嵌入式 Linux 内核调试技术

近年处理器技术发展速度加快，嵌入式领域发生了翻天覆地的变化。特别是网络的普及，消费电子异军突起，嵌入式与互联网成为最热门的技术。在所有操作系统中，Linux 是发展很快、应用很广泛的一种操作系统。Linux 的开放性以及其他优秀特性使其成为嵌入式系统开发的首选。

嵌入式系统开发所面临的问题

嵌入式软件开发有别于桌面软件系统开发的一个显着的特点是，一般需要一个交叉编译和调试环境，即编辑和编译软件在主机上进行，编译好的软件需要下载到目标机上运行 ，主机和目标机之间建立起通讯连接，并传输调试命令和数据。由于主机和目标机往往运行着不同的操作系统，而且处理器的体系结构也彼此不同，这就提高了嵌入式开发的复杂性。

总的来说，嵌入式开发所面临的问题主要表现在以下几个方面。

涉及多种 CPU 及多种 OS

嵌入式的 CPU 或处理器包括 MIPS、PPC、ARM，XScale 等不同的架构，这些处理器上运行的操作系统也有 VxWorks、Linux、μC/OS、WinCE 等多种。在一个企业之内，可能会同时使用好几种处理器，甚至几种嵌入式操作系统。如果需要同时调试多种类型的电路板，那复杂性是可想而知的。这也是我们选用瑞士 Abatron 公司的 BDI2000的原因之一，它是一款功能强大的 JTAG/BDM 通用仿真器。它支持：PPC/MIPS/ARM/XSCALE/ CPU12/CPU32/M-CORE/ColdFire 等多种处理器，支持 Windows/Linux 系统平台，以及多种第三方调试器，并且对 Flash 的烧写也很简单方便。

开发工具种类繁多

通常各种操作系统有各自的开发工具，在同一系统下开发的不同阶段也会应用不同的开发工具。如在用户的目标板开发初期，需要硬件仿真器来调试硬件系统和基本的引导程序，然后进行操作系统及驱动程序的开发调试。在调试应用程序阶段可以使用交互式的开发环境进行软件调试，在测试阶段需要一些专门的测试工具软件进行功能和性能的测试。在生产阶段需要固化程序及出厂检测等等。BDI2000可以适应开发的各个阶段，节约企业的支出和简化管理难度。

对目标系统的观察和控制

由于嵌入式硬件系统千差万别，软件模块和系统资源也多种多样，要使系统能正常工作，软件开发者必须要对目标系统具有完全的观察和控制能力，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、堆栈等。

此外，嵌入式系统变化更新比较快，对开发时间要求比较紧，需要一套功能强大的嵌入式软件集成开发工具，用于嵌入式软件开发的各个阶段。美国 Ultimate Solution 公司的 LinuxScope-JTD 调试器是一个很好的选择，它专门用于配合 BDI2000仿真器，特点是基于 Eclipse 的集成开发环境和插件技术；提供脚本定制功能实现目标机的特殊操作；提供 Linux 内核调试功能，容易修改和观察硬件寄存器；增强的 MI 接口可识别硬件断点及模块跟踪；具有标准调试特性；支持 Linux 应用程序的开发；以及 Windows/Linux 系统平台。

嵌入式 Linux 内核的调试

编译内核

本文所调试的是 MontaVista Linux，硬件是 Intel Xscale PXA250。它拥有超过两千多用户和众多的 MontaVista Linux 产品在市场上销售，覆盖从智能手机、高清电视、机器人、无线网络设备到3G 电信服务器等各种嵌入式应用。MontaVista Linux 本身就是用 BDI2000来开发调试的。

进入内核源码目录下，即可配置完成相关选项并准备编译。配置时不要选中 KGDB(软件基内核调试)，否则会和 BDI2000冲突。为了调试内核，需要在编译时加入调试信息，否则将无法看到源代码。修改内核源码根目录下的 Makefile，在 CFLAGS 宏定义的末尾添加“–ggdb”选项，保存退出。这里加上 gdb 是为了更好的使代码适合 gdb 调试器，另外注意 Makefile 中的优化设置，有些时候优化会调整代码执行的顺序，在内核的调试阶段，不要加入优化选项。

内核调试

首先，配置 BDI2000，确保目标机的正常初始化。通常来说，到了调试内核的阶段，电路板的 boot 程序应该是正常的，可以利用 boot 来完成目标机的初始化；另一种方式是通过 BDI 的配置文件来完成。

接下来就是下载代码进行调试。如果代码已经固化，那仅下载调试信息给仿真器即可；否则需要把代码下载到 RAM 里运行，同时下载调试信息给仿真器。本文所用的是后一种方式。

由于 Linux 运行之后会启动 MMU 而使地址重映射，因此第一个断点通常在函数 start_kernel( )，而且只能设置为硬断点。硬件断点是非常有限的，有的处理器甚至只能设置一个。所以，在调试 Linux 内核时，使用普通的 GDB 进行断点设置会非常不方便。LinuxScope 可以很方便的切换断点模式，并支持软断点，使断点的设置不再受到限制，为调试 Linux 内核提供强有力的支持。具体步骤如下：

1) BDI 配置文件的断点模式：soft

2) LinuxScope 配置默认的断点模式：soft /hard 都可以

3) 用 BDI 下载压缩的内核：load 0x20000 zImage bin

4) 把 PC 指针指到内核入口地址：ti 0x30000

5) 运行 LinuxScope，在 start_kernel 处设置硬件断点

6) go，停下来后再设置软断点即可

模块内核的调试

我们使用 BDI2000来调试 Linux 内核的另外一个重要原因，就是它可以支持调试内核模块。内核模块是一些可以让操作系统内核在需要时载入和执行的代码，这意味着它可以在不需要时由操作系统卸载。这种方式可以扩展操作系统内核的功能，而不需要重新启动系统，这一点对

调试驱动程序的工程师特别有用。因为如果驱动程序编译进内核的话，会增加内核的大小，还要改动内核的源文件，而且不能动态的卸载，不利于调试，所以推荐使用模块方式。

调试 Linux 2.4内核模块

Linux 2.4内核模块的调试比较简单，使用命令“insmod -m”来加载模块。参数“-m”非常重要，它的功能是在把模块加载到内存时产生一个加载 map 表。然后通过 LinuxScope 调试器加载相应的调试信息。例如：

[root@lisl tmp]# insmod -m hello.o >modaddr

查看模块的加载信息文件 modaddr 如下：

.this 00000060 c88d8000 2**2

.text 00000035 c88d8060 2**2

.rodata 00000069 c88d80a0 2**5

……

.data 00000000 c88d833c 2**2

.bss 00000000 c88d833c 2**2

……

在这些信息中，我们用到的只有.text、.rodata、.data、.bss。当然，把相关的信息输入 LinuxScope 调试器，它会把以上地址信息加入到 gdb 中进行模块功能的调试。

这里需要注意的是对模块进行编译时，也需要增加“-g”选项。

另外，这种方法也存在一定的不足，它不能调试模块初始化的代码，因为此时模块初始化代码已经执行过了。如果初始化部分有问题，那么将无法进行调试。遇到这样的情况可以修改代码，延迟初始化部分的执行。另外，也可以采用以下替代方法：当插入内核模块时，内核模块机制将调用函数 sys_init_module (kernel/modle.c)执行对内核模块的初始化。程序代码片断如下：

……

if (mod->init != NULL)

ret = mod->init();

……

在该语句上设置断点，也能在执行模块初始化之前停下来。

调试 Linux 2.6内核模块

在 Linux 2.6内核系统中，由于 module-init-tools 工具的更改，insmod 命令不再支持-m 参数，只有采取其他的方法来获取模块加载到内核的地址。

比较简单的方式是修改内核配置文件，使系统支持 CONFIG_KALLSYMS,这样就可以把相关的符号信息放到目录/sys 下，然后通过 LinuxScope 调试器加载相应的调试信息。通过在模块初始化函数中放置一下代码，也可以获得模块加载到内存中的地址，只是这样要麻烦一些。

应用程序的调试

到了应用程序调试的阶段，仿真器就可以“功成身退”了，剩下的调试任务就由 LinuxScope 调试器来独自完成。此时只要在目标系统中启动“gdbserver”，调试应用程序非常的方便。

结语

面向行业、应用和设备的嵌入式 Linux 工具软件和嵌入式 Linux 操作系统平台是未来发展的必然趋势。跟踪 Linux 的发展，符合标准，遵循开放是大势所趋，嵌入式 Linux 也不例外。Linux 调试技术的进步为 Linux 在嵌入式领域的应用广泛性提供了保证。本文所讲述的仿真器技术和调试器技术可以极大的提高开发者的效率。

近年处理器技术发展速度加快，嵌入式领域发生了翻天覆地的变化。特别是网络的普及，消费电子异军突起，嵌入式与互联网成为最热门的技术。在所有操作系统中，Linux 是发展很快、应用很广泛的一种操作系统。Linux 的开放性以及其他优秀特性使其成为嵌入式系统开发的首选。

嵌入式系统开发所面临的问题

嵌入式软件开发有别于桌面软件系统开发的一个显着的特点是，一般需要一个交叉编译和调试环境，即编辑和编译软件在主机上进行，编译好的软件需要下载到目标机上运行 ，主机和目标机之间建立起通讯连接，并传输调试命令和数据。由于主机和目标机往往运行着不同的操作系统，而且处理器的体系结构也彼此不同，这就提高了嵌入式开发的复杂性。

总的来说，嵌入式开发所面临的问题主要表现在以下几个方面。

涉及多种 CPU 及多种 OS

嵌入式的 CPU 或处理器包括 MIPS、PPC、ARM，XScale 等不同的架构，这些处理器上运行的操作系统也有 VxWorks、Linux、μC/OS、WinCE 等多种。在一个企业之内，可能会同时使用好几种处理器，甚至几种嵌入式操作系统。如果需要同时调试多种类型的电路板，那复杂性是可想而知的。这也是我们选用瑞士 Abatron 公司的 BDI2000的原因之一，它是一款功能强大的 JTAG/BDM 通用仿真器。它支持：PPC/MIPS/ARM/XSCALE/ CPU12/CPU32/M-CORE/ColdFire 等多种处理器，支持 Windows/Linux 系统平台，以及多种第三方调试器，并且对 Flash 的烧写也很简单方便。

开发工具种类繁多

通常各种操作系统有各自的开发工具，在同一系统下开发的不同阶段也会应用不同的开发工具。如在用户的目标板开发初期，需要硬件仿真器来调试硬件系统和基本的引导程序，然后进行操作系统及驱动程序的开发调试。在调试应用程序阶段可以使用交互式的开发环境进行软件调试，在测试阶段需要一些专门的测试工具软件进行功能和性能的测试。在生产阶段需要固化程序及出厂检测等等。BDI2000可以适应开发的各个阶段，节约企业的支出和简化管理难度。

对目标系统的观察和控制

由于嵌入式硬件系统千差万别，软件模块和系统资源也多种多样，要使系统能正常工作，软件开发者必须要对目标系统具有完全的观察和控制能力，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、堆栈等。

此外，嵌入式系统变化更新比较快，对开发时间要求比较紧，需要一套功能强大的嵌入式软件集成开发工具，用于嵌入式软件开发的各个阶段。美国 Ultimate Solution 公司的 LinuxScope-JTD 调试器是一个很好的选择，它专门用于配合 BDI2000仿真器，特点是基于 Eclipse 的集成开发环境和插件技术；提供脚本定制功能实现目标机的特殊操作；提供 Linux 内核调试功能，容易修改和观察硬件寄存器；增强的 MI 接口可识别硬件断点及模块跟踪；具有标准调试特性；支持 Linux 应用程序的开发；以及 Windows/Linux 系统平台。

嵌入式 Linux 内核的调试

编译内核

本文所调试的是 MontaVista Linux，硬件是 Intel Xscale PXA250。它拥有超过两千多用户和众多的 MontaVista Linux 产品在市场上销售，覆盖从智能手机、高清电视、机器人、无线网络设备到3G 电信服务器等各种嵌入式应用。MontaVista Linux 本身就是用 BDI2000来开发调试的。

进入内核源码目录下，即可配置完成相关选项并准备编译。配置时不要选中 KGDB(软件基内核调试)，否则会和 BDI2000冲突。为了调试内核，需要在编译时加入调试信息，否则将无法看到源代码。修改内核源码根目录下的 Makefile，在 CFLAGS 宏定义的末尾添加“–ggdb”选项，保存退出。这里加上 gdb 是为了更好的使代码适合 gdb 调试器，另外注意 Makefile 中的优化设置，有些时候优化会调整代码执行的顺序，在内核的调试阶段，不要加入优化选项。

内核调试

首先，配置 BDI2000，确保目标机的正常初始化。通常来说，到了调试内核的阶段，电路板的 boot 程序应该是正常的，可以利用 boot 来完成目标机的初始化；另一种方式是通过 BDI 的配置文件来完成。

接下来就是下载代码进行调试。如果代码已经固化，那仅下载调试信息给仿真器即可；否则需要把代码下载到 RAM 里运行，同时下载调试信息给仿真器。本文所用的是后一种方式。

由于 Linux 运行之后会启动 MMU 而使地址重映射，因此第一个断点通常在函数 start_kernel( )，而且只能设置为硬断点。硬件断点是非常有限的，有的处理器甚至只能设置一个。所以，在调试 Linux 内核时，使用普通的 GDB 进行断点设置会非常不方便。LinuxScope 可以很

方便的切换断点模式，并支持软断点，使断点的设置不再受到限制，为调试 Linux 内核提供强有力的支持。具体步骤如下：

1) BDI 配置文件的断点模式：soft

2) LinuxScope 配置默认的断点模式：soft /hard 都可以

3) 用 BDI 下载压缩的内核：load 0x20000 zImage bin

4) 把 PC 指针指到内核入口地址：ti 0x30000

5) 运行 LinuxScope，在 start_kernel 处设置硬件断点

6) go，停下来后再设置软断点即可

模块内核的调试

我们使用 BDI2000来调试 Linux 内核的另外一个重要原因，就是它可以支持调试内核模块。内核模块是一些可以让操作系统内核在需要时载入和执行的代码，这意味着它可以在不需要时由操作系统卸载。这种方式可以扩展操作系统内核的功能，而不需要重新启动系统，这一点对调试驱动程序的工程师特别有用。因为如果驱动程序编译进内核的话，会增加内核的大小，还要改动内核的源文件，而且不能动态的卸载，不利于调试，所以推荐使用模块方式。

调试 Linux 2.4内核模块

Linux 2.4内核模块的调试比较简单，使用命令“insmod -m”来加载模块。参数“-m”非常重要，它的功能是在把模块加载到内存时产生一个加载 map 表。然后通过 LinuxScope 调试器加载相应的调试信息。例如：

[root@lisl tmp]# insmod -m hello.o >modaddr

查看模块的加载信息文件 modaddr 如下：

.this 00000060 c88d8000 2**2

.text 00000035 c88d8060 2**2

.rodata 00000069 c88d80a0 2**5

……

.data 00000000 c88d833c 2**2

.bss 00000000 c88d833c 2**2

……

在这些信息中，我们用到的只有.text、.rodata、.data、.bss。当然，把相关的信息输入 LinuxScope 调试器，它会把以上地址信息加入到 gdb 中进行模块功能的调试。

这里需要注意的是对模块进行编译时，也需要增加“-g”选项。

另外，这种方法也存在一定的不足，它不能调试模块初始化的代码，因为此时模块初始化代码已经执行过了。如果初始化部分有问题，那么将无法进行调试。遇到这样的情况可以修改代码，延迟初始化部分的执行。另外，也可以采用以下替代方法：当插入内核模块时，内核模块机制将调用函数 sys_init_module (kernel/modle.c)执行对内核模块的初始化。程序代码片断如下：

```
……

if (mod->init != NULL)

ret = mod->init();

……
```

在该语句上设置断点，也能在执行模块初始化之前停下来。

调试 Linux 2.6内核模块

在 Linux 2.6内核系统中，由于 module-init-tools 工具的更改，insmod 命令不再支持-m 参数，只有采取其他的方法来获取模块加载到内核的地址。

比较简单的方式是修改内核配置文件，使系统支持 CONFIG_KALLSYMS,这样就可以把相关的符号信息放到目录/sys 下，然后通过 LinuxScope 调试器加载相应的调试信息。通过在模块初始化函数中放置一下代码，也可以获得模块加载到内存中的地址，只是这样要麻烦一些。

应用程序的调试

到了应用程序调试的阶段，仿真器就可以“功成身退”了，剩下的调试任务就由 LinuxScope 调试器来独自完成。此时只要在目标系统中启动“gdbserver”，调试应用程序非常的方便。

结语

面向行业、应用和设备的嵌入式 Linux 工具软件和嵌入式 Linux 操作系统平台是未来发展的必然趋势。跟踪 Linux 的发展，符合标准，遵循开放是大势所趋，嵌入式 Linux 也不例外。Linux 调试技术的进步为 Linux 在嵌入式领域的应用广泛性提供了保证。本文所讲述的仿真器技术和调试器技术可以极大的提高开发者的效率。