第 33 卷　第 2 期　　计　算　机　工　程　　2007 年 1 月
Vol.33　No.2　　Computer Engineering　　January 2007
·开发研究与设计技术·　文章编号：1000—3428(2007)02—0277—03　文献标识码：A　中图分类号：N945

交叉开发环境中的目标机代理设计

温利娜，谢　彬，李连云
(华东计算技术研究所，上海 200233)

摘　要：当进行嵌入式应用开发时，需要利用宿主机开发环境中的多种工具，例如交叉编译器、远程调试器、命令解析器和实时资源浏览器。不同的主机端工具需要支持主机-目标机通信，并会产生许多不一致性，使得目标系统的设计变得相当复杂。该文提出了一种实现主机端工具与目标系统交互的通信机制，在自主嵌入式实时操作系统 ReWorks 中，设计开发了一个目标机代理。
关键词：目标机代理；远程调试；交叉开发环境

Design of Target Agent in Cross-development Environment

WEN Lina, XIE Bin, LI Lianyun
(East-China Research Institute of Computer Technology, Shanghai 200233)

【Abstract】 When developing embedded applications, cross-development environment needs to provide various kinds of host tools such as cross compiler, remote debugger, interactive shell, and real-time resource browser etc. The heavy host-target communication and inconsistency caused by tools make target agent more complex. This paper proposes a mechanism which establishes the communication line between several host-resident tools and embedded application, and develop a target agent running in the embedded real-time operating system——ReWorks.
【Key words】 Target agent; Remote debugger; Cross-development environment

交叉开发环境在嵌入式应用开发过程中占有相当重要的地位。当前交叉开发环境主要由文本编辑器、交叉编译器、仿真器、远程调试器、连接器、目标对象查看器、shell 和下载器等工具组成。交叉开发环境需要占有大量资源，通常运行在宿主机上，而开发出的嵌入式应用则运行在嵌入式计算机上，称为目标机。

在嵌入式应用开发前期，程序员开发自己的嵌入式应用，经过交叉编译、链接，可以使用仿真器模拟目标机环境进行初步调试运行。嵌入式硬件系统千差万别，到了嵌入式开发的中后期，需要把嵌入式应用转移到目标机继续进行开发、调试；同时软件开发者需要对目标系统，例如硬件的各种寄存器、内存空间，操作系统的信号量、消息队列、任务和堆栈等等，具有完全的观察、控制和调试能力。而运行在目标机端的嵌入式应用就引入目标机代理，用来管理与主机端工具通信，并为主机端工具提供目标系统信息，或者完成主机端工具发出的命令等。这样的主机-目标机系统就构成一个交叉开发环境。

1　远程交叉开发环境

基于交叉开发环境的系统结构如图 1 所示，这是一个针对自主开发的嵌入式实时操作系统 ReWorks 的集成开发环境，简称 ReDe。

主机端 ReDe 的交叉编译器提供对嵌入式应用的交叉编译，同时链接了应用支持库和操作系统核心代码，并生成可以在目标系统中运行的可执行代码。远程调试器、shell 和目标对象查看器统称为运行时工具，发送请求信息到目标系统并显示返回的结果或信息。仿真器提供模拟的硬件环境仿真，这样开发初期可以在宿主机上运行可执行代码。而工具接口和目标机服务器管理所有主机端工具和在仿真器或目标机中运行的嵌入式应用的通信和信息的交互。

目标机端板级支持包(BSP)提供了对底层硬件的抽象，屏蔽了硬件物理特性的差异。系统核心只提供了最基本的功能，而对硬件设备的驱动还需设备驱动程序。网络库和图形库为嵌入式应用提供网络协议栈、图形应用等功能。

嵌入式应用是由程序员开发的应用程序。目标机代理管理所有主机端工具与目标机的通信和信息交互，目标机代理并不直接访问操作系统，而是通过系统抽象层实现对操作系统的访问。如果目标机代理需要支持调试中断服务程序等功能，其通信层需要 BSP 支持。

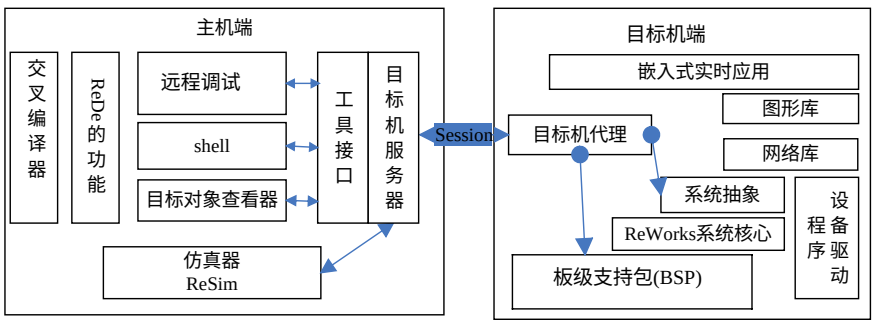

图 1 ReDe/ReWorks 系统结构

2　目标机代理工作原理

代理是可定制的、可连续运行的、自动化的软件模块。代理的最初概念是从人工智能中引入的，大部分智能代理软

作者简介：温利娜(1980－)，女，硕士生，主研方向：嵌入式操作系统及应用；谢　彬，工程师；李连云，硕士生
收稿日期：2006-02-24　**E-mail**：wenlina@ecict.com

件具有学习和自我完善的功能。

本文中的目标机代理以嵌入式操作系统中的任务为实现形式，用于接收和处理主机端工具的请求信息，并返回数据信息。

目标机代理运行在目标机端，初始化时，主机端所有的请求都预先注册到一张表中，每个请求都对应唯一的标识、处理函数等。初始化完成后，目标机代理等待主机端的请求信息；如果获得请求信息，解析出标识号，查表获取处理信息，调用处理函数并返回结果。

目标机代理可以处于系统级，也可以处于任务级。若处于系统级，目标机代理管理自己的内存空间和协议栈；自己获取数据；支持中断服务函数等的调试和相应信息查询；并且调试中遇到断点时，整个操作系统都停止。若处于任务级，目标机代理作为操作系统的一个任务运行，和主机的通信等需要用操作系统提供的函数；只支持对嵌入式应用的调试和信息查询，调试时遇到断点，只有相关的任务停止。本文的目标机代理处于任务级。

基于嵌入式系统开发的特殊性，目标机代理应该有良好的扩充性和可移植性。一个结构良好的目标机代理应该具有以下 2 点：(1)目标机代理支持通信方式的可扩充性。宿主机和目标机的通信方式是由目标系统决定的，目标机平台的多样性导致通信方式的多种方式。(2)目标机代理具有功能可扩充性。在以后的开发中，目标机代理可以添加新的功能。

3 目标机代理的设计

一般情况下，宿主机功能强大，具有较高速度的 CPU、较大的存储空间和功能强大的操作系统。目标机端的 CPU 功能较弱、资源有限，嵌入式操作系统的功能有限。主机端工具的请求尽可能在主机端实现，目标机代理应保留尽可能少的功能。

为了很好地解决目标机代理的可扩充性和可移植性，目标机代理要具有层次性，按功能可划分 4 个部分：通信模块，代理服务模块，功能执行模块和操作系统抽象模块组成。目标机代理的基本结构以及各个模块之间的相互关系如图 2 所示，目标机服务器位于主机端，主机端的运行时工具通过它完成与目标系统的交互。

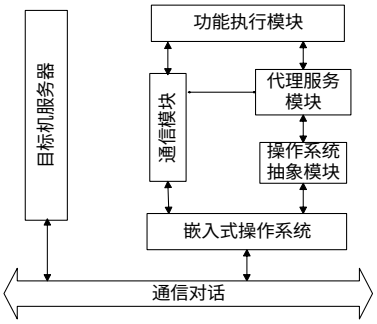

图 2 目标机代理基本结构

(1)通信模块

通信模块的主要功能是接收主机端发来的数据并返回数据，完成对数据头的解析和封装。

它可以采用网络通信或端口通信，按功能可划分 4 个部分：以太网层或端口层，IP 或 SLIP 协议层、UDP 协议层和代理协议层，前 3 层协议是相应通用协议的简单实现，各层的相互关系如图 3 所示。代理协议层输入和输出数据结构如图 4 所示。

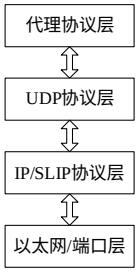

图 3 通信模块功能层次

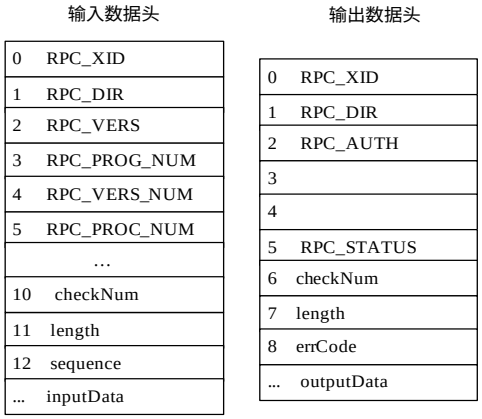

图 4 输入/输出数据头结构

在图 4 中，函数号 RPC_PROC_NUM 对应功能执行模块中“请求执行函数”(即执行主机端的命令的函数)信息的标识号或服务模块注册表子项的服务号，其他遵循标准 RPC 协议规范，但只是部分头标识有效。主要完成数据的网络字节序转换、校验、接收和发送以及函数号的获取。

如果需要添加新的通信方式，只需在 IP 协议层和以太网层定义自己的实现。

通信模块包括两种数据的接收和发送方式：1)轮询方式。隔一段时间读取一次信息；2)中断方式。使用操作系统提供的中断处理函数读取信息。如果目标机代理处于系统模式，只能使用轮询方式直接从网卡驱动中读取数据。如果目标机代理处于任务模式，则可以直接使用操作系统的协议栈等资源，采用中断方式读取和发送数据。本文中目标机代理处于任务模式，因此采用的是中断方式。目标机代理初始化时创建一个任务，管理数据的接收和发送。当有数据到达时，硬件产生中断通知操作系统接收数据，本任务才运行，完成数据的进一步解析。

(2)服务模块

服务模块管理主机端请求的所有实现函数。采用注册机制，即本模块提供一张注册表，注册表是一个固定长度的静态数组，表子项对应数组元素，数组元素结构如下：

```
typedef struct
{   UINT32          svcNum;
    UINT32          (*serviceRtn)();
    int   (*pfnDecode)(void *, void *);
    int   (*pfnEncode)(void *, void *);
} SVC_TAB
```

其中，svcNum 是代理协议层解析获得的函数号，也对应功能执行模块中“请求执行函数”的标识号；serviceRtn 是请求命令的功能处理函数指针；pfnDecode 和 pfnEncode 分别是输入/输出数据的字节序转换函数指针。

该模块提供的函数主要有：1)服务函数的查找。采用顺

序查法，根据服务号自小到大查找。2)服务函数的添加。在注册表中添加子项，即根据服务号自小而大添加服务子项。3)服务函数的分发。它是服务模块的核心函数。主要过程为：根据函数号获得“请求执行函数”对应的子项，转换接收数据字节序，调用“请求执行函数”并获得返回数据，转换返回数据字节序，返回数据到主机端。

(3)功能执行模块

功能执行模块包括多个子模块，主要有连接管理模块、内存模块、寄存器模块、任务执行模块、事件点模块和事件模块等。它不仅需要接收主机端的数据请求信息，完成相应信息的设置和功能执行，并返回信息，而且要实现与操作系统交互的相应功能。

每个子模块都有一个初始化函数，注册自己的“请求执行函数”信息到服务模块的注册表中。

表1显示主要的“请求执行函数”信息以及相应的功能。

表1 命令功能表

功能子模块	服务命令标识	函数的功能
连接管理模块	TGT_PING	测试主机与目标机是否连接
	TGT_CONNECT	连接主机和目标机
	TGT_DISCONNECT	断开连接
寄存器模块	REGS_GET	获取目标机寄存器信息
	REGS_SET	设置目标机寄存器信息
任务执行模块	CTX_STEP	单步执行一个任务
	CTX_CONT	继续执行一个任务
	CTX_STOP	停止一个任务的执行
	CTX _CREATE	创建一个任务
	CTX _DEL	删除一个任务
	CTX _SUSPEND	暂停执行指定任务
	CTX _RESUME	重新执行指定任务
	CTX _STATUS_GET	获得任务运行状态
	FUNC_CALL	创建一个任务并运行指定函数
	DIR_CALL	运行指定函数
事件模块	EVT_GET	获取目标机代理的一个事件
事件点模块	EVTP_ADD	添加一个事件点
	EVTP_DELETE	删除一个事件点
内存管理模块	MEM_READ	读目标机内存
	MEM_WRITE	写目标机内存
	MEM_FILL	以指定格式填写目标机内存
	MEM_SCAN	扫描目标机内存数据格式
	MEM_MOVE	移动目标机内存信息
	MEM_CHECKSUM	校验目标机的指定内存块
	MEM_TXT_UPDATE	在目标机写之后同步主机端信息

功能执行模块提供的“请求执行函数”主要功能如下：

1)寄存器请求：获取或设置目标机上下文的寄存器信息。

2)内存相关：读或写目标机内存以及支持模块信息下载到目标系统而提供内存访问请求，校验目标机内存块以指定格式填写目标机内存，扫描目标机内存数据格式，移动目标机内存信息，在目标机写之后同步主机端信息。

3)任务执行：调试和控制下载到目标机的模块信息。包括继续执行、单步执行、停止执行，创建任务、删除任务、暂停任务、重新执行任务，获取目标机状态信息，调用目标机函数。

4)事件和事件点模块：目标机代理上传事件到主机端，添加和删除异步事件点。

(4)操作系统抽象模块

操作系统抽象模块提供一张映射表，用于访问操作系统函数，屏蔽了操作系统的复杂性，便于目标机代理的移植。主要接口函数有系统信息的获取、内存的分配和释放、加锁和解锁、任务的生成和删除、任务的暂停和重新执行、寄存器信息的设置和获取等。

4 结论

本文所介绍的目标机代理在ReWorks操作系统上获得部分实现。为了测试目标机代理，需要如下过程：在主机端交叉开发环境ReDe中编辑自己的程序，用交叉编译器编译、链接，通过以太网或软盘把二进制文件转移到目标机。

在主机端和目标机端运行的操作系统分别是Windows XP和ReWorks，主机和目标机通过以太网连接，目标机是x86系列32位机器。目标机代理初步实现的功能有建立连接、断开连接、生成一个任务、删除一个任务、中止一个任务的执行、继续执行一个任务和获得任务的状态。

通过使用目标机代理，主机端各个工具不需要额外的时间和带宽与目标系统交互信息，减少连接时带宽的占用。由于目标机代理是独立于目标平台的，需要最少的目标机资源，因此，为嵌入式应用的开发提供了更高的可移植性和更多的功能。

由于本文中的目标机代理的通信依赖于操作系统的协议栈，因此只能处于任务态。在以后的开发中，还会实现处于系统态的目标机代理。

参考文献

1 GNU Organization Stabs[Z]. 2004-10. http://sources.redhat.com/gdb/onlinedocs/stabs.html.

2 GNU Organization Gdb Internals[Z]. 2004-03. http://sources.redhat.com/gdb/onlinedocs/gdbint.html.

3 WindRiver Systems Inc. Tornado API Guide 1.0.1[Z]. Alameda, CA, 2002.

4 Stevens W R. UNIX 网络编程(第2卷)[M]. 北京: 清华大学出版社, 2000-07.

~~~~~~

(上接第263页)

4 小结

利用SIP的呼叫机制和消息格式，本文实现了一个良好扩展性的网络视频监控系统，已运行在社区宽带网络上。第一部分的工作基本完成，下一步将进一步结合社区监控的一些特殊应用给整个系统添加一些模块，如音频呼叫、门禁控制和动态侦测等，从而形成一个可用的更完整的系统。另外我们也在研制前端视频服务器，有望在不久的将来可以支持SIP，从而进一步提高系统的扩展性。

参考文献

1 Rosenberg J, Schulzrinne H. SIP: Session Initiation Protocol[S]. RFC3261, 2002-06.

2 Donovan S. The SIP INFO Method[S]. RFC2976, 2000-10.

3 司端锋, 韩心慧, 龙　勤. SIP 标准中的核心技术与研究进展[J]. 软件学报, 2005, 16(2): 239-250.

4 张智江, 张云勇, 刘韵洁. SIP 协议及其应用[M]. 北京: 电子工业出版社, 2005-01.

5 Moizard A. oSIP2 Document[Z]. 2005. http://www.gnu.org/software/osip/doc/html/index.html.
~~~~~~