第 35 卷 第 4 期 计 算 机 工 程 2009 年 2 月
Vol.35 No.4 Computer Engineering February 2009
·软件技术与数据库· 文章编号：1000—3428(2009)04—0041—03 文献标识码：A 中图分类号：TP311.5

面向 RTEMS 的嵌入式软件集成开发环境

李 曦，张 飞，时 正，吴晓丹
(中国科学技术大学计算机科学技术系，合肥 230027)

摘 要：实现面向 RTEMS 的嵌入式软件集成开发环境，提高嵌入式应用程序的可靠性和开发效率。在编译过程中添加缓冲区溢出动态检测模块，从而在程序运行过程中检测到缓冲区溢出，提高了程序可靠性。利用插桩技术实现远程调试环境，结合能对 RTEMS 进行定制的集成开发环境，提高了嵌入式应用程序的开发和调试效率，缩短了开发周期。

关键词：可靠性；交叉编译；远程调试；嵌入式系统；集成开发环境

RTEMS-oriented Integrated Development Environment for Embedded Software

LI Xi, ZHANG Fei, SHI Zheng, WU Xiao-dan
(Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027)

【Abstract】 In order to improve the reliability and efficiency of embedded application design, this paper realizes an RTEMS-oriented Integrated Development Environment(IDE) for embedded software. A buffer overflow checking module is integrated into the compilation procedure, and it can detect buffer overflow while program running which efficiently improves system reliability. The remote debug module based on gdbstub is integrated to development environment which can be used to customize RTEMS. The development and debugging efficiency of the embedded software are enhanced, and design cycle is shortened.

【Key words】 reliability; cross compiling; remote debugging; embedded system; Integrated Development Environment(IDE)

1 概述

嵌入式系统被越来越广泛地使用，嵌入式应用程序可靠性及其开发效率问题引起了开发人员的关注。通常采用 C 语言开发嵌入式应用程序。C 语言功能强大且代码执行效率高，但由于缺少指针引用边界和数组越界检测机制，因此 C 代码存在缓冲区溢出等可靠性问题。缓冲区溢出会导致系统运行异常甚至崩溃，是影响系统可靠性的重要因素。

嵌入式系统受其自身资源的限制，不具备本地应用程序调试能力，而调试是嵌入式应用开发的必需环节。远程调试在目标机与宿主机之间建立物理连接，宿主机上的调试器与目标机上的调试代理按调试协议共同完成调试工作。软件集成开发环境(Integrated Development Environment, IDE)[1]集成了编辑器、编译器、调试器等模块，负责代码的编辑、编译、调试等工作。在 IDE 上针对嵌入式系统可定制的特点，为不同系统配置提供不同代码模块，能极大提高开发效率，缩短开发周期。

RTEMS(Real Time Executive Multiprocessor Systems)[2]是微内核抢占式实时嵌入式操作系统，具有实时性高、稳定性强、良好的可移植性和可裁剪性等优点，被广泛用于飞行器、防空火箭等系统。本文总结典型缓冲区溢出防范技术，将缓冲区溢出运行时检测模块集成到 GCC 交叉编译器中，增强了 RTEMS 应用程序的可靠性。利用插桩技术搭建基于 GDB 调试器的远程调试环境。基于 RTEMS 模块化设计思路，实现可定制的面向 RTEMS 的嵌入式软件集成开发环境。

2 缓冲区溢出防范技术

指向缓冲区内填充的数据位数超出缓冲区本身的容量，致使溢出的数据覆盖在合法数据上，即缓冲区溢出。根据缓冲区在内存中所处位置的不同，缓冲区溢出可以分为栈溢出、堆溢出、BSS 溢出。禁止向缓冲区填充超出其容量的数据能够有效避免缓冲区溢出。但多数现有 C 程序未在设计阶段进行类似的检查，且由于 C 语言本身缺少指针引用边界和数组越界检测机制，因此存在缓冲区溢出隐患。鉴于此，有必要对 C 程序进行缓冲区溢出检测，尽可能减小其对系统可靠性的影响。可以从不同角度(如编译器、库函数等[3])对典型的溢出检测措施进行分类。

2.1 基于编译器的运行时措施

通过在编译器中集成缓冲区溢出检测机制，可以在不修改源代码的前提下将检测模块插入编译后的二进制文件。若在程序运行时发生溢出，则进入溢出处理模块，根据不同情况进行不同处理。

BoundsCheck, StackGuard, StackShield, Propolice[4]等已有溢出防范技术都是基于编译器的运行时防范措施。BoundsCheck 对每次内存的访问进行边界检测，以避免缓冲区溢出的发生。StackGuard 通过插入“哨兵”并检测“哨兵”的完整性来检测缓冲区溢出是否发生。StackShield 将重要信息备份，在函数返回时用备份信息覆盖原信息以确保应用程

基金项目：教育部高校博士点基金资助项目(20050358040)；安徽省自然科学基金资助项目(070412030)
作者简介：李 曦(1963－)，男，高级工程师、博士，主研方向：嵌入式系统设计；张 飞，硕士研究生；时 正，博士研究生；吴晓丹，硕士研究生
收稿日期：2008-07-13 **E-mail：**llxx@ustc.edu.cn

序的可靠性。Propolice 在 StackGuard 的基础上对局部变量进行重排序，以避免潜在的缓冲区溢出，从而提高系统可靠性。

2.2 基于库函数的运行时边界检查

库函数主要由字符串操作、浮点运算、内存分配和 I/O 操作等函数组成。C 语言中的缓冲区溢出漏洞主要是由大量使用没有提供越界检查的字符串操作函数(gets, strcpy, memset 等)造成的。因此，可以通过修改上述库函数，使其对字符串的操作具有越界检查功能以增强应用程序的可靠性。

Libsafe 是由 Avaya 实验室研制的 C 语言库函数，它针对标准 C 库中容易产生缓冲区溢出漏洞的字符串操作函数进行修改，增加了边界检查机制，有效防止了缓冲区溢出的发生。

2.3 防范技术总结

基于编译器的运行时防范措施不适用于没有源代码、不能重新编译的软件。Libsafe 对库函数的实现仅限于 C 库的一个子集，可以通过设置环境变量来绕开 Libsafe 的防护。Libsafe 不能防范基于堆和 BSS 的缓冲区溢出。

RTEMS 应用程序的开发、调试阶段有宿主机的参与，能提供对程序源代码重新编译的环境，可以很好地克服基于编译器的运行时防范的缺点。综合考虑上述各种防范技术的优缺点和实现成本，RTEMS 应用程序采用基于编译器的运行时防范技术防范缓冲区溢出。

在基于编译器的运行时防范措施中，BoundsCheck 与 StackShield 的开销太大[3]，Propolice 是 StackGuard 的强化版。因此，RTEMS 应用程序采用 Propolice 技术实现对缓冲区溢出的检测。

3 交叉编译技术的实现

交叉编译环境运行在宿主机上，编译并生成能运行在目标机上的二进制文件。交叉编译环境包括 3 个部分：交叉编译器，嵌入式库文件和二进制工具链。交叉编译器实现代码编译功能，是交叉编译环境的核心。库文件提供方便高效的字符串操作、浮点运算和 I/O 函数。二进制工具链由若干二进制工具组成，用于二进制文件分析和生成。

设计开发新的交叉编译环境成本高、效率低，而分析并移植优秀的开源交叉编译环境能提高开发效率并降低开发成本。RTEMS 交叉编译环境的开发采用上述思路[5]。交叉编译环境组件的详细情况如表 1 所示。

表 1 交叉编译环境组件

编译环境组成	名称	来源	版本
编译器	GCC	GNU	3.4.6
库文件	Newlib	Redhat	1.14.0
二进制工具链	Binutils	GNU	2.16.1

GCC(GNU Compiler Collection)是 GNU 编译器集合，它的 C 编译器符合 ANSI C 标准的编译系统，具有效率高、可扩展性好等优点。

Newlib 是由 Redhat 公司为嵌入式系统开发编写的库函数，其特点是轻量级、效率高、移植性好。Binutils 是 GNU 组织提供的二进制工具包，包含汇编编译器、链接器等二进制文件处理工具。

3.1 Propolice 技术

如图 1 所示，Propolice 技术[4]采用 2 种途径，即插入“哨兵”和局部变量重排序来预防缓冲区溢出。

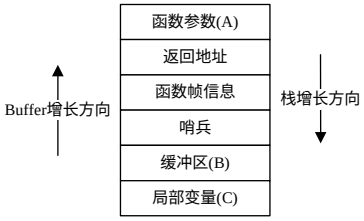

图 1 Propolice 技术

在栈帧的返回地址前插入“哨兵”，当缓冲区溢出并覆盖返回地址时，“哨兵”会被修改。因此，如果“哨兵”的值被修改，则必然发生了缓冲区溢出。该技术最早应用于 StackGuard 技术中，并在 Propolice 中被保留。

局部变量重新排序是对栈帧中的局部变量进行重新排序，使图 1 中各部分数据满足以下要求：

(1)A 中不含有缓冲区或指针变量；

(2)B 中含有缓冲区或包含指针的结构；

(3)C 中不含有缓冲区。

局部变量重新排序是 Propolice 在 StackGuard 技术的基础上进行的改进，排序后的栈结构具有如下特点：(1)函数返回时，当前栈帧外的数据不会被修改；(2)当前栈帧外部的缓冲区不会发生溢出；(3)当前栈帧内部的缓冲区不会发生溢出。

3.2 交叉编译器

深入分析 GCC 是将 Propolice 技术正确集成于编译阶段的前提。GCC 由前端和后端组成，前端负责读取源代码，进行词法、语法分析，生成语法分析树。后端读取语法分析树，转化为 RTL 中间代码，并针对 RTL 代码进行优化，生成汇编代码。

GCC 前端与编程语言相关，后端与目标系统相关。中间代码 RTL 与前端和后端相互独立。此设计模式方便了在 GCC 上扩展前端、扩展后端和针对 RTL 进行定制的优化等工作。

3.3 集成缓冲区溢出检测模块

本文采用 Propolice 作为防范缓冲区溢出的措施，并将其集成于 GCC 编译器中，实现基于编译器的运行时防范技术，提高了系统可靠性。

RTEMS 是一个支持多语言和多处理器架构的嵌入式操作系统。将 Propolice 技术集成于交叉编译环境中时，不能改变 GCC 交叉编译器良好的前端和后端可扩展性，并要保证编译得到的 RTEMS 应用程序能保持其支持多语言和多处理器架构的优势。在 GCC 中集成 Propolice 技术时需要满足上述要求。

由 3.2 节的分析可知，RTL 代码独立于编译器的前端与后端，不同前端和后端都可以共享同样的 RTL 代码生成和优化算法。在 RTL 代码生成和优化阶段集成 Propolice 技术能满足上述要求。

在 RTL 代码生成及优化阶段，函数 tree_rest_of_compilation 读入语法分析结果，生成源代码的原始 RTL 表示，函数 rest_of_compilation 在此原始 RTL 表示之上完成 RTL 优化及汇编代码生成工作。RTL 代码优化包括转移优化、SSA 优化、循环优化、分支预测、寄存器重命名等。RTL 优化之前的初始化工作包括删除注释、调整块顺序等，以实现高效的优化结果。RTL 代码优化后的汇编代码生成阶段负责将优化后的代码转化为指定目标机的汇编代码。为了保持编译器良好的可扩展性和移植性并共享 RTL 代码优化算法，Propolice 技术的实现在 rest_of_compilation 的初始化阶段完

成。集成 Propolice 后的 RTL 代码生成与优化过程见图 2。

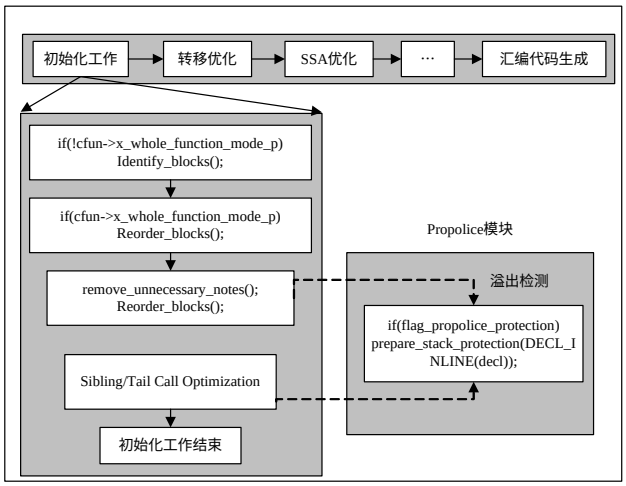

图 2　集成 Propolice 后的 RTL 代码生成与优化过程

在上述工作的基础上，实现交叉编译环境的步骤如下：

(1)配置编译 Binutils，并设置环境变量 PATH；

(2)在 GCC 目录下建立对 Newlib 的链接；

(3)配置编译 GCC。

4　远程调试技术的实现

基于 GDB 的远程调试技术分为 2 种：gdbserver 和 gdbstub。gdbserver 提供应用程序级调试，gdbstub 提供系统级调试。stub 被称为桩，由 GDB 提供，负责处理其所在系统的所有异常，接收并响应宿主机发送的消息。在分析 gdbserver 与 gdbstub 之间区别的基础上[6]，本文采用 gdbstub 技术实现 RTEMS 系统的远程调试。具体步骤如下：

(1)在 i386-stub.c 源代码文件中实现 getDebugChar, putDebugChar, exceptionHandler 这 3 个函数，并实现 memset 函数(由库函数提供)。各个函数声明如表 2 所示。

表 2　函数声明

函数名	返回值	参数	功能
getDebugChar	int	void	从宿主机接收一个字符
putDebugChar	int	int	向宿主机发送一个字符
exceptionHandler	void	int；void *	安装异常变量
memset	void	void *；int；int	设存储区初值

(2)在待调试程序起始部分调用 set_debug_traps()和 breakpoint()函数。

(3)利用第 3 节中提供的交叉编译器将待调试程序和 i386-stub.c 共同编译、链接。

(4)采用串口通信方式连接目标机和宿主机，并将编译好的二进制文件下载到目标机中。

在宿主机上运行 GDB，指定待调试的二进制文件名，并利用 target remote 命令建立宿主机与目标机之间的连接。连接建立后，就能对目标机上的二进制文件进行远程调试。本文实验平台为 PCM-6892 工控板：VIA C3 处理器，主频 1 GHz；SDRAM 内存，容量 128 MB；采用 COM 口作为通信接口。

5　集成开发环境架构设计

RTEMS 系统的集成开发环境采用 QT 应用程序框架，在 KDevelop 开发环境上进行扩展。包括交叉编译器、远程调试器、RTEMS 定制模块、代码编辑器和图形用户界面工具 5 个模块，如图 3 所示。3.3 节实现了集成缓冲区溢出检测模块的 GCC 交叉编译器，第 4 节实现了基于 gdbstub 的远程调试器；RTEMS 定制模块是根据 RTEMS 内核定制需求而实现的前台定制接口，代码编辑器采用 KDevelop 提供的编辑器，图形用户界面为其他 4 个部分提供图形化人机交互界面。

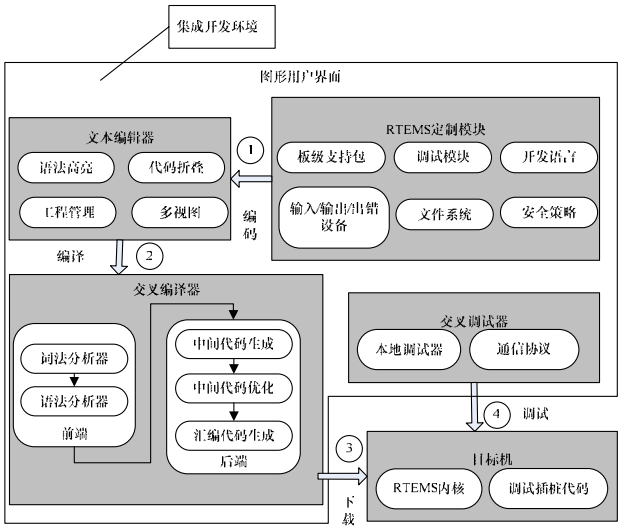

图 3　RTEMS 集成开发环境总体结构

RTEMS 应用程序的开发过程如下：(1)通过 RTEMS 工程初始化向导获取用户需求并对系统参数进行配置，配置选项包括板级支持包、文件系统、可靠性、安全性、系统资源等。(2)在向导生成的代码模板上进行代码编写工作。(3)利用交叉编译环境生成针对目标机的二进制文件并将其下载到目标机中。(4)通过交叉调试器进行代码调试。

6　结束语

本文集成开发环境能方便地针对用户需求设计代码模板，编译具有防范缓冲区溢出功能的嵌入式应用程序，可以实现代码下载与调试功能，极大提高了应用程序开发效率。下一步工作将对本文开发环境进行优化，包括模块之间和模块内部的优化，从而为 RTEMS 应用程序提供更高效的开发平台。

参考文献

[1] 王立泽, 刘　斌. 面向 VxWorks 的嵌入式软件集成开发环境研究[J]. 计算机工程, 2006, 32(3): 55-59.

[2] 程步奇, 尹宝林. 可动态扩展的嵌入式操作系统[J]. 小型微型计算机系统, 2003, 24(2): 216-219.

[3] Cowan C, Wagle P, Beattie S, et al. Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade[C]//Proceedings of DARPA Information Survivability Conference and Exposition. Hilton Head, USA: [s. n.], 2000: 119-129.

[4] Etoh H. GCC Extension for Protecting Applications from Stack-smashing Attacks. (2004-06-15). http://www.trl.ibm.com/projects/security/ssp/.

[5] 邵明超, 杜　强, 李荐民, 等. RTEMS 交叉编译环境的创建[J]. 中国科学院研究生院学报, 2004, 21(2): 254-258.

[6] 李红卫. 嵌入式 Linux 远程调试工具 gdbstub 的剖析与改进[D]. 哈尔滨: 哈尔滨工业大学, 2004.