嵌入式开发交叉调试技术简介

本文主要介绍嵌入式开发环境中使用gdb交叉调试的大体过程。

在后面例子中给出了针对QtExtend环境中使用交叉调试的最简步骤。

主要内容：

一、嵌入式中的交叉调试

二、交叉调试举例

三、其他

一、嵌入式中的交叉调试

============================

       远程调试环境由宿主机GDB和目标机调试stub共同构成，两者通过串口或 TCP连接，使用GDB标准程串行协议协同工作，实现对目标机上的系统内核和上层应用的监控和调试功能，调试stub是嵌入式系统中的一段代码，作为宿主机GDB和目标机调试程序间的一个媒介而存在。

       就目前而言，嵌入式Linux系统中，主要有三种远程调试方法，分别适用于不同场合的调试工作：用ROM Monitor调试目标机程序、用KGDB调试系统内核和用 gdbserver调试用户空间程序，这三种调试方法的区别主要在于，目标机远程调试stub 的存在形式的不同，而其设计思路和实现方法则是大致相同的。

       而我们最常用的是调试应用程序就是采用gdb+gdbserver的方式进行调试，在很多情况下，用户需要对一个应用程序进行反复调试，特别是复杂的程序采用GDB方法调试，由于嵌入式系统资源有限性，一般不能直接在目标系统上进行调试，通常采用gdb+gdbserver的方式进行调试 gdbserver在目标系统中运行，GDB则在宿主机上运行GDB调试，目标系统必须包括gdbserver程序，宿主机也必须安装GDB程序。一般 Linux发行版中都有一个可以运行的GDB，但开发人员不能直接使用该发行版中的GDB 来做远程调试，而要获取GDB的源代码包，针对arm平台作一个简单配置，重新编译得到相应GDB，GDB的源代码包可以从 http: //ftp.cs.pu.edu.tw/linux/sourceware/gdb/releases/下载，最新版本为gdb-6.4，下载到某个目录，假设用户目录：/home/vicky

下载完后，进入/home/vicky目录，配置编译步骤如下：

#tar jxvf gdb-6.4-tar-bz2

#cd gdb-6.4

#./configure --target=arm-linux --prefix=/usr/local/arm-gdb -v

#make

　　这一步的时候可能会有问题，提示一个函数中（具体函数名不记得了）parse error，就是unsigned前边多了一个”}”你用vi进入那一行把它删掉就行了

#make install

#export PATH=$PATH:/usr/local/arm-gdb

进入gdbserver目录：

#./configure --target=arm-linux --host=arm-linux

#make CC=/usr/local/arm/2.95.3/bin/arm-linux-gcc

(这一步要指定arm-linux-gcc的位置)

　　没有错误的话就在gdbserver目录下生成gdbserver可执行文件，把它烧写到flash的根文件系统分区，或通过nfs mount的方式都可以，只要保证gdbserver能在开发板上运行就行。

　　下面就可以用gdb+gdbserver调试我们开发板上的程序了在目标板上运行gdbserver，其实就是在宿主机的minicom下，我的 RedHat Linux装在vmware下的，我是在minicom下#mount 192.168.2.100:/ /tmp后做的(这里参数 -o nolock可以不加，不加这一步执行得反而更快些)，hello和gdbserver都是位于Linux根目录下，把主机根目录挂在到开发板的 /tmp 目录下。

　　要进行gdb调试，首先要在目标系统上启动gdbserver服务在gdbserver所在目录下输入命令：

(minicom下)

#cd /tmp

#./gdbserver 192.168.2.100:2345 hello

　　192.168.2.100为宿主机IP（这里我没有加这个IP也管用），在目标系统的2345端口开启了一个调试进程，hello为要调试的程序，

　　出现提示：

Process /tmp/hello created: pid=80

Listening on port 2345

(另一个终端下)

#cd /

#export PATH=$PATH:/usr/local/arm-gdb/bin

#arm-linux-gdb hello

(gdb) target remote 192.168.2.223:2345

（192.168.2.223为开发板IP）

　　出现提示：

Remote debugging using 192.168.2.223:2345

[New thread 80]

[Switching to thread 80]

0x40002a90 in ??()

同时在minicom下提示：

Remote debugging from host 192.168.2.100

(gdb)

　　连接成功，这时候就可以输入各种GDB命令如list、run、next、step、break等进行程序调试了。

　　以上针对通过nfs mount和tftp的方式，只能在主机上调试好后下载到开发板上运行，如果有错误要反复这个过程，繁琐不说，有些程序只能在开发板上调试，所以笔者采用了gdbserver的远程调试方式，希望对大家调试程序有用！

二、交叉调试举例

============================

以上资料源于csdn，内容比较全面但是也比较多，下面通过一个针对QtExtended开发环境下的，简单例子，对其过程进行演示。其他开发环境过程也类似。

大致步骤如下：

1)qbuild.pro中添加如下：

CONFIG+=debug

这一步是Qtextended添加调试标记用的，如果你的程序不是qtextended，那么使用"gcc -g"打开调试选项进行编译即可。

2)开发板子中：

#gdbserver :22222 myapp

3)开发主机中：

#arm-linux-gdb myapp

4）最后运行如下即可开始调试：

(gdb)target remote 192.168.100.200:22222

这里192.168.100.200是开发板子ip。

三、其他

============================

本文主要参考：

http://blog.csdn.net/borefo/archive/2009/12/23/5060682.aspx

http://qtextended.org/doc/qtextended4.4/debug-qtextended.html

问题联系:

作者：QuietHeart

Email:quiet\_heart000@126.com

日期：2011年2月24日