基于“龙芯”的 VxWorks 系统函数在轨更新研究

史毅龙[1,2]，薛长斌[2]
（1. 中国科学院大学 北京 100190；2. 中国科学院空间科学与应用研究中心 北京 100190）

摘要：由于我国的空间技术的迅速发展，航天嵌入式系统的复杂性急剧增加。在航天领域要求对嵌入式操作系统 vxWorks 进行剪裁工作以适应航天设计的要求，而剪裁掉文件系统的 VxWorks 操作系统存在地面不能对星上的事件进行有效干预的问题。文中在“龙芯”计算机平台上通过对 VxWorks 操作系统进行配置，设计了一种可在轨编程的方案，并针对其中的出现的修改后的函数中全局变量和调用函数的链接地址发生变化的问题提出了解决方案并完成了软件实现该功能。文中 CPU 采用 wh1770，通过修改原被调用函数的初始代码实现对于新函数的调用，并针对全局变量以及调用函数在更新函数中链接地址发生变化设计了在轨更新接口函数和更新代码提取工具，从而实现函数的在轨更新，给出了部分设计流程图以及代码提取工具的测试结果。测试结果显示该工具实现了设计目的，在航天工程领域具备一定的利用价值。

关键词：wh1770；龙芯；VxWorks；函数在轨更新
中图分类号：TN-9 **文献标识码**：A **文章编号**：1674-6236(2015)21-0106-04

VxWorks on-orbit renewal research based on loognson

SHI Yi-long[1,2], XUE Chang-bin[2]
(1. *University of Chinese Academy of Sciences*, *Beijing* 100190, *China*;
2. *Center for Space Science and Applied Research*, *Chinese Academy of Sciences*, *Beijing* 100190, *China*)

Abstract: Due to the booming in the space technology, the complexity of RTOS in aerospace is increasing, It is necessary to tailor embedded operating system to adapt the demand of aerospace design. And there comes the problem that the VxWorks OS which is tailored File System can't tackle the event on the satellite efficiently. By the configuration on the VxWorks OS which is based on the Loongson computer platforms, I design a software which can realize On-orbit programming. And the resolution and a software for the issue that link address of the global variable and called function may change is given. The CPU in the article is wh1770. Based on modifying the first several codes of the old called function and designing the interface function and tools for extracting modified codes ,we can realize the on-orbit function updating. Part of the flow charts and test result are also given. This software can achieve the purpose of the design and has some value in aerospace applications.

Key words: wh1770; loongson; vxworks; on-orbit updating function
DOI:10.14022/j.cnki.dzsjgc.2015.21.032

在航天领域，由于软件设计的复杂程序越来越高，尽管软件在研制过程中已发现和解决了大量软件问题，但难免在轨飞行时仍发现少量软件存在缺陷；同时某些需要载荷根据在轨飞行应用情况调整某些算法。基于以上等方面原因越来越需要软件具有在轨更新能力[1]。一方面可保证航天应用的可靠运行，另一方面可提高软件维修性。同时由于嵌入式操作系统为了符合航天领域的应用需要进行剪裁，本文所采用的基于“龙芯”计算机平台的 vxWorks 操作系统为适应航天领域的需求剪裁掉了文件系统，所以无法使用动态加载模块的方式实现函数的在轨更新，需要使用新的方法来实现在轨编程。同时更新函数中如果有函数调用则其调用的函数和全局变量的链接地址可能会发生变化，需要对函数和全局变量的调用指令做出修改。

结合这些背景，本文提出了针对“龙芯”体系结构的函数在轨更新方案以及设计了可调整调用指令的更新代码提取工具，以满足软件具有在轨更新能力的航天任务要求。

1 函数在轨更新方案简介

文中基于“龙芯”平台的 VxWorks 操作系统由于要符合航天系统操作系统尽量小的要求，剪裁掉了文件系统，所以不能直接使用动态加载模块的方式实现函数在轨更新，本文的在轨更新基于 RAM 进行，系统软件在 RAM 中保留了一片区域，专用来保存在轨更新注入的函数代码。龙芯计算机平台下使用“JAL 指令+函数名”的方式完成函数调用，在链接时，函数名会被链接器替换为函数的地址。JAL 指令在执行时，会将自身地址写入到寄存器 ra 中。通常 JAL 指令之后会跟延迟槽，因此返回地址为原函数调用指令地址+8[2]。

收稿日期：2015-01-26 **稿件编号**：201501221
作者简介：史毅龙(1988—)，男，湖南邵阳人，硕士研究生。研究方向：龙芯嵌入式软件设计。

若将待更新函数的前几条指令进行替换，修改为“JAL 指令+新函数的地址”，这样调用该函数后，替换的 JAL 指令会调用新的函数。从新函数返回后，将回到原函数的代码继续执行。因此需要将原函数第三条指令修改为返回指令，返回到原函数的调用处。

由于 JAL 指令调用新函数时，会当前地址写入寄存器 ra 中，故会将原函数的返回地址（之前寄存器 ra 中的内容）覆盖掉，因此在 JAL 指令之前，须将寄存器 ra 中的内容保存到堆栈中。在从新函数返回后，接着将堆栈中的原函数返回地址写回到 ra 中，并通过返回指令回到原函数调用指令地址+8 处，即回到原函数调用处的下一指令。

2　方案设计及内存分配

2.1　函数替换模块设计

本设计的结构图如图 1 所示，原调用方式为通过 JAL 指令直接跳转到被调函数然后通过 jr ra 返回，替换后的函数调用方式为将通过将原函数的前 8 条指令进行修改，替换为以下形式的指令

```
addiu $sp,$sp,-4
sw $ra,0($sp)
jal.....
nop
lw $ra 0($sp)
jr $ra
nop
addiu $sp,$sp,4
```

从而实现对新函数的调用，图 1 中给出了在轨替换前后的函数调用示例，可发现函数替换对其被调用者而言是不可见的，对其被调用者无任何影响。

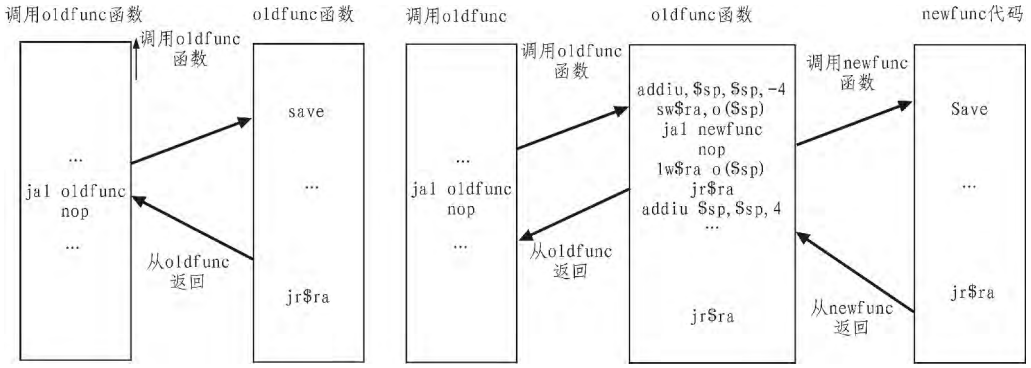

图 1　在轨替换前后函数调用示例

Fig. 1　On-orbit replacement sample before and after the function call

2.2　在轨更新内存分配

为了保存在轨更新的函数代码，系统软件在 RAM 中预留了一片空间，专用来存储函数更新代码，并建立函数更新表对其进行管理，记录原函数地址、更新代码的存储地址、更新函数的长度等信息[3]。

目前系统软件启动完成后的内存分配见图 2，其中 USER_RESERVED 区域为用户保留区，不会被系统软件所使用。该区域的大小在 BSP 的 config.h 文件中定义。

对 BSP 中 RAM 初始化的代码（对于基于 ROM 的和引导 ROM 的 vxworks 映像在 romInit.s 中）进行修改，使初始化时不对 USER_RESERVED 区域清零。在系统首次加电时由应用程序对该区域清零，而在系统复位时则不做清零处理。这样修改是为了在系统复位后，USER_RESERVED 区域中保存的内容不被清除掉。如果每次复位后该区域的内容均被擦除，其中保存的在轨更新代码必须重新注入，在进行了函数在轨更新后，须时刻关注系统状态，在复位发生后立即重新注入。这样会对数据注入链路造成较大的压力。

2.3　更新区域的布局

2.2 中提到在 RAM 中预留了用户保留区域 USERRESERVED 区域来建立函数更新表对函数更新进行管理，该区域内将记录原函数地址、更新代码的存储地址、更新函数的长度等信息。在更新代码注入完成且校验正确后，需检查是否有空间来存储更新代码，如果有则在函数更新表中增加一项，并确定更新代码在更新专用区域中的存放地址，然后将注入的代码从缓冲区拷贝到该地址。

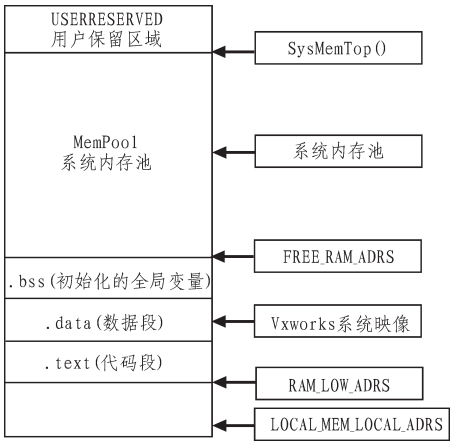

图 2　系统软件启动完成后的内存分配图

Fig. 2　Memory allocation system diagram

为了保证可靠性，函数更新表及更新代码存储区域均采

用三模处理。函数更新表在内部三模。更新代码存储区域则是将同一更新代码存储在 3 个区域,在执行更新替换前进行比对，防止错误。在轨更新区域内存布局如图 3,ONLINE_UPDATE_BEGIN 定义了在轨更新区域首地址 ONLINE_UPDATE_LEN 定义了区域长度,ONLINE_UPDATE_END 则定义了区域末地址,这些常量均需要在在轨更新头文件(本文定义在 onlineUpdate.h)中定义。

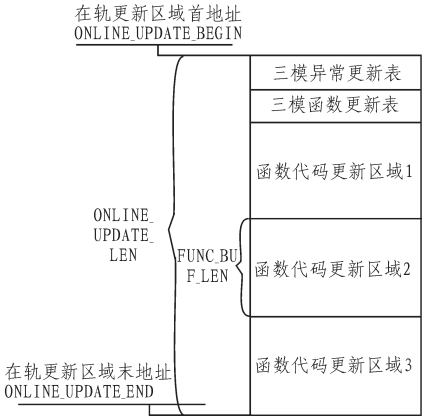

图 3　函数更新表

Fig. 3　Function update figure

3　在轨更新接口函数

为了便于应用程序使用在轨更新功能,制定并实现了在轨更新功能接口,分别实现在轨更新区域初始化,获取下一个在轨更新的函数第一条指令的存储地址,设置函数在轨更新,以及启用和禁用函数更新功能。

4　系统调试代码提取工具及调试结果

4.1　代码提取工具

若待更新的函数中调用了其它函数,或是使用了全局变量,由于重新编译链接后,函数和全局变量的链接地址均可能发生变化,从而导致更新的函数中引用的地址与当前星载计算机上正在运行的软件中地址不一致[4]。在龙芯计算体系结构下,并不需要对全局变量进行处理,只需对函数调用指令进行处理。本文设计的代码提取工具完成更新代码的提取,并将更新代码中的函数调用指令进行调整。

1)全局变量地址变化的处理方式

Gcc-wrs-mips 交叉编译器生成的数据段地址 4 k 字节对齐。更新后的函数体积若相对于原函数增加过大,则会造成数据段链接地址改变[5]。由于龙芯体系结构(基于 mips 体系结构)中存在着全局指针寄存器即 gp 寄存器,使用该指针能方便的存取 static 和 extern 数据，编译器会将全局变量存放在 gp 中的全局指针指向的 32 kB 的地址范围内,这样链接地址的改变并不影响全局变量在 gp 中的指针指向的范围的存放地址,因此调用时并不需要对链接地址改变了的全局变量进行修改,图 4 ,图 5 为测试结果,图 4 为全局变量 gData1 的链接地址，图 5 为 gData1 链接地址变化前后编译器通过 gp 寄存器调用 gData1 的指令对比。

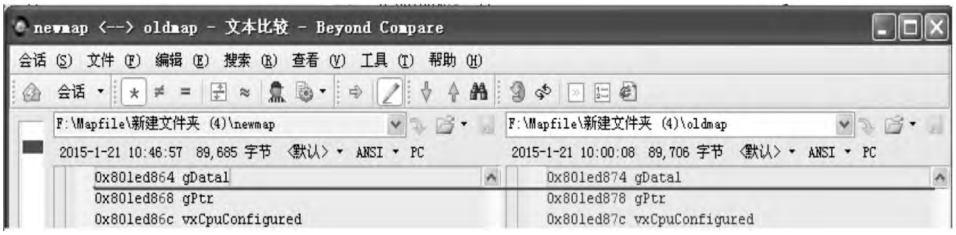

图 4　函数更新后全局变量链接地址变化

Fig. 4　Global variable link address of updated function

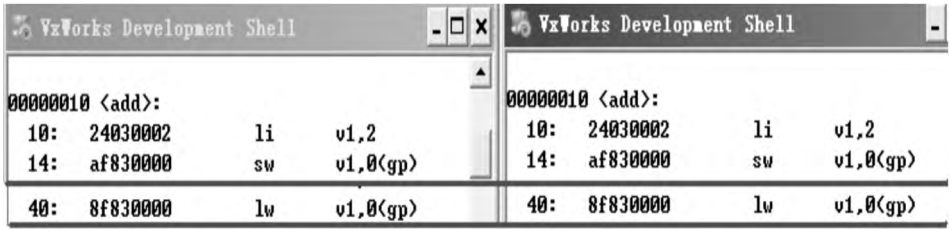

图 5　全局变量链接地址发生变化前后 gp 指令对比

Fig. 5　Comparison of gp instructions before and after link address change

在轨运行的软件中,gData1 的地址分别为 0x801ed874,而在本地编译生成的更新函数代码中,gData1 的地址为 0x801ed864。而此时使用 gp 寄存器调用 gData1 的指令并未发生变化,由此可见,通过利用 gp 寄存器,并不需要对改变了链接地址的全局变量改变做出修改。

2) 函数链接地址的变化

在修改函数代码重新编译后,被修改函数之后的所有函数链接地址均会发生变化，由于修改了函数 add，导致函数 add 和 add 之后的函数链接地址都发生了变化。add 中调用了 retOne 函数,由于龙芯的函数调用指令 JAL 指令是由指令操作符和目标函数所在地址的低 28 位右移 2 位形成的指令偏移值一起构成,如图 6 所示,因此需对 JAL 指令的调整。

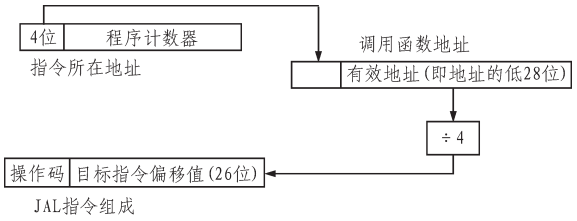

图 6　jal 指令构成

Fig. 6　Jal instruct

通过对更新代码中的 JAL 指令进行解析，通过指令编码中的偏移地址与指令地址高 4 位计算出 JAL 调用的函数地址。查找新的 MAP 文件，得到调用的函数名。然后利用函数名查找旧的 MAP 文件，获取星载计算机中该函数的链接地址[6]。然后对调用函数的链接地址对 JAL 指令进行调整，即完成了 JAL 指令的调整工作。

4.2　代码提取工具执行结果

代码执行结果如图 7 所示。

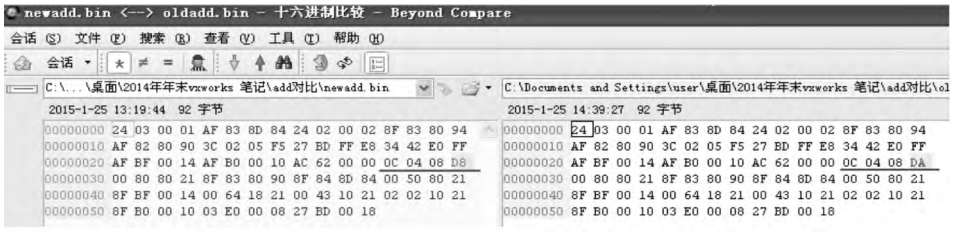

图 7　代码提取工具与原码对比

Fig. 7　Comparison between code from onlineUpdate tool and the old one

5　结束语

本文设计的基于龙芯计算机平台的 vxWorks 函数在轨更新研究可行，设计的代码提取工具能完成任务需求。测试结果显示软件能达到了设计目的，在航天工程领域具备较好的工程利用价值。

参考文献：

[1] 安军社，刘艳秋，孙辉先. VxWorks操作系统板级支持包的设计与实现[J]. 计算机工程，2003，29(7):117–119.
AN Jun-she，LIU Yan-qiu，SUN Hui-xian. The design and implementation of operating system board support package[J]. computer engineering，2003，29(1):117–119.

[2] 张然峰，郝贤鹏，金龙旭，等. 空间相机软件在轨重注方法研究与实现[J]. 光机电信息，2011，28(6):30–34.
ZHANG Ran-feng，HE Xian-peng，JIN Long-xu，et al. On-orbit space camera software research and implementation of heavy injection method[J]. OME Information，2011，28 (6): 30–34.

[3] 乔从连. Vx Works系统的BSP概念及启动过程[J]. 舰船电子对抗，2005，28(1):61–64.
QIAO Cong-lian. VxWorksSystem concept of BSP and concept of starting the process[J]. Ship Electronic Countermeasures，2005，28(1):61–64.

[4] 高进. 卫星在轨编程技术研究[C]. 中国宇航学会计算机应用专业委员会学术交流会论文集，2004.

[5] 朱虹，王海燕. 一种星载软件在轨编程功能的设计和实现技术[J]. 上海航天，2004(1):26–31.
ZHU Hong，WANG Hai-yan. An orbit space-borne software programming function design and implementation of the technology[J] Aerospace Shanghai，2004(1):26–31.

[6] 胡奇明. VxWorks操作系统的重新编译和优化研究与实现[D]. 长沙：国防科学技术大学，2008.

(上接第 105 页)

43–47.

[3] 杨丽莎，王慧. 基于模型的嵌入式软件设计[J]. 计算机应用研究，2004，21(12):76–78.
YANG Li-sha，WANG Hui. Model-based design of embedded software[J]. Application Research of Computers，2004，21 (12):76–78.

[4] 林枫. 基于SCADE的形式化验证技术研究[J]. 测控技术，2011，30(12):71–74.
LIN Feng. Research on SCADE-Based formal verification technology[J]. Measurement and Control Technology，2011，30 (12):71–74.

[5] Jean-Louis Camus，Bernard Dion. Efficient development of airborne software with SCADE suite[M]. Esterel Technologies，2011.

[6] Berry G. The Constructive Semantics of Pure Esterel Draft V3[M]. Esterel Technologies，2009.