《VXWORKS 内核、设备驱动与 BSP 开发详解》

作者：华清远见

第 2 章 编译工程——Builder

本章简介

本章将介绍 Tornado 所带的编译器和编译的过程。任何高级语言代码都需要经过编译、链接，才能成为 CPU 可以执行的机器码。对于不同机器架构来说，CPU 所能处理的指令集、寻址空间等都存在着不同之处。所以，在使用编译器的过程中，需要注意选择不同的编译器完成编译。Tornado 在安装之后就带有默认的编译器，这一点为开发者提供了方便，但同时也限制了开发者的选择。

相对于 Tornado 2.0，Tornado 2.2 并未在编译器上限制用户，用户可以根据需要选择不同或经过优化的编译器来完成编译任务。

2.1 编译工程及 Boot Rom

首先看看简单的编译过程。本节中，还会介绍到引导目标机的必要步骤——制作引导盘。

2.1.1 编译工程

使用 Tornado 集成开发环境可以轻松地编译整个工程：既可以在菜单中选择（如图 2.1 所示），也可以在工程管理器中选择（如图 2.2 所示）。

图 2.1 所示为 Tornado 环境的编译菜单，其中提供了创建、重新创建全部、编译 3 项主要功能和创建 BootRom、依赖关系分析、停止编译 3 项附加功能。

创建功能是指使用编译器、链接器将整个工程编译并链接形成最终的目标文件。在编译的过程中，如果有文件已经存在编译后的目标代码且该文件没有发生改动，编译器将忽略该文件的编译，直接将其二进制代码链接到目标文件。在工程的开发和调试阶段，往往选择创建功能而不是重新创建功能，以避免重复编译，节约编译时间。

重新创建全部的功能类似于创建功能，也是对整个工程进行编译和链接，从而形成最终的目标文件。不同的是，重新创建全部目标文件的第一步是将所有目标文件都清除，而后才开始创建。这样做将可以避免因为没有检测到文件的改动而发生错误。重新创建选项多用在工程的最后发行阶段。

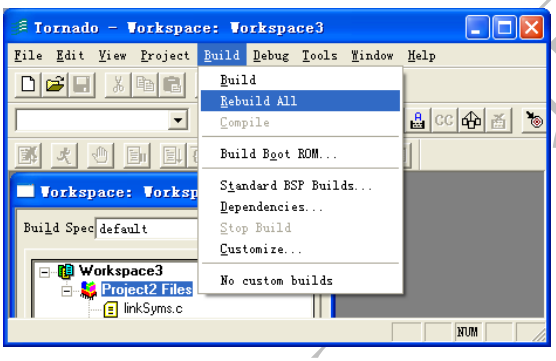

图 2.1 通过“Build”菜单编译工程

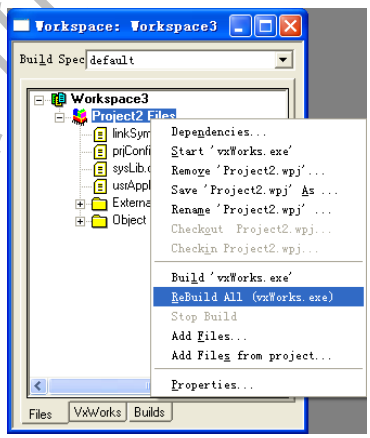

图 2.2 通过工程管理器编译工程

编译功能仅对指定的单个文件执行编译操作，既不做清空目标文件的动作，也不进行目标文件的链接。然而，正是因为少了这两个动作，才使得这项功能在创建单个目标文件时速度很快。编译所生成的目标文件是工程的中间目标文件，如果对应的文件已经存在，则将其覆盖。

在辅助功能中，创建 BootRom 的功能将在之后介绍，停止编译也可以通过字面的意思理解，这里要突出的功能是依赖关系分析。

大多数 C 语言文件都会包含头文件，依赖关系分析的作用就是将 C 语言文件自动包含在工程中——既可方便开发者修改头文件中的定义，又可避免文件发生混淆。经过依赖分析后，所有的头文件都会被放置在工程管理器的扩展依赖文件（External Dependencies）分支下。

选择编译或创建命令之后，Tornado 开发环境会弹出“创建输出（Build Output）”窗口（如图 2.3 所示），其中会显示创建过程中输出的信息。该窗口是一个只读文本窗口，无法进行编辑，却可以选择。

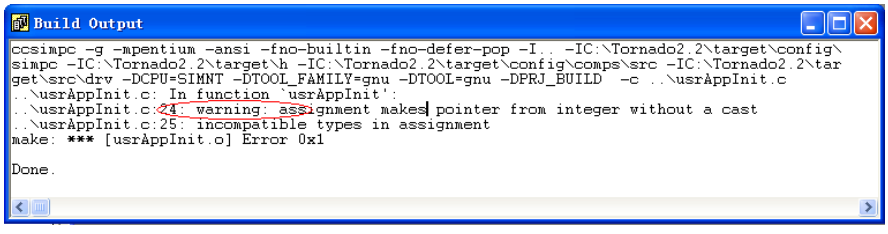

图 2.3　编译输出窗口

编译过程中输出的信息通常会很多，大部分信息是普通用户不需要关心的。一般来说，用户会关心两方面的内容：发生了哪些错误，以及出现了哪些警告。第一种情况发生时，系统会自动停止编译并报告错误，错误信息较为明显，用户可以通过鼠标双击错误信息跳转到发生错误的代码行。而第二种情况发生时，系统不会中止正常的编译，此时用户可以使用查找关键词“warning”的办法来找到提出警告的位置（见图 2.3）。

创建的过程除了生成每一个文件所对应的目标文件外，对于可引导工程来说，还会生成带有操作系统 VxWorks 和用户应用程序的映像，这个映像是可以直接下载到目标机并运行的。而对于可加载工程来说，这个文件由工程中的每一个二进制文件链接后生成。当然，根据编译规则的不同，生成的最终目标文件也有些细微的差异，这一点将在第 4 章提及。

2.1.2　编译 Boot Rom

除了创建工程目标代码之外，VxWorks 还需要创建一种叫作 BootRom 的目标代码。在 Tornado 2.0 时代，菜单中没有编译 BootRom 的选项，用户必须在“命令提示符”环境下手动编译 BootRom。在 Tornado 2.2 中，WindRiver 公司对此进行了改进，用户只需点点鼠标就可以编译生成 BootRom。然而，Tornado 的改造并不彻底，在后面的实例练习中将会发现，在有些步骤中还是不得不进入“命令提示符”环境。

单击编译菜单的“Build Boot Rom”一项将弹出创建 BootRom 的窗口，该窗口中有 3 个可选内容，分别是选择 BSP 包、选择所需创建的映像和选择编译工具，如图 2.4 所示。

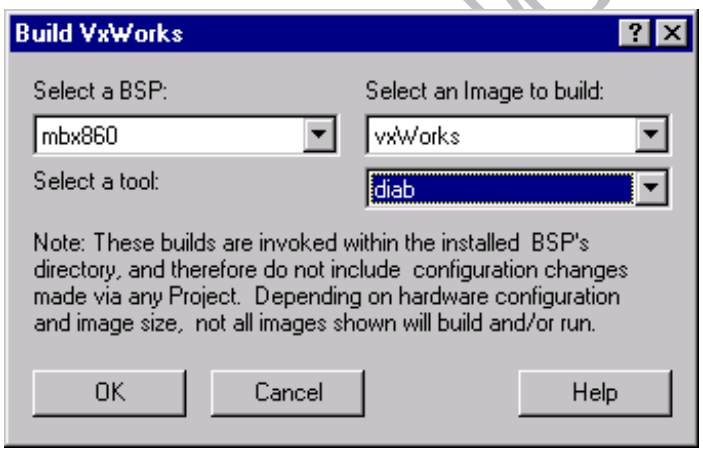

图 2.4　编译 Bootrom

“选择 BSP 包”用于选择需要编译的 BSP 包，其下拉框中的选项来自于 Tornado 的安装目录。从第 1 章中的目录说明可以了解到，目标机所对应的目录都在 target 目录下，BSP 就来自于 target/config 目录。除了 all、comps 两个目录之外，目录中所有的 BSP 都会显示在下拉列表框中。

编译器选择框中将会列出用户在 Tornado 中安装的编译器，在本章的后面可以看到如何向 Tornado 中增加编译器。

在映像类型选择框中，用户可以选择多种类型的映像形式。选项中较为特殊的一个是“clean”功能，执行该功能可以将编译后的目标文件清除，避免出现未检测到文件改动而导致的错误。其余的映像类型根据 3 种不同的格式和 3 种不同的压缩方式组合出了 9 种不同的类型。

可以看到，9 种不同影像类型分别由 3 种不同前缀和 3 种不同后缀组合而成，其说明如表 2.1 所示。

表 2.1　　BootRom 的类型

前缀或后缀	说　　明
bootrom	压缩的 bootrom 映像
bootrom_uncmp	非压缩的 bootrom 映像

bootrom_res	Rom 驻留型的 bootrom 映像
无后缀	ELF 可执行文件格式
.hex	Motorola S-record 文件格式
.bin	裸的二进制格式

2.1.3 实例：制作软盘引导盘

本小节的实例将描述如何制作目标机的软盘引导盘。

1．材料准备

练习本例需要准备以下物品。

- x86 目标机一台（可以使用 386 及以上机型，效果类似）；
- 软盘一张；
- 安装有 Tornado 系统的主机一台。

2．实例说明

由于目标机的关系，本例中以最容易搭建的 X86 主机为蓝本，其他架构主机可以根据实际情况适当练习。

3．操作步骤

（1）编译 BootRom 并设置环境。
选择菜单“build→build bootrom”，在弹出窗口中选择 BSP、映像类型和编译工具。例如，选择 BSP 为 pcPentium，选择映像类型为 bootrom，选择编译工具为 gnu，单击“OK”按钮。Tornado 工具将会生成一个 bootrom 映像，文件名为 bootrom。
（2）建立编译环境。
打开“命令提示符”，进入 Tornado 安装目录下的主机目录“C:\Tornado2.2\host\x86- win32\bin”，运行“torvars.bat”批处理命令。该命令为用户设置路径等环境变量，搭建出必要的工作环境。
（3）制作引导盘。
首先，将软盘格式化，文件系统选择 FAT 或 FAT16 格式。该过程可以通过 format 命令完成，也可以在资源管理器中完成；
然后，在“命令提示符”窗口中使用命令“cd ..\..\..\target\config\BSPName”，进入 BSP 对应的目录。
最后，使用 Tornado 提供的 mkboot 工具将 bootrom 映像复制到软盘中，制作成引导盘。该过程只能通过命令行由 mkboot 命令完成，如下所示。

```
    mkboot a: bootrom
```

其中，a:是软盘的盘符，bootrom 是 VxWorks 映像的文件名。该命令首先利用 vxsys.com 将一段简短的引导程序写入软盘的引导扇区，然后将 bootrom 经过处理复制到软盘中，并将其重命名为 bootrom.sys。关于 mkboot 的具体过程将在下一节讨论。
至此，一个 VxWorks 引导盘就制作完成了。
（4）引导目标机。

用引导软盘引导目标机的方法极为简单：将目标机设置为从软盘启动，插入制作好的引导软盘，开机即可，如图 2.5 所示。

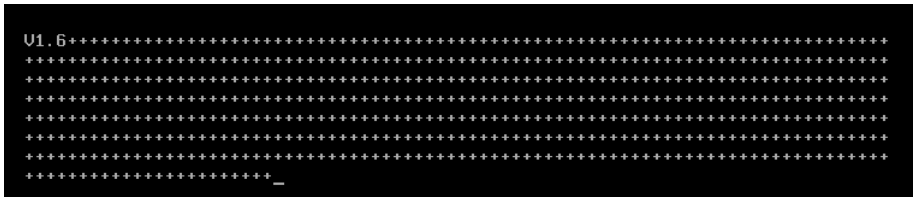

图 2.5　Bootrom 的引导过程

其中，V1.6 表示引导程序的版本，连续的“+”号则表示引导的进程。根据 bootrom 映像类别的不同，“+”号的长度也不同。

（5）显示引导控制台。

VxWorks 5.5 与 VxWorks 5.4 的一个不同点体现在引导过程中对控制台的定义。默认的情况下，VxWorks 5.5 不提供控制台的支持。如果需要支持控制台，则要修改 config.h 中如下所示的一行。

```
#undef INCLUDE_PC_CONSOLE
```

将其中的“#undef”修改为“#define”，而后重新制作 bootrom 并引导目标机，即可以进入 bootrom 提示行状态，如图 2.6 所示。

如果在引导过程中找不到 bootrom.sys 文件，系统将会出现图 2.7 所示的提示。

出现这样的情况需要将 bootrom.sys 文件重新复制到软盘上。

软盘引导目标机是最简单的办法，也是 VxWorks 的默认引导办法。在软盘不再大量使用的今天，其他类型的引导设备更具有快速、便利等优势。下面几个小节将提供其他几种引导盘的制作方法，由于步骤大体相似，部分过程不再详细说明。

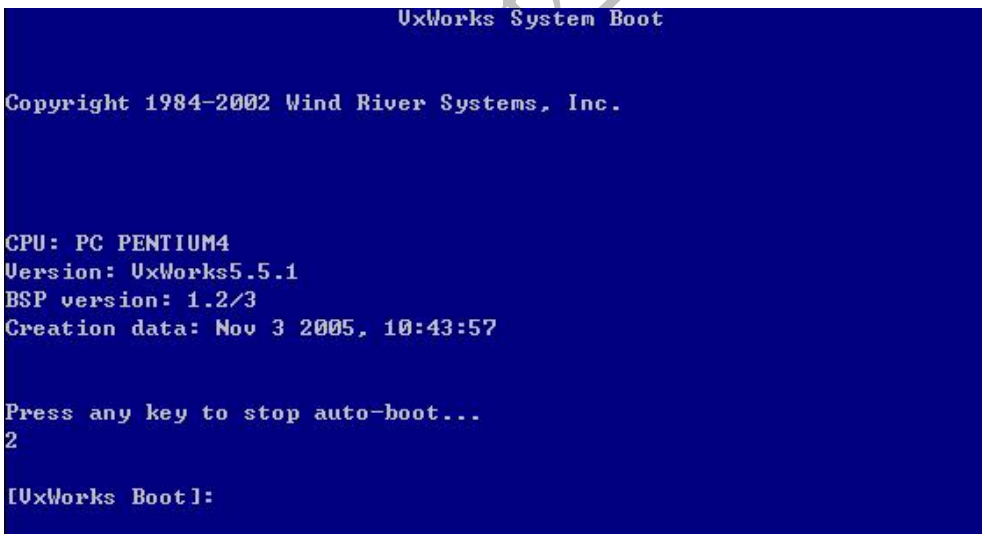

图 2.6　正确引导 BootRom 后的界面

图 2.7　系统提示

2.1.4　深入 1：批处理文件 mkboot

上一小节的实验的第三步中用到了一个叫作 mkboot.bat 的工具，这是 Tornado 为制作引导文件提供的批处理文件，用于根据编译后的目标文件生成适合于引导的文件。

mkboot 在 Tornado 安装目录下的主机文件目录中，可以在“安装文件夹\host\x86-win32\bin”目录下找到。mkboot 总共完成了 3 件事：使用 vxsys 为磁盘创建引导扇区，生成适合引导的二进制文件，以及检查文件是否连续。

编辑 mkboot.bat 可以在第 5 行看到 vxsys 工具的使用。vxsys %1，意思是为参数%1 所指定的磁盘创建引导扇区。vxsys 可以通过直接磁盘访问的方式为软盘、硬盘、U 盘等创建引导扇区，只需传入对应的盘符参数即可，例如，为 A:盘创建引导扇区可以使用 vxsys a:。

需要注意的是，在 Windows 系统中，直接硬盘访问是受到限制的。在 Windows 的命令提示符下无法利用 vxsys 在硬盘或 USB-HDD 格式的 U 盘上创建引导扇区，只有在 DOS 6.22 下制作或者使用 lock 命令解锁硬盘才能解决这个问题。

引导扇区创建完成之后，mkboot 开始创建 bootrom.sys 文件。根据来源文件的不同，mkboot 区分了两种情况创建 bootrom.sys 文件。一种是带有格式信息的二进制文件，包括 bootrom.dat、bootrom、bootrom_uncmp、vxWorks_rom 等，这些文件都需要将格式信息去掉之后才可以作为引导文件，mkboot 使用 objcopypentium 命令将格式信息裁减掉并创建 bootrom.sys 文件；另一种输入文件是纯二进制文件，包括 bootrom.bin、bootrom_uncmp.bin、vxWorks_rom.bin，这些文件中仅包含程序代码的二进制信息，可以直接使用 copy 命令复制成为 bootrom.sys 文件。

最后一步，也是非常关键的一步，就是磁盘检查。mkboot 使用 chkdsk 命令检查磁盘情况并确定 bootrom.sys 文件是否连续存放。这一步并不产生实质上的效果，然而，为什么一定要有这样的一步呢？这需要从引导过程说起。引导过程是一个复杂的过程，其实际情况还要在关于 BSP 的章节中详细说明，这里仅简单描述一下引导扇区开始执行后的过程。这里只针对 x86 系统，其他体系架构有少许不同（参见关于 BSP 的章节的说明）。

众所周知，引导扇区中保存的是一段小程序，这段程序只有 512 字节。计算机引导成功后，BIOS 从磁盘的引导扇区读取这段程序，然后将 CPU 执行的权利交给它。不同操作系统的引导扇区中代码也不相同，然而，不论什么样的操作系统，在如此小的程序中也只做一件事——加载一个可以引导系统的文件。VxWorks 的引导代码所做的工作就是将 bootrom.sys 文件解压缩并加载在内存适当的位置中，然后将 CPU 程序指针指向装载后的程序入口处并执行。由于引导代码容量的限制，VxWorks 无法完成太多的任务，于是它采取按照连续顺序读取文件的方法读取 bootrom.sys。然而，问题就在这个方法上出现了，如果 bootrom.sys 文件在磁盘中的存储位置不连续（关于为什么可以不连续请参考 DOS 相关书籍），引导代码读入的文件就可能是任何数据。对于这样的数据，解压缩并将其安装在内存中，再将程序的执行权交给它，后果可想而之。通常，这时可以看到系统重启，也可能会看到系统引导过程停止，或者还有其他情况发生。基于这样的原因，chkdsk 的检查就变得很重要了，而且，最需要关注的是检查完后是否会出现下面的报告。

```
All specified files are contiguous.
```

如果出现了上述的语句，表示离成功已经很近了。如果出现类似于下面的语句，则表明磁盘或者有坏块，或者无法为 bootrom.sys 文件分配连续的块。

```
bootrom.sys contains 2 non-contiguous blocks.
```

这时，可以试着删除磁盘上的其他文件，然后重新复制 bootrom.sys 文件。

2.1.5 深入 2：制作 U 盘引导盘

目前最流行的设备接口当属 USB 接口，嵌入式设备也不甘人后，纷纷增加 USB 设备接口。USB 设备使用方便，具有较高的访问速度，在制作引导盘上远远优于软盘引导盘。下面将简要介绍制作 USB 引导盘的过程。

1．材料准备

练习本例需要准备以下物品。

- x86 目标机一台（需要可以在 BIOS 中设置 U 盘引导的目标机）；
- U 盘一块，为确保成功最好不要使用容量太大的 U 盘；
- 安装有 Tornado 系统的主机一台。

2．实例说明

该实例描述制作 U 盘引导盘的过程，制作好的 U 盘可以引导 VxWorks 操作系统，但是无法通过 U 盘加载操作系统映像（加载操作系统映像的具体办法参见第 3 章）。实例依然是以 x86 体系架构为例，使用其他体系架构的读者请根据步骤自行分析制作过程。

3．操作步骤

（1）在制作 VxWorks 的 U 盘引导盘之前，首先要在主机上将 U 盘制作成为一个可以引导 DOS 系统的系统盘。

制作这样的系统盘有很多选择，USBoot、FlashBoot 等都是很好的引导 U 盘制作工具。建议将 U 盘制作成为 USB-HDD 模式（这里没有任何限制，USB-HDD、USB-ZIP 都可选择，这样只是为了较易描述下面步骤）。不同目标机对不同格式的支持程度也不一样，需要根据实际情况多试几次。

制作好引导 U 盘之后，需要在目标机上试验一下，如果能够引导 DOS 系统则可以进行步骤（2）。不同的目标机需要在 BIOS 进行不同的设置，以比较常见的 AwardBIOS 为例，需选择主菜单下的“Advanced BIOS Features”选项，而后选择引导设备。根据 U 盘制作过程中的不同选择相应的选项，在这个实例中，选择 USB-HDD 设备作为第一引导设备。

选择完毕后重新启动目标机。如果一切正常的话，将会进入 DOS 系统，即开始步骤（2）；如果不能正常引导，很可能出现下面的几种情况。

- 目标机不支持 U 盘格式。根据 BIOS 的不同，目标机可能支持的 U 盘格式也不同，不被支持的引导盘格式就不能顺利引导系统。在这种情况下，可以使用工具将 U 盘格式化成不同的格式一一实验。
- 目标机不支持 U 盘。如果试验过每一种格式后，系统仍然不能顺利引导，可能的原因是 BIOS 不支持 U 盘。虽然 USB 协议中定义了海量存储器的通信规范，然而 BIOS 对 U 盘的支持还是有一定的限制。这种情况下，可以试着更换一些比较早期的 U 盘，重新制作引导盘。
- 无法正确引导系统。即使是一些很新的 BIOS 也可能会因为某些原因不能引导 U 盘，例如，DELL 个别型号的机器无法正确识别出某些工具制作的引导盘。此时，可以试着换一个工具制作引导盘。

可能会出现的问题都可以通过更换格式、更换 U 盘或更换工具解决，只有顺利完成第一步才可以继续接下来的步骤。

（2）编译 Bootrom 并创建工作环境。

这个步骤与之前介绍的步骤相同，这里不再赘述。然而，制作引导型 U 盘的过程有一点不同，也就是 mkboot 的执行将不会成功。

由于 Windows 对硬盘的保护，mkboot 中的第一个步骤——创建引导扇区无法正确执行，这就需要手工执行下面的步骤。

根据对 mkboot 的分析（参见上一小节），可以轻松地找到执行后两个步骤的命令，如下所示。

```
objcopypentium -O binary --gap-fill=0 %2 %1bootrom.sys
chkdsk %1bootrom.sys
```

其中，%1、%2 都是批处理文件的参数。在这里需要将其替换掉，于是在 BSP 所在目录执行下面的操作。

```
objcopypentium -O binary --gap-fill=0  bootrom  bootrom.sys
```

这一步操作的目的是生成用于引导的 bootrom.sys 文件。接下来，将这个文件复制到目标 U 盘上，同时需要复制的还有“C:\Tornado2.2\host\x86-win32\bin”目录下的文件 vxsys.com。

（3）创建引导扇区。

准备好引导 U 盘之后，用其在目标机上引导 DOS 环境。如果引导的是 Windows 98 的 DOS 环境，需要执行 lock 命令解除系统对硬盘的锁定；如果引导的是 DOS 6.22 环境，则可以免去这一步操作。执行 lock 命令使用下面的格式。

```
Lock c:
```

这里的 C 盘不是系统中的硬盘而是 U 盘的盘符，以不同的工具、不同的格式制作出来的引导盘在引导之后盘符不同，所以这条命令要根据实际情况执行。

接下来创建引导扇区，在提示符下键入下面的命令。

```
vxsys c:
```

这里 C 盘的含义与上一步相同。执行完这一步之后，vxsys 就将引导扇区写入到磁盘当中，也就是说，这个磁盘已经无法引导 DOS 环境了，也意味着 VxWorks 的 U 盘引导就要制作成功。

（4）检查 bootrom 的连续性。

做完上述步骤之后，如果没有问题应该可以直接引导 VxWorks 了。为保险起见，还是先检查一下 bootrom.sys 的连续性。

将 U 盘插回主机，在命令提示符环境下运行如下命令。

```
chkdsk i:\bootrom.sys
```

I：是主机上 U 盘的盘符。chkdsk 命令将会报告 bootrom.sys 的连续性，不连续的文件将会导致引导过程失败。

至此，引导 VxWorks 的 U 盘就制作成功了。VxWorks 5.5 支持引导参数的保存，然而能够保存引导参数的设备较少，目前提供的有软盘、硬盘等，并没有提供对 U 盘的访问。所以，与软盘引导不同的是，在保存参数的过程中会出现错误。如果需要在 U 盘上保存参数，需要在 BSP 上稍加改动，这一点将在关于 BSP 的章节中提到。

2.1.6 深入 3：制作硬盘引导盘

引导代码在完全确定之后可能会固化在硬盘上，制作硬盘引导也就成为嵌入式操作系统必备的功能。硬盘引导具有绝对的速度优势，它比软盘引导快很多。与软盘引导盘的制作相比，制作硬盘引导的步骤有部分不同。

1．材料准备

练习本例需要准备以下物品。

- x86 目标机一台（需要配有硬盘）；
- 安装有 Tornado 系统的主机一台。

2．实例说明

制作硬盘引导的过程与制作其他类型引导盘相似，区别在于在制作硬盘 VxWorks 引导之前，首先需要制作一个可以引导操作系统的软盘或 U 盘引导系统。该实例与制作 U 盘引导盘非常相似，这里只做简要说明。

3．操作步骤

（1）制作 U 盘引导的 DOS 环境。这里需要做的事情与制作U盘引导盘的前两个步骤相同。
（2）引导目标机系统，制作引导扇区。
制作好引导型 U 盘之后，即可用其引导目标机系统制作硬盘引导。首先，通过 U 盘引导目标机系统进入 DOS 环境。如果硬盘已经格式化好且其格式为 FAT32 或 FAT16，那么进入 DOS 环境之后可以直接安装引导扇区，否则需要首先格式化硬盘。
与制作 U 盘的步骤相同，首先利用 vxsys 命令制作引导扇区，如下所示。

```
vxsys c:
```

这里的 c：就是硬盘的盘符。如果引导的是 Windows 98 制作的 DOS 环境，需要在这之前使用 lock 命令解除硬盘锁定。
创建引导扇区之后，依然是将 Bootrom.sys 文件复制到 c：盘根目录下。
（3）重新引导系统。
做完上述步骤之后即可重新引导系统了。由于前一步使用了 U 盘引导系统，重启之后需要将 BIOS 参数重新设置为硬盘引导。硬盘引导过程与软盘的引导过程相同，只是速度会快很多。

2.1.7 深入 4：制作一个引导 ROM

对于多数体系架构来说，bootrom 并不是作为文件被保存在磁盘中，最有可能的是保存在一个只读存储器中。本小节以一块 PowerPC 主板为例来演示如何将编译好的 bootrom 文件保存在只读存储器中。

1．材料准备

练习本例需要以下物品。

- PowerPC 目标机一台；
- 安装有 Tornado 系统的主机一台；
- 芯片烧写器一台。

2．实例说明

进行这个实例须保证两个条件：使用正确的 BSP，使用可以正常工作的 PowerPC 开发板。这两个前提缺一不可，任何一个出现问题都将导致这个实例的失败。如果没有正确的 BSP，可能需要仿真器来进行 BSP 开发，这不是本章涵盖的内容。如果没有一个可以正常工作的开发板，可以联系开发板提供商索取一块。

3．操作步骤

（1）编译并制作 bootrom。制作 bootrom 的过程与之前的实例相同，这里不再赘述，用户可以根据需要创建不同种类的 bootrom。
（2）将 bootrom 制作为二进制文件。
有些类型的 bootrom 不是只包含代码的二进制文件，需要将其转换为仅具有代码信息的二进制文件。
使用下面的命令可以达到该目的。

```
elf2bin bootrom bootrom.bin
```

当然，为了避免麻烦，可以在第一步生成 bootrom 时直接将之生成为仅包含二进制代码的文件。

（3）烧写芯片，引导系统。经过上一步的转换之后，bootrom 文件已经适合于烧写芯片。通过烧写器将其固化到芯片中，然后安装芯片到开发板上就可以引导 VxWorks 系统了。实际上，x86 系统的 Bootrom 也可以烧写到 ROM 中代替 BIOS 使用，这样可以大大提高系统引导速度，但需要对原有 BSP 进行较多的修改。不同的开发板具有不同的特性，很多开发板都没有显示机制，不能直接显示系统的引导过程。解决这个问题的方式很多，很多开发板都通过串口向主机发送信息。将串口与主机相连，通过超级终端即可看到主机的引导过程，如图 2.8 所示。

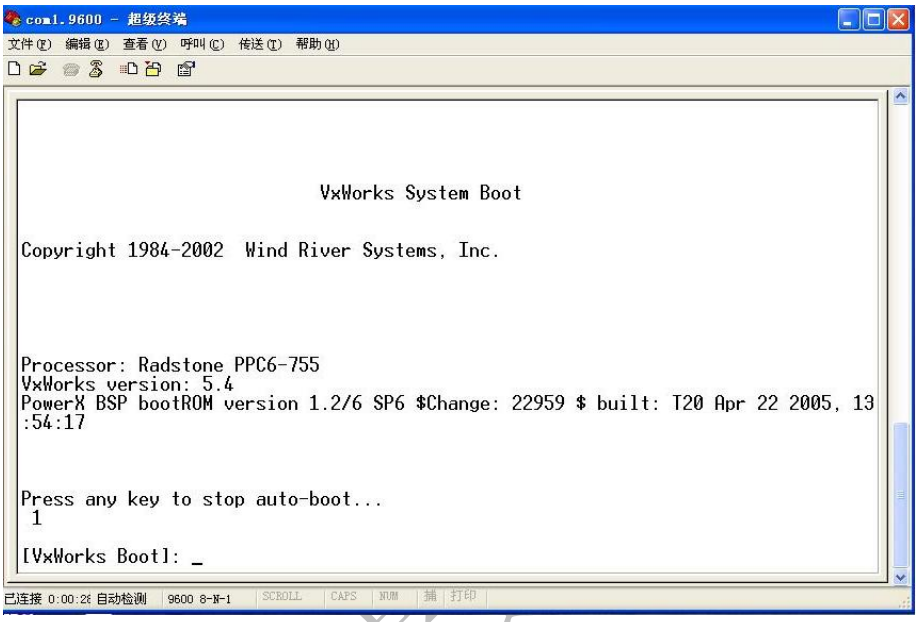

图 2.8　在超级终端中显示的引导信息

2.2　深入编译环境

2.2.1　创建的过程

利用 Tornado 2.2 创建目标文件的过程只是简化了手工的操作，其实质还是调用了外部的编译器、链接器。在选择创建目标文件之后，Tornado 系统会调用相应的命令完成创建。下面以“Rebuild All”为例演示创建过程。

执行“Rebuild All”命令后，Tornado 首先调用“vxrm”完成目标文件清除任务，如下所示。执行完该任务之后，所有可能生成的目标文件都会被删除，以避免出现因时间改变而导致的编译错误。执行这个程序同时被删除的不仅仅是目标文件，还包括工程之中的一些必要文件，这些工程文件将会在下一步中由模板文件重新创建。这样做的目的是避免错误的修改导致系统出现问题。

```
vxrm *.o *.rpo ctdt.c symTbl.c vxWorks* *.out *.pl
vxrm ..\prjComps.h ..\prjParams.h ..\prjConfig.c ..\linkSyms.c
vxrm ..\libs.nm ..\libs.size
```

vxrm 命令实际是一个批处理命令，编辑该文件可以看到真正完成清除工作的是“rm.exe”文件：“if exist %1 rm -f %1”。

其后，Tornado 通过“wtxtcl”执行 Tcl 程序“configGen.tcl”，生成工程所必需的文件，如下所示。

```
wtxtcl C:\Tornado2.2\host\src\hutils\configGen.tcl ..\xxx.wpj
```

xxx.wpj 就是例子工程的工程文件。

做完这样的准备工作后，Tornado 调用编译器“ccpentium”完成每一个文件的编译，如下所示。

```
    ccpentium -c -g -mcpu=pentiumiii -march=p3 -ansi -nostdlib -fno-builtin -fno-defer-pop -P
-xassembler-with-cpp  -I..  -IC:\Tornado2.2\target\config\  ETXA5363  -IC:\Tornado2.2\target\h
-IC:\Tornado2.2\target\config\comps\src  -IC:\Tornado2.2\target\src\drv  -DCPU=PENTIUM4  -DTO
OL_FAMILY=gnu -DTOOL=gnu -DPRJ_BUILD  -fvolatile C:\Tornado2.2\target\config\ETXA5363\sysALib.s
-o sysALib.o
    ……
```

最后，由链接器“ldpentium”生成目标文件 VxWorks，如下所示。

```
    ldpentium  -X  -N  -e  sysInit  -Ttext  00308000  dataSegPad.o  partialImage.o  ctdt.o  -T
C:\Tornado2.2\target\h\tool\gnu\ldscripts\link.RAM -o VxWorks
```

除了上述的编译链接工作外，Tornado 最后还调用“vxsize”，向用户报告所编译的目标文件大小。

这就是整个创建过程。在创建的过程中，IDE 环境所做的就是根据用户的选项执行不同的步骤。例如，单个文件的创建只包含项目文件的处理过程和编译过程。

```
    wtxtcl C:\Tornado2.2\host\src\hutils\configGen.tcl ..\G2NBase.wpj
    ccpentium -g -mcpu=pentiumiii -march=p3 -ansi -nostdlib -fno-builtin -fno-defer-pop -I..
-IC:\Tornado2.2\target\config\ETXA5363                -IC:\Tornado2.2\                target\h
-IC:\Tornado2.2\target\config\comps\src    -IC:\Tornado2.2\target\src\    drv    -DCPU=PENTIUM4
-DTOOL_FAMILY=gnu -DTOOL=gnu -D PRJ_BUILD  -c ..\G2N.c
    Done.
```

2.2.2 抛弃 IDE

从上一小节的描述可以看到，Tornado 在编译过程中根据用户的需求调用了各种工具，实际上，这样的过程完全可以由手工完成。抛弃自动化，使用手工操作有两个理由：其一，手工操作更灵活，能完成一些自动化做不到的事情；其二，了解了手工操作的原理，可以在解决问题的时候分析问题的细节。

下面来看看手工操作与自动化操作的对应关系。

首先从操作环境入手。自动化操作有其必要的环境，手工操作也需要必要的环境。这方面，Tornado 已经为用户考虑到了。打开“命令提示符”，进入 Tornado 安装目录下的主机目录“C:\Tornado2.2\host\x86-win32\bin”，运行“torvars.bat”批处理命令。该命令为用户设置路径等环境变量，搭建出必要的工作环境。如果经常进行手工编译的话，可以创建一个批处理文件直接完成环境设置的过程。批处理命令的内容如下（此处以 C 盘为 Tornado 的安装目标盘）。

```
    C:
    cd\
    cd C:\Tornado2.2\host\x86-win32\bin
    call torvars.bat
    cd\
```

运行该批处理文件的效果与手工操作效果一致。

设置完操作环境之后，就可以手工编译工程了。这里要用到的命令是 make，而对于 make 来说，最重要的文件是 MakeFile。

单是有关 make 命令的参数和操作模式就可以独立写成一本书。本书不会对其深究，只是比较一下手工操作的创建过程与自动化的创建过程之间的联系。MakeFile 是以文本形式存在的编译脚本文件，它的作用将在下一节中提到。

使用 make 命令可以简单地完成编译的任务。这里只有两个命令格式用于对应自动化的编译过程——“make”和“make clean”。

经过前一节的分析可知，“Rebuild All”命令首先执行了“vxrm”命令，而在手工操作时这个命令则是通过“make clean”完成的。运行“make clean”，即可执行清除目标文件动作。
剩下的编译工作就更简单了，键入“make”命令即可执行由“makefile”定义的全部动作。

2.2.3 使用 Diab

Diab 编译器是 WindRiver 公司提供的 C/C++编译器，目前最新版本为 5.2。该编译器支持多种体系架构，如 PowerPC、MIPS、68K、ARM、SPARC 等。
WindRiver 专门为嵌入式系统优化了 Diab 编译器，使其生成的代码具有体积更小、执行效率更高的特点。WindRiver 宣称，配合 Diab，VxWorks 将提供更优秀的性能。在编译 C++程序时，Diab 编译器的效果确实非常明显，其编译得到的目标代码体积较小。
安装时，按照提示执行安装步骤即可在 Tornado 2.2 中添加 Diab 编译器。安装完毕后，可在编译工程的时候选择使用 Gun 编译器或 Diab 编译器，如图 2.9 所示。

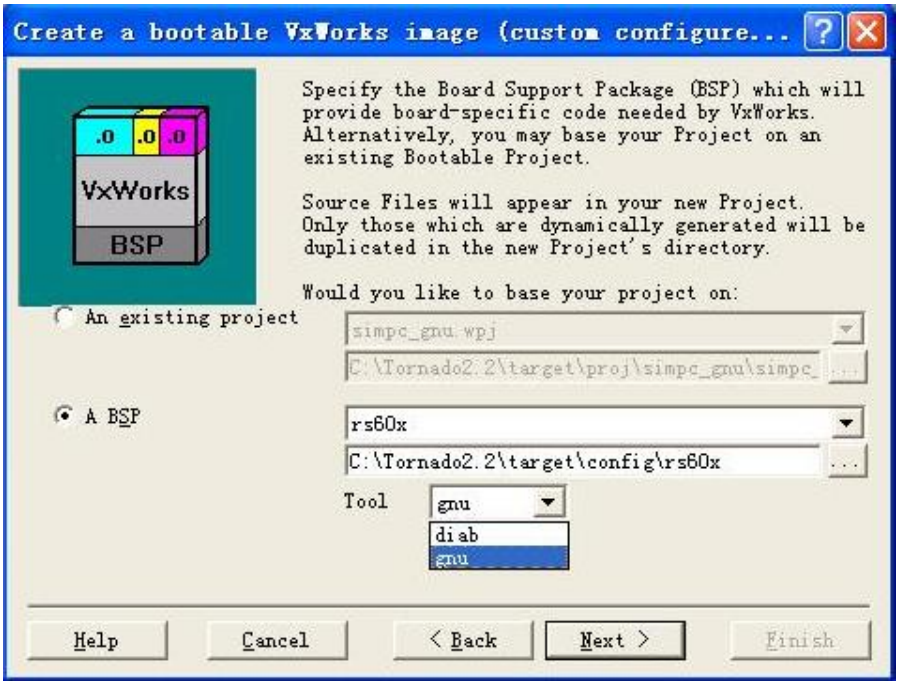

图 2.9　安装 Diab 后的工具链选择

使用 Diab 编译器需要注意的一点是，在编译 C++源文件时，由于符号表不同，由 Diab 编译的工程文件和由 Gun 编译的工程文件之间不能互相调用。

2.3 了解 MakeFile

在 Windows 下编程基本上不需要了解 MakeFile，因为其友好的开发环境已经为用户完成了相关的工作。然而，使用 GNU 编译器的时候则必须了解一下 makefile 的内容和功能。深入了解 makefile 将会为工作带来很多便利，有时，只有修改 makefile 才可以实现某些功能或解决一些错误。

2.3.1 MakeFile 的作用

笼统地说，makefile 是一种编译描述脚本，这种脚本文件决定了工程编译时的规则、编译顺序、目标代码的类型等内容。

对于大型工程来说，其工程源文件不计其数，源文件一般按照类型、功能、模块等被分别放置在不同的目录中，这样就会为编译器带来问题。哪些文件需要编译？哪些文件需要首先编译？哪些文件有什么样的特殊编译要求？这些问题都由 makefile 来解决。

makefile 的两个主要功能就是提供文件之间的依赖关系和目标文件生成方法。makefile 由 make 命令来解析，并根据其脚本规定的内容执行编译动作。

makefile 为用户带来的好处是编译自动化。用户不需要每一次编译都重新执行源代码的编译命令，而只要建立一个 makefile 文件指定编译规则，在之后的编译中执行 make 就可完成编译过程。这样的做法极大地提高了软件开发效率。

2.3.2 MakeFile 格式

makefile 的基本格式很简单，其变体却可以很复杂。这里仅讨论简单的 makefile 格式，目的是使读者能够读懂 makefile 文件，简单分析其编译过程。关于 makefile 的复杂变体设计的内容，有兴趣的读者可以查看 gnu 编译器的相关说明，也可以参考 Linux 下 makefile 的说明。

makefile 的基本规则如下。

```
    target: prerequisites
                command
```

其中，target 表示目标文件，它也可以是一个集合或一种规则的名称；prerequisites 表示目标文件所依赖的其他文件，通常这是代码源文件和头文件，多个文件使用空格分隔开；command 则表示了所要执行的编译命令，当然并不只限于编译命令，其他命令也是可以的。这里可以创建一个简单的工程，看看 Tornado 生成的 makefile 文件是什么形式的。创建一个基于模拟器的可引导工程，打开工程文件所在目录下的 makefile 文件，可以看到如下的代码（为方便描述，这里每一行都添加上标号）。

```
    ①usrAppInit.o: $(PRJ_DIR)/usrAppInit.c

    ②usrAppInit.o:
    ③  $(CC)  -g  -mpentium  -ansi  -fno-builtin  -fno-defer-pop  -I$(PRJ_DIR)
-I$(WIND_BASE)/target/config/simpc -I$(WIND_BASE)/target/h -I$(WIND_BASE)/target/ config/comps/src
-I$(WIND_BASE)/target/src/drv  -DCPU=SIMNT  -DTOOL_FAMILY=gnu  -DTOOL=gnu  -DPRJ_BUILD  -c
$(PRJ_DIR)/usrAppInit.c
```

可以看到，这里的语法要复杂一些，实际上这是一种变体，三句放在一起就比较好理解。第一句包含了目标和依赖关系，编译所生成的目标文件是 usrAppInit.o，其来源是工程目录下的 usrAppInit.c 文件。第二句和第三句是一个整体，表示生成目标文件 usrAppInit.o 需要用到第三句所包含的命令。两者相结合表示使用给出的命令编译生成 usrAppInit.o 文件。

为了提高灵活性，makefile 还提供了变量的概念。在上例中，以“$(…)”形式出现的内容就是变量的引用。在刚刚创建的工程的 makefile 中，可以看到类似下面的语句。

```
    CC            = ccsimpc
    PRJ_DIR       = ..
```

这里就是为之后所引用的变量赋值。在这个文件中，复制之后$(CC)的含义就是“ccsimpc”，$(PRJ_DIR)的含义就是“..”。在替换后，前例中的第三句就变为：

```
    ccsimpc -g -mpentium -ansi -fno-builtin -fno-defer-pop –I.. -I$(WIND_BASE)/ target/config/simpc
-I$(WIND_BASE)/target/h  -I$(WIND_BASE)/target/config/comps/  src  -I$(WIND_BASE)/target/src/drv
-DCPU=SIMNT -DTOOL_FAMILY=gnu -DTOOL=gnu -DPRJ_BUILD  -c ../usrAppInit.c
```

搜索整个 makefile 之后，可以发现一个奇怪的问题——这个文件中没有定义“WIND_ BASE”。那么这个定义又是从哪里来的呢？这就牵扯出 makefile 的环境变量问题。

环境变量就是操作系统中所定义的，用于指定操作系统运行方式的参数，在不同的操作系统中都有定义。例如，在 Windows 系统下，可以在“我的电脑”属性的高级选项卡中看到环境变量的参数，如图 2.10 所示。

图 2.10　Windows 下的环境变量

也可以在 DOS 提示符下使用 set 命令查看环境变量。makefile 中可以应用当前系统中的环境变量，其使用方法与 makefile 的文件相同。

为了尽量简化工作，加强 makefile 的功能，makefile 还支持自动推导。也就是说，对于相同的目标文件来说，通过 makefile 可以找到并执行其指定的功能。回过头看看前面的例子，第一句与第二句、第三句之间还有其他的语句，但是通过自动推导功能可以知道，编译目标文件所使用的源文件是 usrAppInit.c，其编译命令由第三句指出。

所以，在 Tornado 所生成的 makefile 中很明确地分为多个部分，功能相似的代码被放置在一起以方便用户查看，例如，依赖关系的定义放在“## dependencies”注释之后，搜索路径则定义在“## set searching directories for dependencies”部分中。

最后还有一个问题，make 文件如何区分命令和目标？难道只是根据命令之前的空格？

确实如此，然而这不是简单的空格，这是一个用“Tab”键输入的表格符。通过二进制方式查看 makefile 文件，可以发现命令前的字符 ASCII 码是“09”，也就是按下“Tab”键所产生的字符。当然，一个或多个“Tab”键都被解释成相同的内容，即“这是一个命令的开始”。

2.3.3　解决 MakeFile 所导致的错误

本小节所描述的错误只会出现在 Tornado 2.0 下，Tornado 2.2 已经解决了这个问题。之所以翻出来这个问题，主要是想说明一下 makefile 的重要性。用户如果在编译过程中碰到问题可以分析一下相应的 makefile。

在复制或移动使用 Tornado 2.0 创建的工程后，通常会遇到编译问题，如某某目录下的某某文件找不到，从而导致编译错误。

导致这个问题的原因就是，Tornado 在生成工程 makefile 时将其中的工程路径变量赋值成了工程绝对路径。例如，在 d:\MyProj 目录下创建了工程，其 makefile 中就会有如下的一项。

```
PRJ_DIR      = D:\MyProj
```

这样，如果工程目录发生了改变，而编译器还是寻找这个目录，就会导致编译错误。解决的方法就是将路径变量改为相对路径，如下所示。

```
PRJ_DIR      = ..
```

此时重新编译工程文件就不会再出现文件找不到的问题了。

联系方式

集团官网：www.hqyj.com 嵌入式学院：www.embedu.org 移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn 物联网学院：www.topsight.cn 研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路 250 号银海大厦 11 层 B 区，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218