基于调试代理的远程协同调试模型

胡先浪，张培培
（中船重工集团公司江苏自动化研究所，江苏 连云港 222006）

摘 要：针对嵌入式软硬件资源的限制问题，以及现有的软件开发工具仅支持一对一的调试模式的问题，在嵌入式远程调试原理的基础上，分析了嵌入式交叉调试原理，在此基础上提出了基于调试代理的远程协同调试模型，并详细给出了远程协同调试实现的核心——调试代理功能的实现；其次在国产集成开发环境JARI-IDE和国产嵌入式操作系统JARI-Works上给出了实现方法和功能验证，实现了分布在不同地点的开发人员共享资源，实时协同工作，为模块化分工调试提供了一种新的实现方法。

关键词：交叉调试；调试代理；协同；模型
中图分类号：TP311.5 **文献标识码**：A **文章编号**：1673-629X(2011)05-0029-04

Remote Collaborative Debugging Model Based on Debugging Agent

HU Xian-lang, ZHANG Pei-pei
(Jiangsu Automation Research Institution of CSIC, Lianyungang 222006, China)

Abstract: Aiming at the definition of the embedded software and hardware, analyses the embedded cross debugging theory firstly, because the software research tools support one-to-one debugging mode only, and it puts forward remote collaborative debugging model based on debugging agent, and realizes debugging agent function. The experiment method and the function proof are given on the national integrated developing environment JARI-IDE and the national embedded operating system JARI-Works, it realizes resources share, provides a new method for the debugging modules.

Key words: cross debugging; debugging agent; collaborate; model

0 引 言

调试是嵌入式软件开发中的重要环节之一，调试功能是衡量嵌入式软件开发工具优劣的一个重要指标。随着嵌入式应用迅速发展，嵌入式软件规模和复杂度逐渐增大，软件的规模突破了几万行甚至几十万行、上百万行，一个嵌入式系统或设备中往往集成了多人多家开发的软件，协同开发逐渐成为嵌入式软件开发的重要模式。在该模式下，团队中的多个成员能够同时通过各自的调试器控制运行在一台或多台目标机上的多个软件模块。这种模式的转变也对开发工具提出了新的需求，特别是对团队协同调试能力的需求。

现有的软件开发工具在团队协同开发方面主要强调配置管理，但仅支持一对一的调试模式，也就是同一时刻一台目标机通常只能连接一台宿主机，只能由一个开发人员通过调试器控制目标机的程序，即便是支持系统级、任务级两级调试模式的Tornado，虽然支持搭接不同的任务[1]，但是一台目标机同一时刻仅能支持一个IDE(集成开发环境)搭接一个任务，无法满足团队多对一的协同调试需求。随着调试技术的发展，支持网络连接的调试技术为实现团队协同调试提供了可能。文中在分析嵌入式远程调试原理的基础上，提出一种基于网络调试代理的嵌入式软件远程协同调试模型，支持宿主机和目标机间的多对一团队协同调试模式。

1 嵌入式交叉调试原理

与Windows等桌面操作系统开发方式不同，受限于嵌入式软硬件资源的限制，嵌入式系统的开发通常采用宿主机/目标机的交叉开发模式，软件的编辑、编译、链接、定位等工作在宿主机上进行，通过串口、USB、以太网络或仿真器将目标代码下载或烧录到目标机/板上执行。

嵌入式系统调试时，目标程序的指令代码和数据都在目标机上，通常目标机上还运行一段称作监控(monitor)的控制代码，它通过串口或网络与宿主机通讯，获取调试控制命令(设置断点、单步执行、查看内

收稿日期：2010-10-13；**修回日期**：2011-01-08
基金项目：国防预研基金项目(51315040301)
作者简介：胡先浪(1982-)，男，江苏泗阳人，研究方向为嵌入式调试技术。

存或变量等),依据命令控制目标机上目标代码的执行,并将信息回送到宿主机。利用存放在宿主机上的目标文件中在程序编译链接时生成的调试信息,获取实现目标程序与源程序之间的映射,从而在源码级实现对程序执行情况的控制和观察。调试控制点和数据在源程序与目标程序之间的映射关系是宿主机方的调试器通过分析目标代码的文件格式获得的[2]。如图 1 所示。

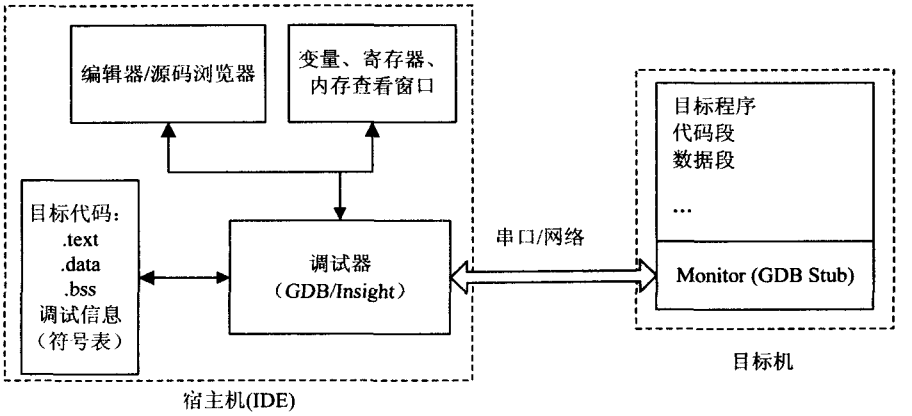


图 1 嵌入式软件交叉调试原理示意图

2 基于调试代理的远程协同调试模型

团队协同开发中,最常见的需求是多个开发者同时调试运行在同一台目标机上的各自开发的软件模块,这些软件模块相互作用共同完成嵌入式设备赋予的功能使命。例如硬件驱动开发厂商、中间厂商和应用厂商的组合。由于这些开发者可能来自不同的厂商,出于商业技术秘密保护的原因,不允许将这些开发者的源代码放在同一台宿主机上编译下载调试,因此需要分别使用各自的宿主机共同配合完成调试过程。这就需要一种支持多对一模式(多个宿主机同时连接一个目标机)的调试工具[3]。

通过对嵌入式交叉调试原理分析可以看出,在传统的集成开发环境中,调试器、编辑器、源码浏览器及目标代码紧密耦合在同一台机器上,监控与固定的调试器通过网络或串口进行通讯。为支持多个宿主机的连接,或者需要目标机监控能与多个宿主机上的调试器通讯,也可把调试器的一部分功能从 IDE 中剥离出来,支持与多个宿主机通讯。前者需要修改目标机监控的逻辑流程,使其可以区分来自于不同调试器的命令,并能回送相应的信息,适用于网络连接方式,对于串口连接的方式不适用[4]。后者则需要把调试器的协议处理模块和网络模块单独提出来,实现一个类似于代理服务器的功能,可同时处理多个宿主机和目标机的命令,目标机监控不需要改动。一般来说,修改目标机监控会有如下问题:

(1)目标机监控为第三方提供,只包含对应的中间件接口或相关协议,无对应源码,无法对其进行修改。

(2)目标机监控为自己开发,但是由于设计等原因,要实现协同功能,需要对其代码进行重新设计和编写。

(3)目标机监控为开源软件,如 GDBServer 等,要实现对它们的修改就必须花费大量时间和精力熟悉其源代码框架体系,再者修改会影响原有代码的健壮性。

(4)修改目标机监控会增加协同功能模块与监控的耦合性。

基于上述原因考虑,文中采用基于调试代理的远程协同调试模型(如图 2 所示)。

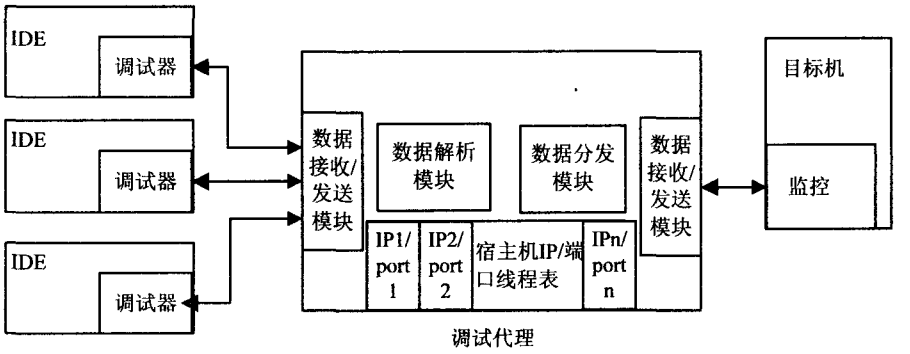


图 2 基于调试代理的远程协同调试模型

在这种方式下,每个开发者在本地 IDE 完成源代码的编辑、编译、链接,本地调试器通过调试代理与目标机监控建立连接。

目标机监控仅与调试代理通讯,从调试代理获取命令,并将处理结果回送给调试代理。调试代理处于枢纽位置,内部的数据接收模块负责接收来自于不同宿主机的命令,数据解析模块负责完成命令包的检查,如协议格式等内容,最后通过数据发送模块转发给目标机监控。调试代理实时接收目标机监控返回的执行结果,通过数据分发模块和内建的 IP/端口线性表实现宿主机的正确分发[5,6]。目标程序与源程序之间的映射仍由每个宿主机中的调试器解析存放在本地的目标文件调试信息完成。

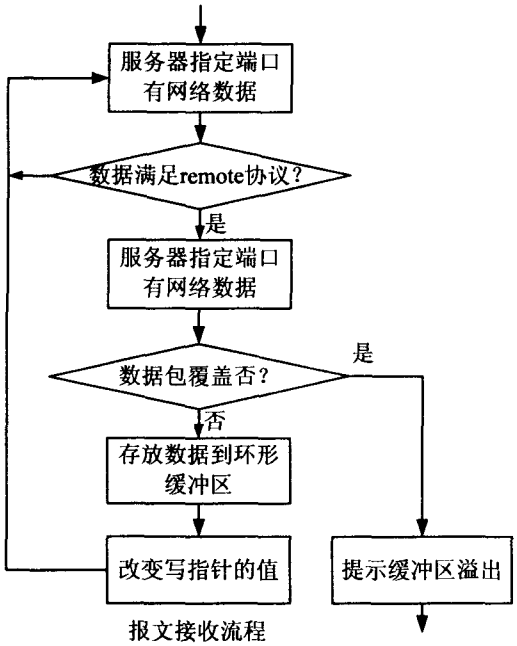
3　远程协同调试实现

远程协同调试实现的核心为调试代理功能的实现,调试代理作为一个网络服务程序既可以运行在一台独立的计算机上,也可与某一个 IDE 共存在一台计算机上,是宿主机与目标机的通讯桥梁,它负责处理所有来自宿主机的命令,并即时回复目标机返回的数据。随着宿主机数量的增加,调试代理的处理负荷成倍增长,数据缓冲区溢出以及目标机数据与宿主机之间的映射等都是需要面临的问题[7]:

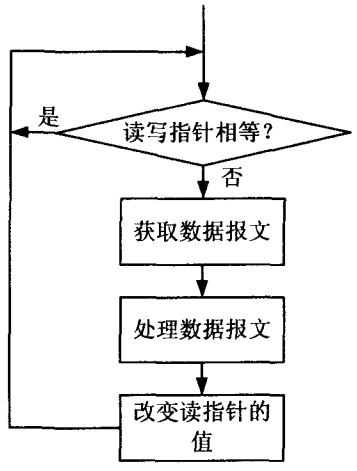
(1)调试代理如何解决缓冲区溢出问题。

(2)调试代理如何将监控回送的信息分发到正确的宿主机上。

对于问题 1,调试代理发起了报文接收和报文处理两个线程。其中报文接收线程采用环形缓冲存取接收到的报文,当接收到一帧数据包时,改变写指针的大小,并根据实际的宿主机数量和报文的流量设置环形缓冲的大小。报文处理线程中根据读写指针的大小判断是否有数据包需要处理,处理结束后改变读指针的大小。由于报文接收线程仅负责把接收到的数据存放到缓冲区,不涉及数据的处理工作,从而保证了宿主机数据的即时接收,报文处理线程实时监测缓冲区并按照既定的协议处理缓冲区数据。数据包处理的速度如果低于接收速度在理论上可以造成数据缓冲区覆盖,通过判断读写指针可以发现这个问题并给与提示。其实现流程如图 3 所示。



报文接收流程



报文处理流程

图 3　调试代理数据处理流程图

对于问题 2,调试代理在内部建立一个宿主机 IP 地址到端口的线性映射表,当接收到连接命令时,调试代理通过查询线性映射表判断是否为一个新的连接,并且为新连接获取一个闲置端口。此后,宿主机通过该端口向调试代理发送命令,调试代理对每一个分配的端口均创建一个报文接收线程和报文处理线程。受制于代理服务器资源和系统线程总数的限制,目前最大支持 32 个宿主机。

当目标机有数据时,调试代理中对应端口的接收线程首先判断数据的正确性,如是否满足既定协议、校验和是否正确等,处理完毕后再通过线性映射表获取对应的宿主机并进行转发。协同调试中的宿主机有独立的端口和独立的读写处理线程,调试代理通过查询线性映射表获取对应的宿主机,保证了数据处理的即时性和目标机数据的正确分发[8~10]。

4　验证及时间性能测试

验证环境中有两个宿主机(简称 h1、h2)和一个目标机(简称 t1)。宿主机上运行国产集成开发环境 JARI-IDE,目标机上运行国产嵌入式操作系统 JARI-Works[11,12]。

如图 4 所示,h1、h2 共享目标机进程列表:

(1)h1 进程号为 40,ip 地址为 172.16.20.133。

(2)h2 进程号为 42,ip 地址为 172.16.20.134。

(3)在验证过程中,h1 早于 h2 占有了信号量,h2 此时只能阻塞等待 h1 释放信号量。

(4)h1 处于停止态,可以执行单步进入等方式释放信号量。h2 处于运行态,不停检测信号量的状态,此时单步功能处于失效状态。

(5)h1 查看变量可以获取共享内存的值,h2 查看变量为无效值。

当 h1 释放信号量后,h2 将由运行态变为停止态,停止位置由断点设置决定(例中为 160 行),此时变量窗口中值一栏变为红色,表示其值发生了变化(从图 4 中的无效地址变为有效地址)。

目标机运行包括三个窗口:JARI-IDE 窗口、正计时窗口和倒计时窗口,鼠标箭头,位图以及公共打印区。

其中,h1 完成窗体初始化,实现画位图,并完成正计时时间窗口的创建。h2 获取窗体共享变量,实现画鼠标箭头,并完成倒计时时间窗口的创建。JARI-IDE 窗口由 h1、h2 共同完成,公共打印区实时显示 h1、h2 的运行状态。二者各有分工

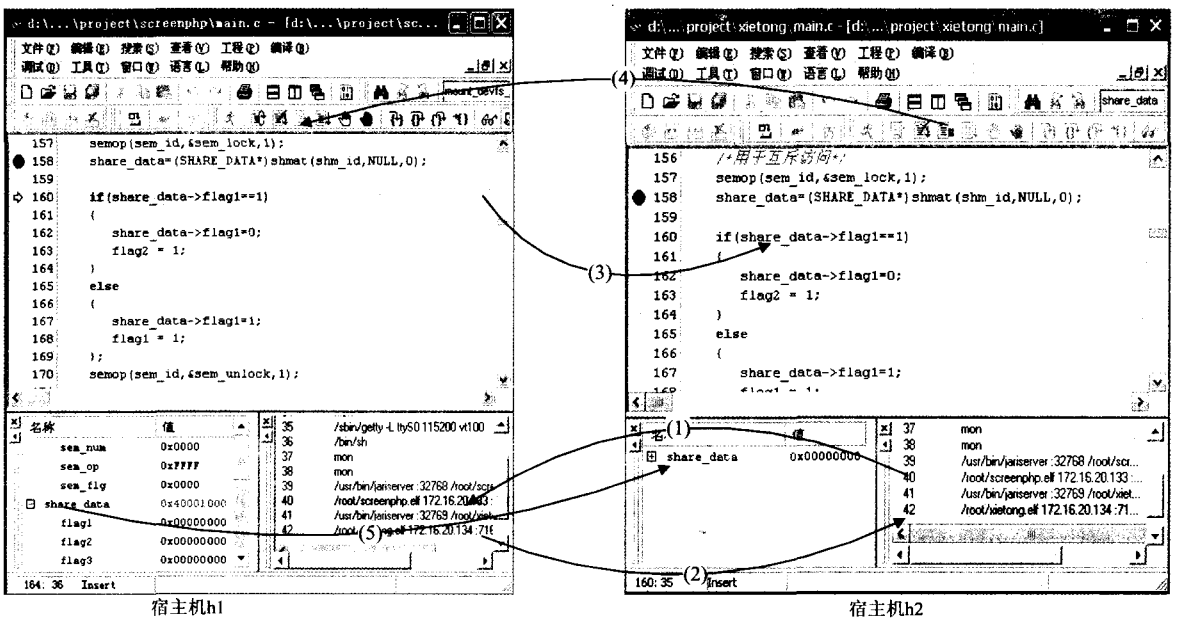


图 4 协同调试执行效果图

又相互协作。

与原来一对一模式比较验证：连接、获取进程、杀除进程。其时间性能对比如表 1 所示。

表 1 时间性能比较结果

调试功能 \ 实现方法	一对一模式	协同调试模式
连接	1.12ms	1.20ms
获取进程	60.34ms	65.15ms
单步进入	47.73ms	47.70ms
单步越过	325.32ms	310.21ms
杀除进程	2.52ms	2.54ms
获取寄存器	2.07ms	2.09ms
获取内存	0.99ms	0.95ms

其中，连接、获取进程和杀除进程采用固定端口与调试代理通讯，其它则为调试代理分配的端口。考虑到单步进入/越过的时间与实际执行的代码有关，如单步越过一个几千行代码的函数和一个只有几行代码的函数，其运行时间没有可比性。

表 1 的测试均在同一个代码中的相同接口中测试。从测试结果可以看到，在协同调试模式下，时间性能没有本质变化。

5 结束语

基于调试代理的协同调试模型改变了传统的一对一的调试模式，满足了团队多对一的协同调试需求，实现了分布在不同地点的开发人员共享资源，实时协同工作，为模块化分工调试提供了一种新的实现方法。当然，该模型仍有需进一步完善的地方，由于网络故障等原因导致单个或多个节点失效后，目标机返回的处理信息因超时机制被丢弃，当故障解除后，无法进行单点故障恢复和多点同步，所有的协同工作由于不同步而可能带来意想不到的错误，这也是下一步要研究的内容。

参考文献：

[1] Wind River. VxWorks Programmer's Guide 5.5[EB/OL]. 2003. http://www.wrs.com.

[2] 乔 容，彭思鹏，柏桂枝. 基于 ELF 的嵌入式软件源码级交叉调试技术[J]. 舰船电子工程，2004(3)：1-7.

[3] Stevens W R. UNIX 网络编程第 2 卷：进程间通信[M]. 北京：清华大学出版社，2000.

[4] Stevens W R，Rago S A. UNIX 环境高级编程[M]. 北京：人民邮电出版社，2006.

[5] 张 磊，王学慧. linux 内核调试技术[J]. 计算机工程，2003(10)：81-83.

[6] 李善平，刘文峰，李程远，等. linux 内核 2.4 版源代码分析大全[M]. 北京：机械工业出版社，2002.

[7] Clarke J. Automated test generation from a behavioral model [R]. Presented at Software QualityWeek. [s. l.]：[s. n.]，1999.

[8] 孙昌爱，靳若明，刘 超，等. 实时嵌入式软件的测试技术[J]. 小型微型计算机系统，2000，21(9)：920-924.

[9] 柳宇霞，陈伏鄂，刘亚斌，等. 某通用测试系统的研究[J]. 微计算机信息，2007(4)：159-161.

[10] 中航一集团计算机软件可靠性管理与测评中心. 通用嵌入式软件测试环境用户手册[M]. 2004.

[11] 周启平，张 杨. VxWorks 下设备驱动程序及 BSP 开发指南[M]. 北京：中国电力出版社，2004：160-168.

[12] 孔祥营，张保山，俞烈彬，等. VxWorks 驱动及分布式编程[M]. 北京：中国电力出版社，2007：32-39.