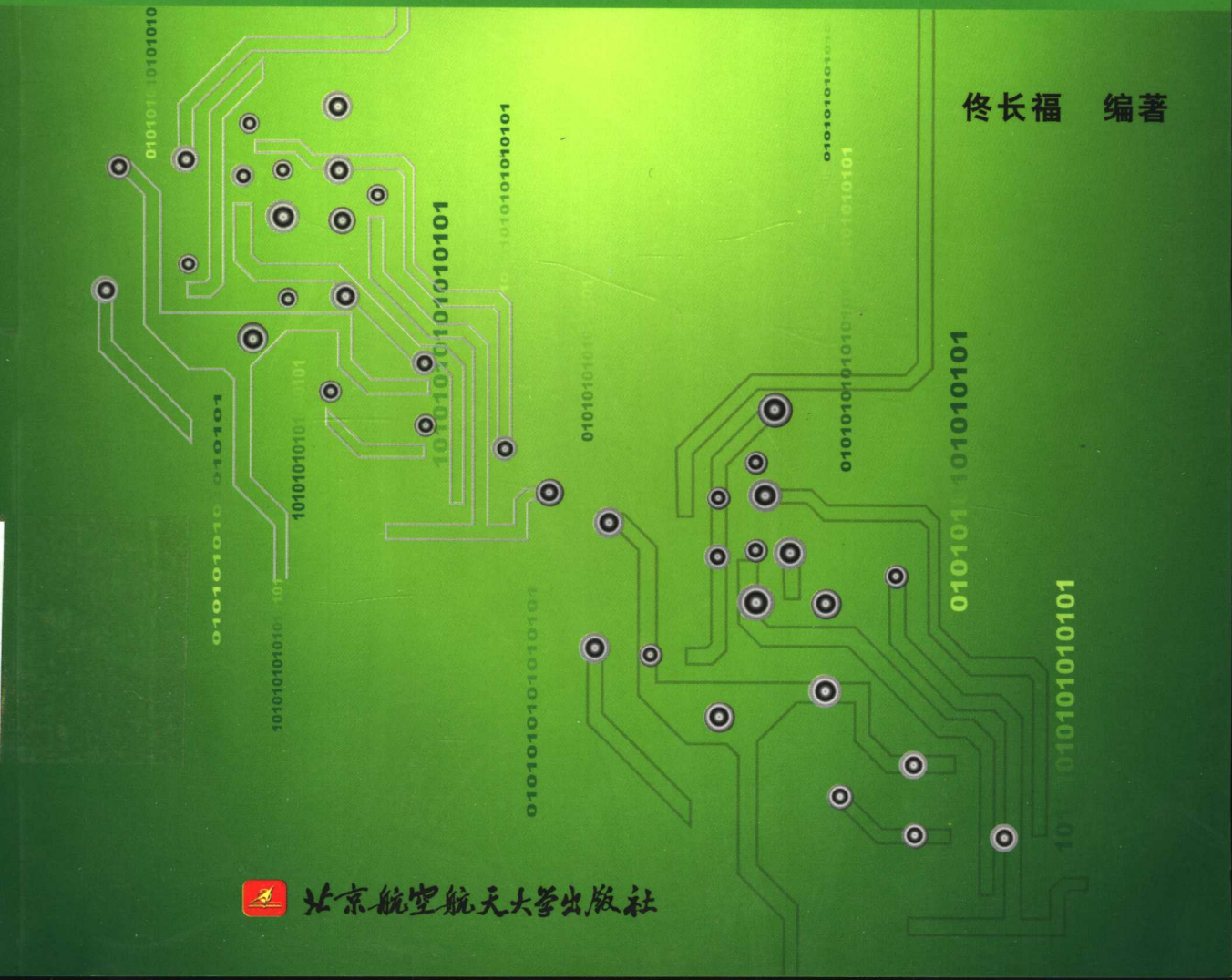
MCU

AVR单片机GCC程序设计

佟长福 编著



北京航空航天大学出版社

AVR 单片机 GCC 程序设计

佟长福　编著

北京航空航天大学出版社

内 容 简 介

本书全面讲述基于 AVR-GCC 的 AVR 单片机 C 语言程序设计。首先根据不同编译器对单片机存储器操作上的不同，详细介绍 AVR-GCC 的操作存储器方法；随后以大量的实例程序演示 AVR 单片机内部集成功能模块的 C 语言程序设计方法。多数示例程序均基于 ATmega8 单片机，并在实际硬件上调试通过，对掌握和编程其他 AVR 器件具有较高的参考价值。

本书适合于有一定单片机和 C 语言基础知识的工程技术人员、高等院校相关专业师生使用。

图书在版编目(CIP)数据

AVR 单片机 GCC 程序设计/佟长福编著．—北京：北京航空航天大学出版社，2006.1

ISBN 7-81077-724-6

Ⅰ.A… Ⅱ.佟… Ⅲ.①单片微型计算机，AVR—程序设计②C 语言—程序设计 Ⅳ.①TP368.1②TP312

中国版本图书馆 CIP 数据核字(2005)第 114514 号

© 2006，北京航空航天大学出版社，版权所有。

未经本书出版者书面许可，任何单位和个人不得以任何形式或手段复制或传播本书内容。侵权必究。

AVR 单片机 GCC 程序设计

佟长福 编著

责任编辑 王慕冰

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(100083) 发行部电话：010-82317024 传真：010-82328026

http://www.buaapress.com.cn E-mail:bhpress@263.net

涿州市新华印刷有限公司印装 各地书店经销

*

开本：787×960 1/16 印张：17 字数：381 千字

2006 年 1 月第 1 版 2006 年 1 月第 1 次印刷 印数：5 000 册

ISBN 7-81077-724-6 定价：28.00 元

前　言

首先，建议读者访问本书的网站 http://www.chip-art.net，从网站可下载本书中示例程序的源代码和一些补充内容，并可查看勘误表等信息。希望读者以网站上的联系方式提出对本书的建议并指出内容中存在的错误，作者将设法保证网站的必要更新。

2004 年 10 月作者用“芯艺”署名，在网上发布了一个标题为《AVR 单片机与 GCC 编程》的 PDF 格式文档，之后得到了广大网友的认可和支持，他们提出了很多宝贵的意见和建议。本书正是在此基础上完成的，感谢这些网友的支持。真正撰写本书是在北京航空航天大学出版社胡晓柏编辑的鼓励和支持下开始的，在整个出版过程中他给予了很大的帮助，在此表示衷心的感谢。

本书为 AVR 单片机 GCC 开发者提供了有价值的信息，内容包含众多完整的示例程序供参考，尽可能地采用简单而直接的方式描述问题。本书的读者假设为具有良好的 C 语言基础和单片机知识的设计人员。建议读者在学习过程中努力创造硬件环境。作者认为，每当一个单片机按预先编好的程序正确运行时，都会带来一种成就感，使人对此产生更大的兴趣，激励自学。AVR 单片机的硬件开发可以做到非常廉价，以致于在校学生都可以接受。

书中多数示例选用的单片机为 ATmega8。ATmega8 是 ATMEL 公司在 2002 年推出的一款 AVR 单片机，它内部集成了 AVR 系列单片机的多数功能，并且价格低廉，在国内的应用较广，因此比较适用于学习。如果示例程序没有特殊说明，均使用版本 WinAVR20040720 编译，随着 AVR 不断地推出新款单片机，WinAVR 的更新也较快，目前的最新版本为 WinAVR20050214。之所以更新那么快，是为了支持更多新器件，所以不论哪个版本，只要支持所用到的器件，都是可以用的。

本书的结构是按尽可能最好地解释和描述“AVR 单片机 GCC 开发”的方式来编排的。全书共包含 12 章。

- 第 1 章描述了 AVR 单片机及 GCC 的总体情况，并以一个简单示例的方式介绍了用 WinAVR 编译一个 AVR 应用程序的整个过程；还介绍了学习本书内容时所用到的软硬件环境和作者用于测试本书示例的实验板 CA-M8。
- 第 2 章详细描述了 AVR 单片机内部各种存储器的组织结构及在 C 语言程序中的操作方法。
- 第 3 章主要讨论了单片机程序结构问题，相信这对于初学者会有所帮助。
- 第 4 章介绍了 AVR 单片机内部功能模块及其编程操作方法，并为每个模块的应用列举了相应示例。

- 第 5 章详细叙述了异步串行通信模块的应用技术。之所以把 UART 单独列入一章，是因为在作者的观点中 UART 对单片机系统非常重要。
- 第 6 章介绍了一种用 AVR 单片机实现的 AT89S52 编程器。实际上，它是一个 AVR 单片机最基础的应用示例。
- 第 7 章举例介绍了 AVR 单片机内部集成的两线串行接口模块(TWI)及其编程。
- 第 8 章举例介绍了 AVR 单片机 BootLoader 功能。
- 第 9 章介绍了 AVR-GCC 对汇编语言的支持，其中包括在 C 语言程序中嵌入汇编、独立的汇编语言支持及 C 语言与汇编语言混合编程等方面。
- 第 10 章介绍了 AVR-GCC 对 C++语言的支持。
- 第 11 章详细叙述了三个应用实例，分别是“双基色 LED 屏控制”、“工作小时计的制作”和“电话远程控制系统”，它们包括了单片机应用中的各个方面，对于设计开发人员具有很高的参考价值。
- 第 12 章主要介绍了 AVR 单片机上一个自由操作系统 AVRX。它能运行于大多数 AVR 单片机。

以上各章节是按循序渐进的方式编排的，作为初学者，应从第 1 章开始阅读，并努力做好书中每一个实验，这将有助于加深印象。

尽管一开始觉得写这本书有些挑战，但每次读者的认可和建议让作者意识到了它的价值所在，希望我们共同努力的结果能在本书的内容中得以体现。再次感谢提出过建议的网友，也感谢购买本书纸版的读者。你们的建议将是对作者最大的鼓励，你们的认可将是给作者带来的最大快乐。另外，为本书的编写提供帮助的还有范士勇、庆格勒图、宋艳楷、姚国珍、满都胡、张翼、张华、周坤、陆新志、李虎林、安强、乔飞和杨勇等，在此表示衷心的感谢。

作　者

2005 年 10 月

于内蒙古包头

试读结束，需要全本PDF请购买 www.ertongbook.com

目　录

第 7 章　硬件 TWI 端口编程

第 8 章　BootLoader 功能应用

第 9 章　汇编语言支持

试读结束，需要全本PDF请购买 www.ertongbook.com

第1章 概 述

1.1 AVR 单片机 GCC 开发概述

1.1.1 AVR 单片机介绍

1. 诞 生

1997 年 ATMEL 公司挪威设计中心的 A 先生和 V 先生出于市场需求的考虑，推出了全新配置的 8 位精简指令集微处理器(RISC，Reduced Instruction Set CPU)，命名为 AVR。

2. 系列和主流

AVR 是一种指令内核的统称，它内部又分 ATtiny、AT90S 和 ATmega 三大系列，分别对应 AVR 的低、中、高档产品。ATtiny 系列中常用的有 ATtiny15、ATtiny2313 和 ATtiny26 等产品；AT90S 系列中常用的有 AT90S2313、AT90S8535 和 AT90S8515 等。AT90S2313 的引脚兼容 AT89C2051；而 AT90S8515 的引脚兼容 51 单片机，因此在设计中很容易替代 51 单片机。但是目前，AT90S 系列的绝大部分已停产。当某一个 AT90S 系列芯片停产时，ATMEL 公司通常会在 ATmega 或 ATtiny 系列中推出一个新的替代产品，替代产品往往在引脚兼容的基础上增加内部资源及提高性能。例如，ATtiny2313 作为 AT90S2313 的替代产品，在 AT90S2313 的基础上增加了片内标定振荡器、增强型上电复位、可编程的掉电检测等多种功能。与此类似，ATmega8515 和 ATmega8535 分别作为 AT90S8515 和 AT90S8535 的替代产品继承了很多 mega 系列的特性。

ATtiny26、ATtiny2313、ATmega48/88/168、ATmega8、ATmega16、ATmega32、ATmega64 和 ATmega128 是 ATmega 系列的主流产品。值得关注的是，ATmega8 这款单片机，以丰富的片内资源、低廉的价格而深受广大设计人员的喜爱，并在国内得到了较好的推广。然而，ATmega48/88/168 作为它的兼容产品，为用户提供了更多可选择的功能。

ATmega16 也是用量较多的器件，其引脚兼容 AT90S8535，可取代产品中的 8535 芯片。相对于 ATmega8，它除了 I/O 引脚多之外，内部集成了 2 倍于 ATmega8(16K)的 FLASH 程序存储器。

随着国内 AVR 用户的增多，ATMEL 公司也开始了主流器件数据手册的中文翻译工作。

从网络上可找到 ATtiny26、ATtiny2313、ATmega48/88/168、ATmega8、ATmega16、ATmega32 和 ATmega64 等主流器件官方翻译的中文数据手册。这给学习和使用 AVR 器件提供了极大的帮助。

3. 硬件结构

有关 AVR 的硬件结构和各功能模块的描述请参考数据手册或相关书籍。

4. 高级语言开发工具

更适合采用高级语言开发是最初设计 AVR 单片机的目的之一。目前，AVR 单片机高级语言开发工具主要有 IAR C、WinAVR、ICCAVR、CodeVision、BASCOM-AVR(BASIC 语言)。IAR 是与 AVR 内核协同开发的，很多对 AVR 更适合 C 语言开发方面的改进是根据 IAR 开发过程的，它也是 ATMEL 公司推荐的 C 编译器，但其价格几乎让人难以接受，达到上万元人民币。ICCAVR 是国内 AVR 主要推广单位双龙公司代理的 C 编译器，其价格低廉，友好的界面把很多烦琐的项目管理和编译设置隐藏起来，为此很受部分开发人员的欢迎。CodeVision 也是很好的 C 编译器，目前在国内也有一定数量的用户。WinAVR 是免费的 AVR 开发程序集，它以著名的自由软件 GCC 为 C/C++编译器。下面所有章节都会介绍如何使用 WinAVR 开发 AVR 单片机。学习 GCC 的意义绝不仅仅是为了开发 AVR 程序，正如 21ICBBS 上一位网友所说："如果其他编译器是一棵树，那么 GCC 就是树林"，GCC 支持多种处理器，包括 ARM、DSP、X86 等 32 位 CPU。它的历史足以说明它是成熟的编译器。

如果不是业余的程序开发人员，不会建议使用 BASIC 编写 AVR 程序，因为在编写与硬件密切相关的单片机程序时，通常需要想像编译器是如何将这些代码翻译成汇编语言程序的，而 C 语言更适合这样做。

1.1.2 GCC 编译器

UNIX 中最原始的 C 编译器叫 CC(C Compiler，C 编译器)，源于此，GNU 的 C 编译器叫作 GCC(GNU C Compiler)；然而，随着 GCC 支持语言的增加，GCC 这个缩写的意义已演变成了 GNU 编译器集合(GNU Compiler Collection)，它是 GNU 项目的一个产品，是一个开放源代码软件。

GCC 可编译多种语言，目前支持的语言有 C、C++、Objective-C、Fortran、java 和 Ada。

这些高级语言程序通过编译程序前端(front-end)后产生解析树，然后与器件相关的后端(back-end)程序将它们解释成实际的可执行指令集。前端与后端是完全分开的，解析树是它们中间的产物。GCC 这样的设计使得任何一种语言只要通过合适的语法解析器产生符合格式的解析树，就可以产生 GCC 后端程序支持的所有器件上的可执行指令集。同样，任何一种器件只要将树结构翻译成汇编语言，就可使用 GCC 前端支持的所有语言。

要承认的是，以上描述是理论化的，便于理解，实际操作并没有想像的那么简单。事实上，前端和后端都不是孤立的。幸运的是，AVR 的确得到了 GCC 的支持，它也是到目前为止

GCC 所支持的惟一一种 8 位处理器。不仅如此，还可在 Windows 平台上安装程序包 WinAVR 来使用 GCC 的 AVR C/C++编译程序。

WinAVR 是一组开放源代码的程序集，用于 ATMEL 公司 AVR 系列单片机的开发。它主要包含：

- GNU 程序包 Binutils。GNU Binutils 非常庞大，WinAVR 仅包含与 AVR 相关的部分，有 AVR 汇编器、连接器以及与机器指令相关的一些工具。
- GCC。它是 C/和 C++编译器，是 GNU 项目中的一部分。
- AVR-LIBC。AVR-LIBC 是 AVR 单片机 C 语言运行时库，它为应用程序提供标准 C 语言函数以及 AVR 器件专用的非标准库函数。

另外，WinAVR 还包含软件调试器 GDB、器件编程软件(avrdude 或 uisp)、文件格式转换工具(Srecord)和文本编辑器(Programmers Notepad)等多个有用工具，这里不一一列出，请参考 WinAVR 说明文档。WinAVR 项目的 Web 地址是 http://sourceforge.net/projects/winavr，这里可以下载最新的版本。

WinAVR 包含的程序包均属于 Linux 平台应用程序，这些 GNU 软件之所以能在 Windows 环境下运行，是因为有一个叫 Cygwin 的模拟 Linux 软件的支持。Cygwin 是一个可安装在 Windows 上的 Linux 环境。它由一个动态连接库 cygwin1.dll 和一些模拟 Linux 命令的工具组成。要知道，并不是 Linux 上的应用程序在 Cygwin 下直接可用，而是必须将其在 Cygwin 环境下重新编译才可运行。可以下载 WinAVR 在 Windows 下已经编译好的版本。它是一个以.exe 为扩展名的安装文件，安装方法与安装 Windows 应用程序类似，而不用另外下载 Cygwin 并安装。

1.2 一个简单的例子

为了先有一个感性的认识，首先看一下如下程序及其编译、连接过程。文件 demo1.c：

```
/*************************************************************************
    这是一个简单而完整的示例
    功  能：发光二极管闪烁
*************************************************************************/
#include <avr/io.h>                 //包含系统提供的 I/O 专用寄存器定义文件，在下面的代码中要
                                    //用到 DDRB、PORTB 等符号，为此要包含它

int main( void )                    //单片机 C 语言程序从该函数开始执行
{
    unsigned char  i, j, k,led = 0;
```

```
    DDRB = 0x01;                    //将 PB0 口设置为输出口

    while (1)                       //主程序通常要进入一个无限循环
    {
        if(led)                     //根据标记,置位或清零 PB0 口
            PORTB| = 0x01;
        else
            PORTB& = 0xFE;

        led = ! led;                //标记取反

        //延时操作
        for (i = 0; i<255; i++)
        {
            for(j = 0; j<255;j++)
                k++;
        }
    }//main loop
}//main
```

这是一个使接在 PB0 口的 LED 发光管闪烁的程序。有了源程序文件 demo1.c,就可以编译它了。通过选择“开始菜单”→“运行”,在弹出的对话框中输入 command ,打开控制台窗口,并在命令行输入:

```
avr-gcc  -mmcu=at90s2313  -c  demo1.c
```

如图 1-1 所示。

必须告诉编译器程序的 MCU 类型,这是通过命令行选项-mmcu 来指定的,指定的器件为 AT90S2313。-c 选项告诉编译器编译完成后不连接。

编译完成后,在工作目录新生成了一个文件 demo1.o,它是目标文件。再使用连接器将它连接成可在器件上执行的二进制代码。在命令行输入:

```
avr-gcc  -mmcu=at90s2313  -o  demo1.elf  demo1.o
```

之后会在工作目录看见连接器生成的 demo1.elf。GCC 连接后,生成的文件为 ELF 格式,在命令行通常用.elf 指定其扩展名。ELF 格式文件除了包含不同存储器的二进制格式内容外,还包含一些调试信息,所以还要借助一个有用的工具 avr-objcopy 来提取单片机程序存储器内容。在命令行输入:

```
avr-objcopy -j .text -j .data -O ihex demo1.elf demo1.hex
```

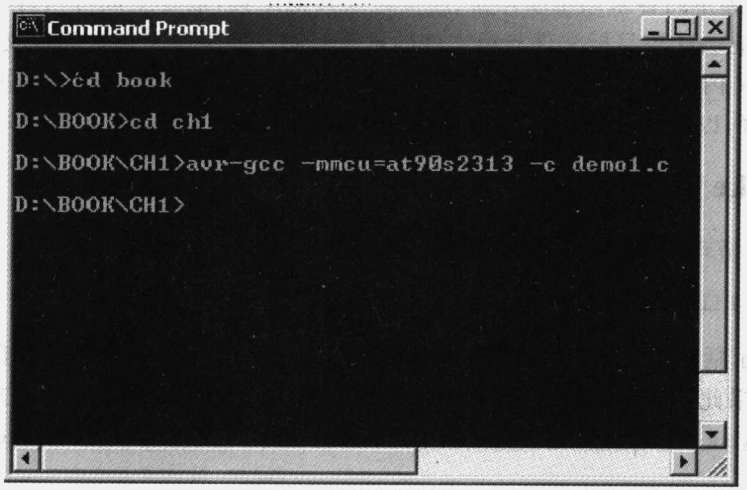


图 1-1 控制台窗口

GCC 把不同类型的数据分到不同的段落，相关程序存储器的段有. text 和. data，用选项-j 指定要提取的段。选项-O 用来指定输出格式，这里指定为 ihex (intel HEX file)。

至此得到了最终可以写入单片机 AT90S2313 FLASH 存储器的 demo1. hex 文件。用编程器将 demo1. hex 内容写入到单片机。PB0 引脚接一串联电阻 1 kΩ 的发光二极管(注意方向)后接到电源或地，上电复位后便可看到接在 PB0 口的 LED 不断地闪烁。

以上对一次编译过程的描述只是为了说明 GCC 编译一个 C 语言源程序的步骤，在实际应用中很少用这种方式编译每一个源程序和每一个更新后的程序，而是借助一个叫 make 的项目管理工具来进行编译操作。

1.3 用 makefile 管理项目

通常，一个编译器(泛指高级语言编译器、汇编器、连接器等)、项目管理器和文本编辑器构成一个完整的编程环境。

其中项目管理器负责为每一个源程序文件调用编译器，生成目标文件后用连接器将它们组合在一起生成可执行文件。例如，Keil μVision 进行编译时，将项目中的每一个源程序文件进行编译后生成对应的. obj 文件，然后将这些目标文件连接到一起生成可执行(例如 iHex 格式)文件。这是对编译过程粗略的描述，Keil μVision 掩盖了很多细节，使用过早期 C51 编译器的读者对此一定有所了解。

WinAVR 没有像 Keil μVision 那样集成项目管理器，所以需要写一个叫做 makefile 的文件来管理程序的编译连接。makefile 是一个脚本文件，一个标准的可执行文件 make. exe 负责

解析它，并根据脚本内容来调用编译器、连接器或其他工具，最终生成可执行代码文件。每次调用 make 时，它会比较目标文件与源文件的更新时间。如果源文件比目标文件还要新，则它会执行 makefile 内的相关命令，并更新目标文件；如果目标文件与源文件一样新，则它就跳开这个源文件的编译，以避免重复工作。这对于一个较大的工程来说是一种节省时间的有效方法。

1.3.1 make 的调用

make 指令格式如下：

```
make [-f filename] [names]
```

方括号表示括号里边的内容可以省略。其中 filename 代表 make 所使用的项目描述文件(makefile)，如果此项省略，则从当前目录下按下列顺序寻找默认的项目描述文件 GNUmakefile、makefile、Makefile(当然，在 Windows 下不区分大小写文件名，所以 makefile 与 Makefile 是相同的)。

names 指定目标名或宏名。若不指定目标名，则 make 命令总是把在 makefile 文件中遇到的第一个目标当作默认目标执行。

了解 makefile 脚本内容是非常有用的，但这不是必需的，因为 WinAVR 中一个小工具 mfile 可生成功能足够的 makefile 样本。通常只需按要求调整几个变量即可。如果对 makefile 不太感兴趣，则可以跳过下面的内容。

1.3.2 makefile 项目描述文件

1. 规　则

规则是 makefile 脚本核心内容。规则的书写格式如下：

```
target … : prerequisites …
command
…
```

make 命令引入了目标(target)的概念。target 可以是目标文件，也可以是标签(label)，make 将为生成或达到此目标而执行规则内的命令(command)。prerequisites 是目标生成的依赖文件。如果 prerequisites 比目标更新，则 command 将被执行，并生成最新的目标。这就是 makefile 的规则。

2. 目　标

make 命令必须处理至少一个目标，否则不会得出任何结果。正如在一个没有 makefile 文件的目录下敲入 make 一样，make 会输出以下的结果：

```
MAKE: * * * No targets specified and no makefile found.  Stop.
```


试读结束，需要全本PDF请购买 www.ertongbook.com