_state)==0;// 删除 F1
```

最后剩下的就是新的共享库了。

2 Linux 的裁减一般步骤:

我们可以根据实际中应用系统设计的具体需要对 Linux 内核源代码进行修改、裁剪、增加模块以及重新编译。其一般步骤为:

2.1 根据实际的需要,首先应该重新编译一个与需求最为接近的 Linux 版本。此处的裁减方法可以采用上面的 4 种基本方法,或者根据实际的情况自己配置内核。比如对于大量的驱动程序在嵌入式系统中是无用的,象硬盘驱动,声音驱动,以太网驱动程序,串/ 并口驱动等,又如某些嵌入式功能比较单一,可能不需要 Linux 中的高级调度等等, 这些都可以根据实际的情况进行选择。

2.2 编写自己的 Boot Loader 引导程序, 将编译好的内核可以加载到内存中。

2.3 重新编写自己的驱动程序和应用程序, 加载到内核中去,需要重新编译内核。

2.4 最后对已经可以正常使用的内核版本做最后的优化,以达到系统设计本身的需要。

3 结束语

内核裁减是一个很复杂的问题,它涉及一个最优化的问题,在实际中还有很多工作和理论需要去完善和实现。

参考文献:
[1]卢延云,孙玉芳.嵌入式 Linux 库裁剪技术分析与改进[j].计算机科学 2004.
[2]金西,黄汪等.Linux 小型化技术[j].计算机工程,2001,27(1).
[3]阳富民,李文海等.嵌入式 linux 系统动态库小型化技术研究.华中科技大学学报(自然科学版).
[4]赵炯.Linux 内核完全注释[M].北京:机械工业出版社, 2004.

作者简介:彭丹 (1985-),男,江西人,贵州大学电信学院 2006 级硕士研究生,主要研究方向:Linux 内核研发。