

BC3.1 软件开发工具使用简介

英创信息技术有限公司

2005 年 3 月

§ 1 概述

美国 Borland 公司的经典产品 BC3.1 是一套应用于 x86 系列 CPU 平台、支持 C/C++ 及汇编编程的集成开发调试软件工具。BC3.1 自推出以来，就以它卓越的编译性能及简单明了的操作界面获得了巨大的成功，深受世界各地广大开发工程师的喜爱，在当时的 PC/DOS 环境风靡一时。时至嵌入式系统的兴起，BC3.1 则成为 x86 系列嵌入式产品开发中的重要工具之一，如著名的 uCOS-II 多任务操作系统的基本版本就是采用 BC 来编译的。本文将针对基于英创嵌入式网络模块的应用程序的开发特点，对 BC3.1 的使用作一基本的使用介绍，以让广大的开发工程师能分享 BC3.1 为我们带来的高效与便捷。

嵌入式系统的应用程序开发一般由程序设计和调试两部分组成，本文主要涉及程序设计，而程序的调试则在我们提供的《Turbo Debugger 使用简介》中介绍。本文认为读者已掌握了用 C 语言进行程序设计的基本知识；对需要使用 C++ 的读者，则已掌握了 OOP 的基本概念；对需要在程序中嵌套汇编的读者，则已对 x86 的汇编有了基本的编写技能，因此本文对 C/C++ 及汇编不再作任何介绍，而主要介绍 BC 集成开发环境（IDE）的设置；工程文件（PRJ）的使用；在线帮助的使用；以及程序编译连接中的出错处理。

建议用户在工作盘上以网络模块名建立根目录，如 NetBox2、ETR100、ETR232i、ETR186 等等，再根据不同的应用或测试建立子目录，以便于程序代码管理。本手册采用 NetBox2 为例，但所介绍的 BC 使用方法是通用的。本手册所引用的示例 Step1 和 Step2，用户可从附带的开发光盘的“使用必读”目录中找到。

§ 2 启动 BC31 集成开发环境

在 BC31 正确安装后(安装的根目录为 C:\BC),BC 的所有可执行文件均存放在 BC\BIN 目录下。本文认为用户已把 BC\BIN 目录加入到了 AUTOEXEC.BAT 中的路径 (PATH) 定义中, 因此用户可在任意工作目录下启动 BC。

用户通常先打开 MSDOS 窗口, 并转换到自己编写的应用程序所在目录, 如 D:\NetBox2\Step1>, 然后执行操作: “bc”, 若客户是第一次运行 BC, 则界面如图 1 所示:

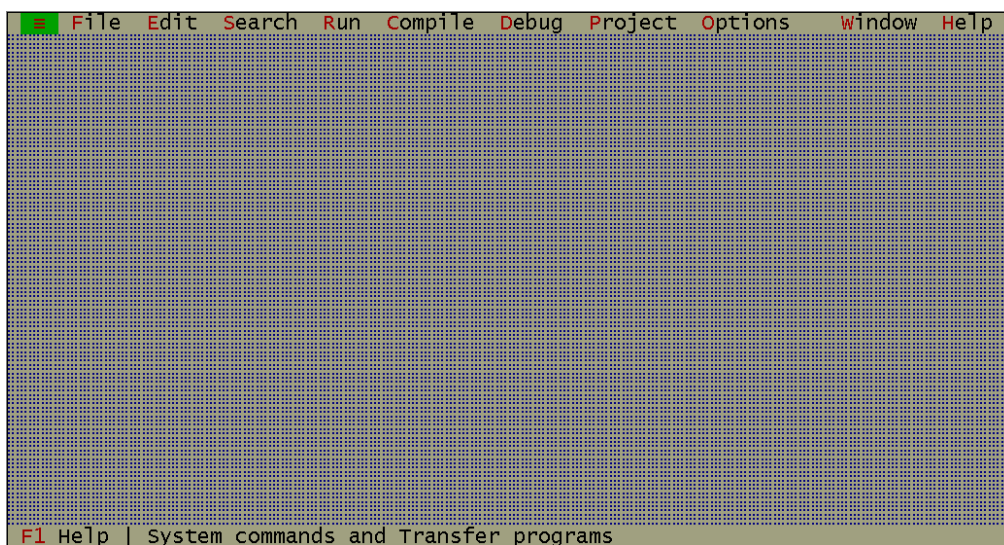


图 1 BC31 主界面

进入 BC 集成开发环境中后, 通常可按快捷键 **F3** 弹出打开文件对话框 (图 2)。BC31 的窗

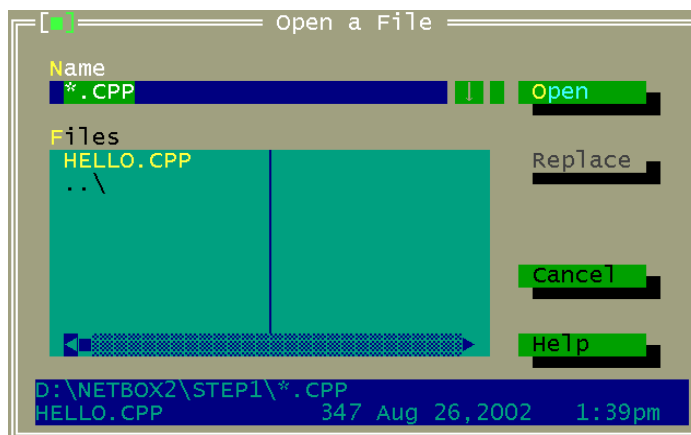


图 2 打开文件对话框

口或菜单中, 带红色或黄色的字母为关键字, 一般用 **Alt+<关键字>** 来激活该栏目, 用 **<Tab>** 键在栏目之间跳转。按上述方法选择文件 `hello.cpp` 并按 **F5** 键放大编辑窗口后, 屏幕上将显示如图 3 所示:

```

File Edit Search Run Compile Debug Project Options Window Help
HELLO.CPP
#include <stdio.h>
#include <conio.h>
#include <time.h>
#include <dos.h>

int main( )
{
    struct  time t;

    clrscr( );
    printf( "Hello!\n" );
    for( ; ; )
    {
        gettime(&t);
        printf("%2d:%02d:%02d\r", t.ti_hour, t.ti_min, t.ti_sec );
        if( kbhit( ) ) { getch( ); break; }
    }
    return 0;
}

1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```

图3 打开文件后的 BC31 主界面

图3中中顶上一行为BC主菜单；中间窗口为编辑区；最底下一行为快捷键提示行。用户可按 **Alt+<红色字母>**来打开各主菜单项，关闭菜单的操作均为按<Esc>键。退出BC集成环境则按 **Alt+X**。

常用的热键功能说明如下：

热键	功能
F1	获取BC的在线帮助菜单
F2	在BC编辑环境下，将编辑区中的文件存盘
F3	打开一个文件
Alt+F3	关闭已打开的CPP文件
F5	扩大编辑窗口到整个屏幕
Alt+F9	编译编辑窗口的CPP文件
F9	编译连接所有文件，并生成.EXE文件
Alt+X	退出BC编译环境

§3 BC31 的编译链接环境的设置

为了让BC31能编译出能在英创嵌入式模块上正确运行的应用程序代码，需对BC集成开发环境的相关参数做出相应的设置，主要在于以下几个方面：

编译路径的设置

在主菜单中使用 **Alt+O** 选项 (**O**ptions)，然后选择“**D**irectories”，将弹出“**D**irectories”

对话框，把 BC 软件所在目录设入 BC 集成开发环境的目录选项中，若 BC 安装在 C:\BC，则 Include 目录应设为 C:\BC\INCLUDE；而 Library 目录应设为：C:\BC\LIB。

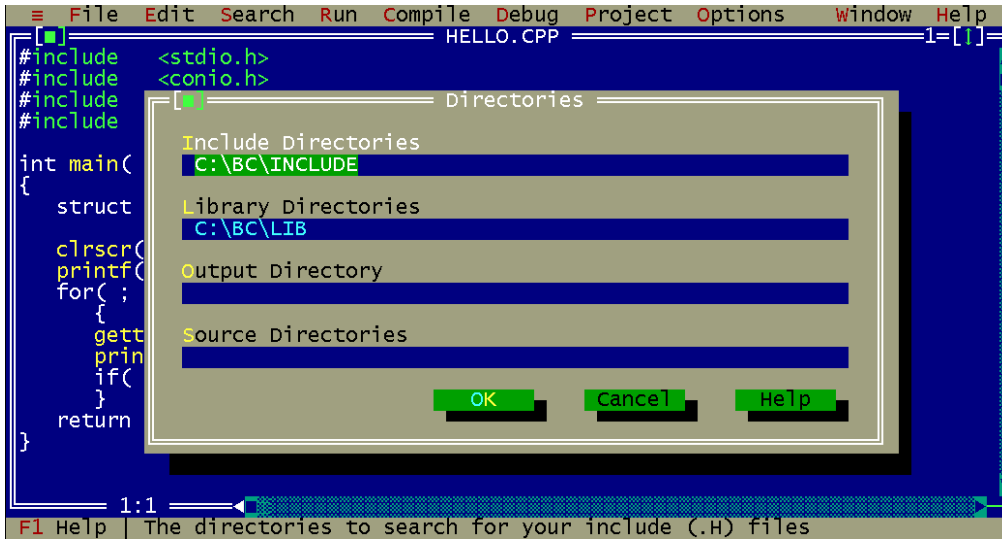


图 4 目录对话框

编译选项的设置

在使用英创嵌入式网络模块进行产品开发时，用户应将编译模式选择为 **Large** 模式。这是由于我们的 TCP/IP 库采用的是 **Large** 模式，因此用户在包含 TCP/IP 库的工程文件中，需要将编译模式设置为 **Large** 模式。如果用户采用了我们提供的 RTOS 库文件，还需要将编译模式设置为 **Huge** 模式。

设置编译模式的方法是：在主菜单中使用 **Alt+O** 选项 (**O**ptions)，然后选择“**C**ompiler”，如图 5 所示：

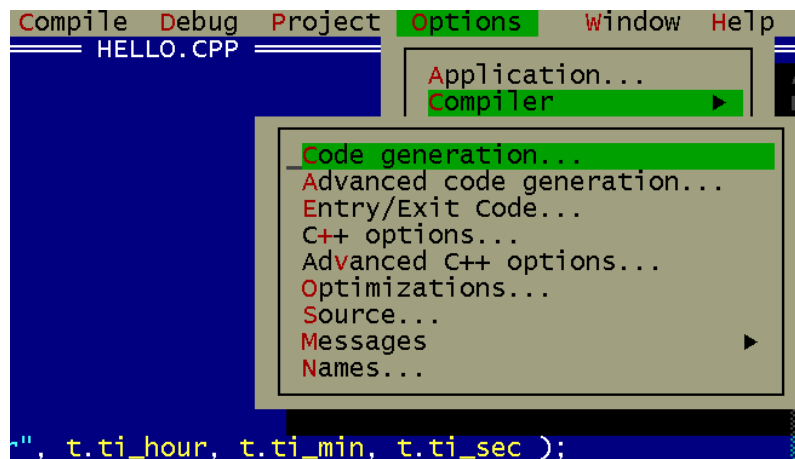


图 5 编译设置菜单

再选“**C**ode generation...”将出现如图 6 所示的代码生成对话框，直接用 **Alt+<黄色字母>**来设置各个选择项。最后按回车键 **<Enter>** 进行确认，对话框将自动关闭。

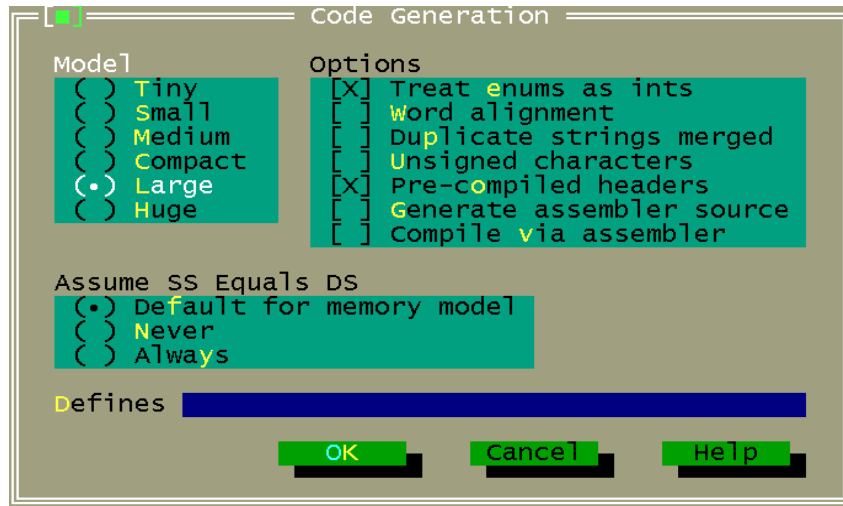


图 6 代码生成对话框

进一步地请按图 5 所示选择“Advanced code generation...”，打开高级代码生成对话框，并参照图 7 用 Alt+<黄色字母>来设置各选择项。

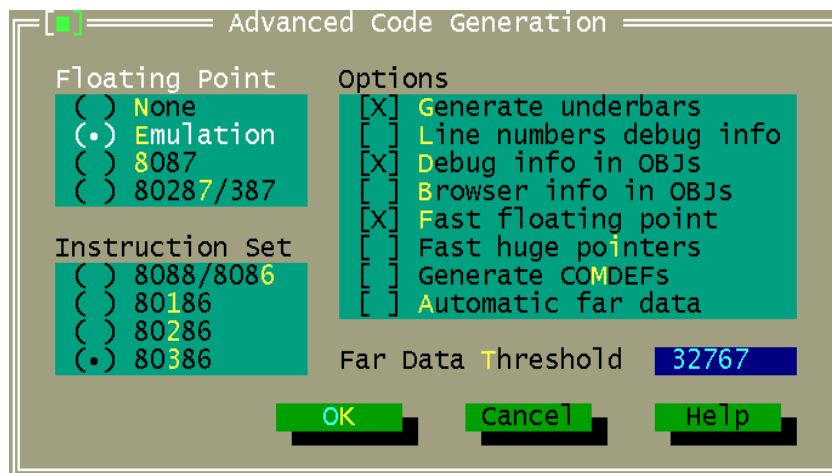


图 7 高级代码生成对话框

其中有 3 项内容用户需仔细确认设置，它们是：

浮点设置 --- 英创嵌入式网络模块产品无论是 386 系列还是 186 系列其 CPU 均不带有协处理器，CPU 只能通过仿真库来处理浮点数据，因此“Floating Point”项必须设置为“Emulation”模式。

指令设置 --- 对于英创 386 系列产品最好选择“80386”；对于英创 186 系列产品指令必须设置为“80186”。注意 BC31 的“Instruction Set”的缺省设置为“80386”，此时编译的程序会产生 186 CPU 无法识别的非法指令，因此不能在英创 186 系列产品上正常运行。

调试设置 --- 建议用户设置带调试信息的编译，这样可在 TD 中进行源码调试。

链接选项的设置

与编译选项的设置类似，选择“Options”->“Linker”->“Libraries...”，如图 8 所示。由于英创产品已不支持通常的 VGA 显示，所以无需连接图形库“Graphics library”。标准的 Run-time 库应设置为静态“Static”，注意其他库的选项都应设置为无“None”。

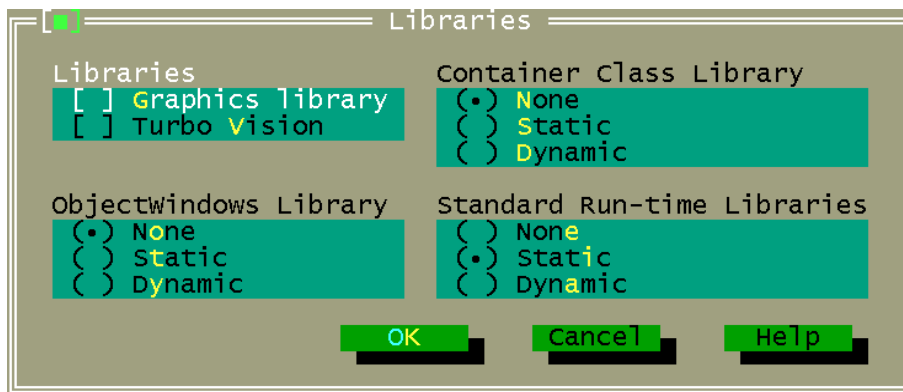


图 8 运行库选项设置对话框

§ 3. 管理工程文件（PRJ 文件）

工程文件可把多个程序模块方便地组合在一起进行编译连接，这样便于合理地安排程序结构，是设计专业程序的基本手段。在英创公司提供的软件编程的测试例程中大都是采用工程文件（PRJ）形式，这样便于将底层的驱动程序模块和实现应用功能程序模块分开，有利于程序的调试和维护。本节的内容主要是介绍两种方法来建立自己的工程文件：一是完全创建一个新的 PRJ 文件；二是利用已有的 PRJ 文件。为了更好地利用英创提供的软件例程，建议用户最好选择第二种方式。

本节我们以光盘“\使用必读\Step2”目录中的程序为例，建议读者在自己的工作盘中建立目录：D:\NetBox2\Step2>，并把光盘的内容拷贝到所建目录中。注意：从光盘中拷贝的文件其属性都是只读，需调用命令“attrib -R *.*”去掉只读属性。

创建一个新的 PRJ 文件

首先转到目录 D:\NetBox2\Step2>，调用命令“del *.prj”删除已有的工程文件。

进入 BC 后，按 Alt+P 打开 Project 菜单，选择 Open Project，弹出对话框如图 9 所示，并在“Open Project File”栏输入一个工程文件的名称“232demo.prj”。按下<Enter>键确认后，BC 会自动打开一个叫“Project: 232demo”的 Project 窗口，按 F5 键可将当前活

动窗口放大如图 10 所示的。

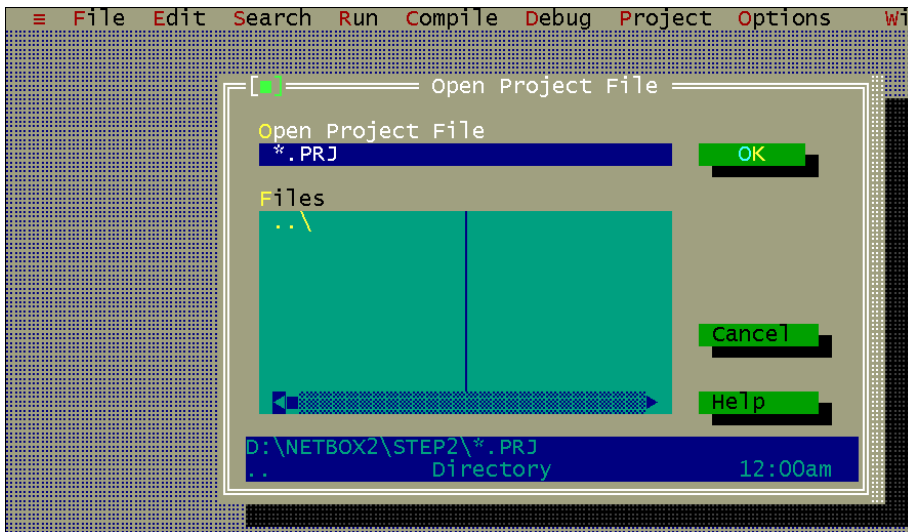


图 9 打开工程文件对话框

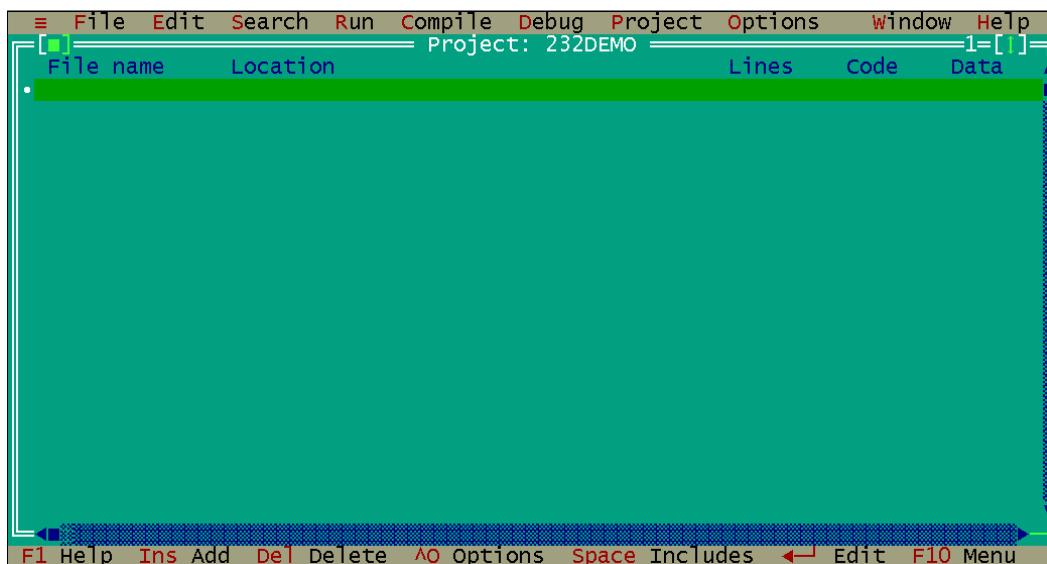


图 10 工程文件窗口

此时按下热键<Insert>或者选择菜单 **P**roject 下的 **A**dd Item 项，会弹出一个对话框，如图 11 所示。该对话框中列出当前目录下所有的 CPP 文件，在此基础上可选择地向工程文件中添加所需的 CPP 文件。用户可以修改“Name”栏的文件扩展名成“*.LIB”，按<Enter>确认后，就可向工程文件中添加 LIB 文件。

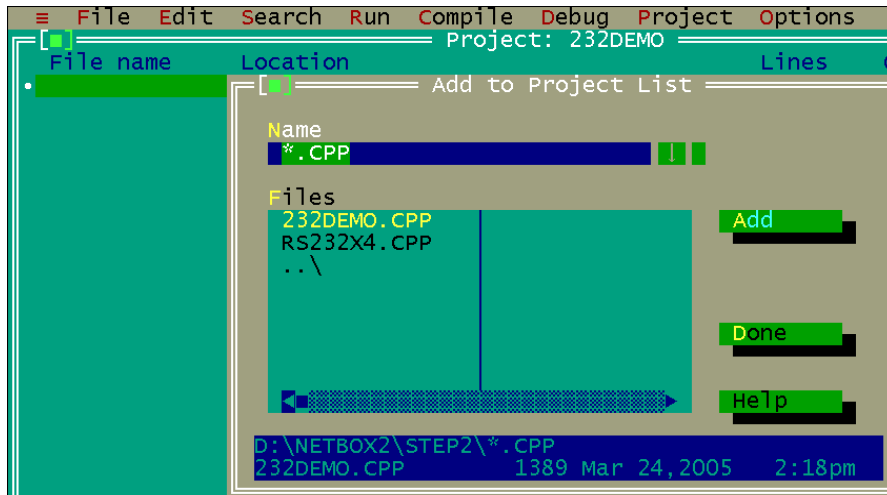


图 11 添加工程文件项目对话框

以 232DEMO.PRJ 为例，在 CPP 文件列表中，其中 RS232X4.CPP 模块是底层串口驱动程序，而 232DEMO.CPP 模块是包含了 C 语言入口函数 main（）的主控模块，该模块调用 RS232X4.CPP 模块中定义的串口通讯 API 函数，实现串口数据接收并显示并及时将接收到的数据向外发送的功能。因此需分别选择 RS232X4.CPP 和 232demo.CPP 添加到该 PRJ 文件中，如图 12 所示。建议用户把底层的模块放在 Project 窗口的上部，而把上层的控制模块放下面，带 main（）的主控模块（最好与 Project 同名）总是放在最下面，这样可提高 Project 的可读性。Project 窗口中的白色圆点标志是当前加入文件的所在位置。

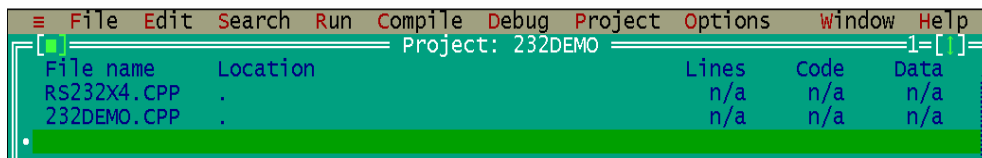


图 12 添加 CPP 文件到 Project 窗口

注意刚加入的 CPP 文件，其相关的编译信息“Lines”、“Code”、“Data”是没有的。

PRJ 文件中项目添加完毕后，直接按 **F9** 键或者选择菜单 **Compile** 下的 **Build all** 进行编译链接，编译链接成功将生成可运行 232demo.exe 文件，用户还可从编译链接的弹出窗口（图 13）中了解相关信息。若在编译链接时出现错误，也将在弹出窗口提示。

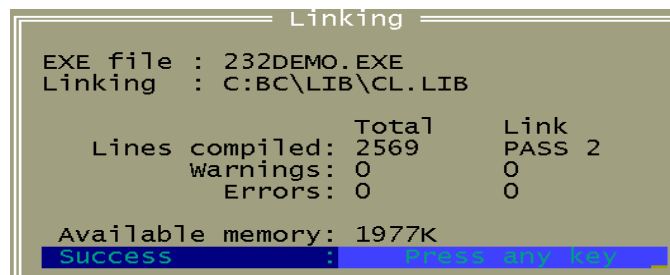
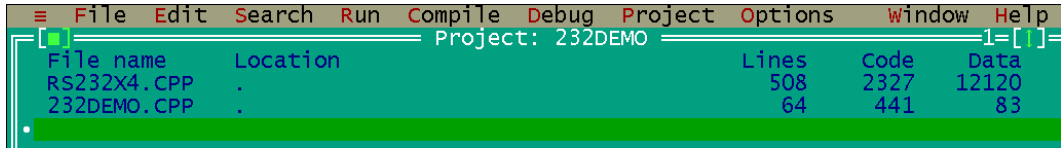


图 13 编译链接弹出窗口

按<Esc>键关闭编译链接弹出窗口，这时 Project 窗口将显示各个 CPP 模块的编译信息如图 14 所示。其中“Lines”列表示每个模块中程序代码的行数；“Code”列表示每个模块中程序代码的大小（以字节为单位）；“Data”列表示每个模块中定义的静态数据的大小（以字节为单位）。



File name	Location	Lines	Code	Data
RS232X4.CPP	.	508	2327	12120
232DEMO.CPP	.	64	441	83

图 13 编译链接弹出窗口

至此整个 Project 建立及编译链接的过程已完成。

利用已有的 PRJ 文件

仍以 Step2 目录下 232demo.prj 为基础，生成一个新的 PRJ 文件 23demo1.prj。在原 232demo.prj 文件中的 RS232X4.CPP 模块是底层串口驱动程序，可不作任何的改动，232demo.cpp 模块包含了 main（）函数，如果应用功能上需要作些调整，一般建议直接先将该文件存为另一个新文件，如存为 232demo1.cpp，然后再作修改，以实现新的应用需求。

通常是先打开 MSDOS 窗口，并转换到相应的目录下，以上面的 232DEMO.PRJ 为例，如 D:\NetBox2\Step2>，然后执行操作：“copy 232demo.prj 232demo1.prj”，

```
D:\NETBOX2\STEP2>copy 232demo.prj 232demo1.prj
1 file(s) copied.
D:\NETBOX2\STEP2>
```

图 13 拷贝生成一个新的 PRJ 文件

进入 BC 环境，操作 Alt+P -> Project -> Open Project，弹出打开工程文件对话框，通过按<Tab>键切换到 PRJ 文件列表窗口，然后用<↑>、<↓>键选择 232demo1.prj。

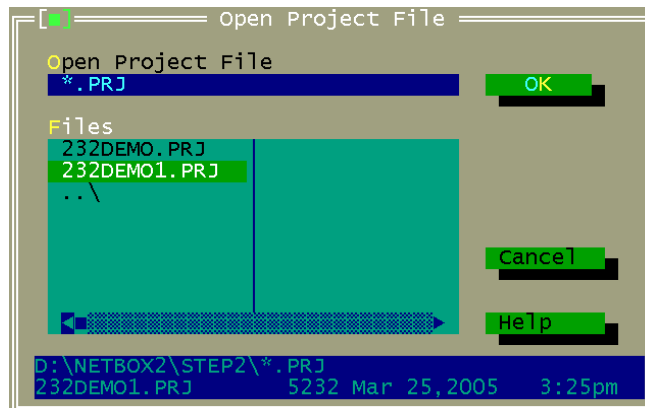


图 14 打开工程文件对话框

再按<Enter>键将该文件打开,这时 BC 将自动打开 Project 窗口如图 15 所示。

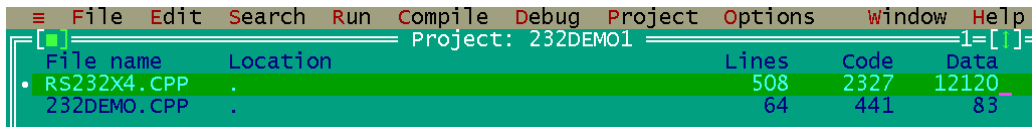


图 15 打开后的 Project 窗口

通过操作<↑>、<↓>键选择其中 232demo.cpp 后,按<Enter>键可打开该文件的内容,BC 将自动切换到 232demo.cpp 的编辑窗口,此时按 Alt+F, 并选择 Save as...

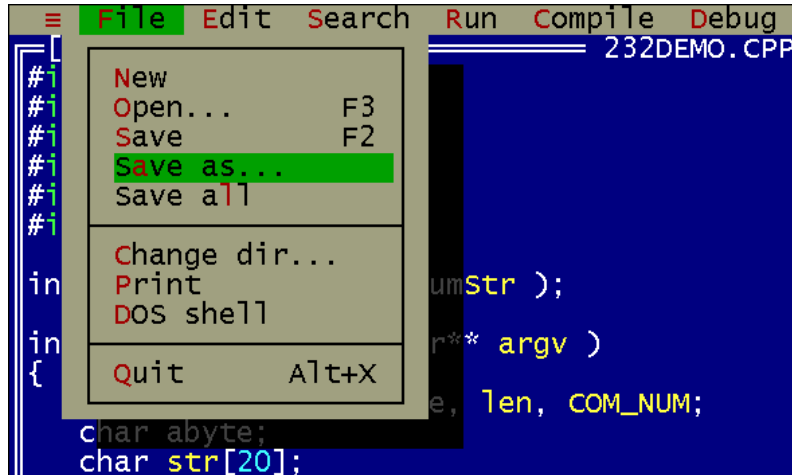


图 16 选择文件另存菜单项

由于该代码模块包含有 main () 函数,所以另存文件名最好与 Project 文件名一致。

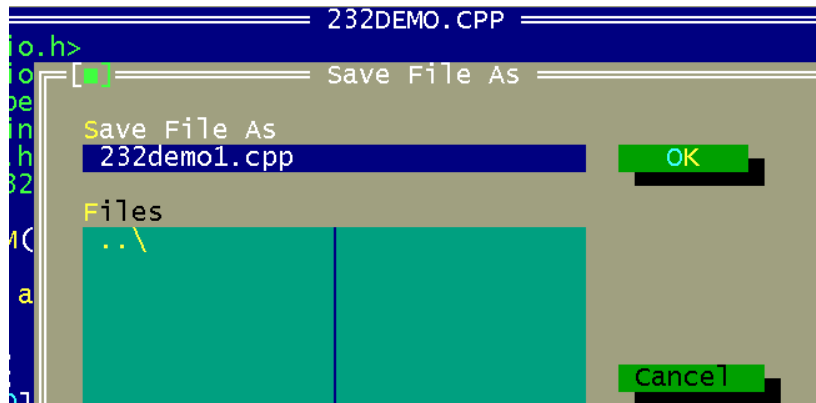


图 17 文件另存对话框

232demo.cpp 另存为 232demo1.cpp 后,需要在 232demo1.prj 中用 232demo1.cpp 来代替 232demo.cpp。首先选择 Alt+W 打开 Window 菜单(图 18),然后按快捷键 P 将活动窗口切换到 Project 窗口(图 15),此时可进行 PRJ 文件中各项目文件的删除或添加。

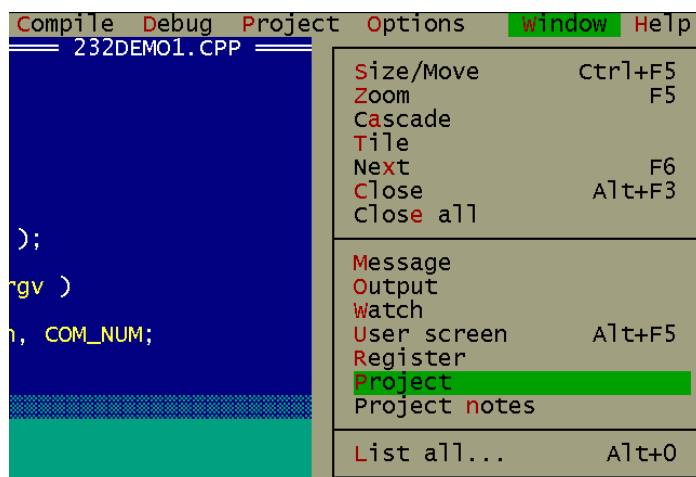


图 18 从 Window 窗口中激活 Project 窗

按热键 **<Delete>** 或者在菜单 **Project** 下的 **Delete Item** 删除工程文件不需要的文件 232demo.cpp 后，再采用前面已介绍的方法添加新的 CPP 文件 232demo1.cpp（图 19）。

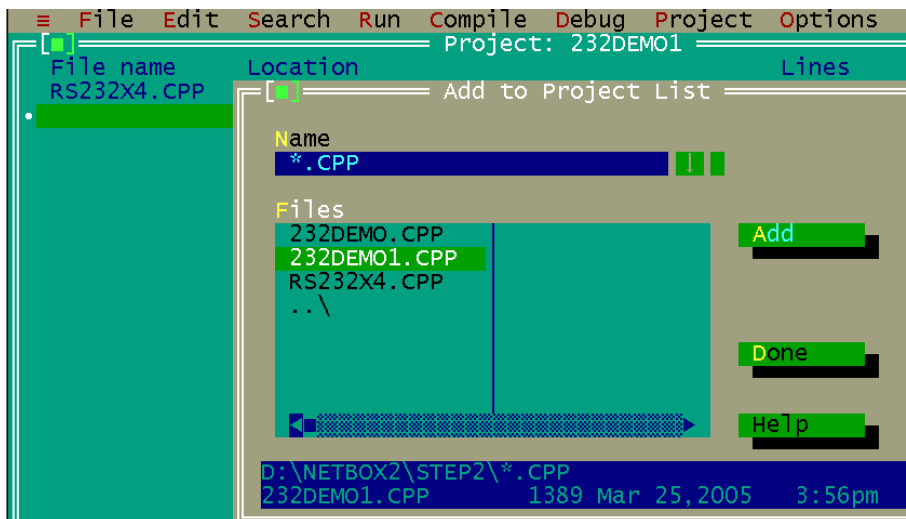


图 19 添加 CPP 文件到 Project 窗口

最后得到的 Project 窗口如图 20 所示：

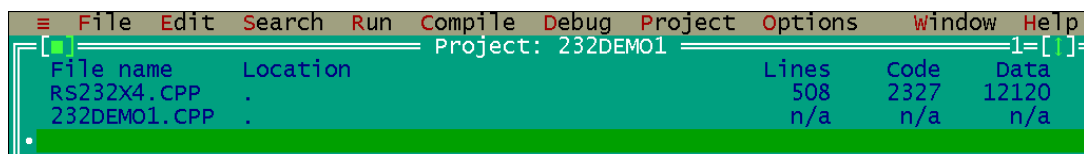


图 20 经过删减添加后的新 Project 窗口

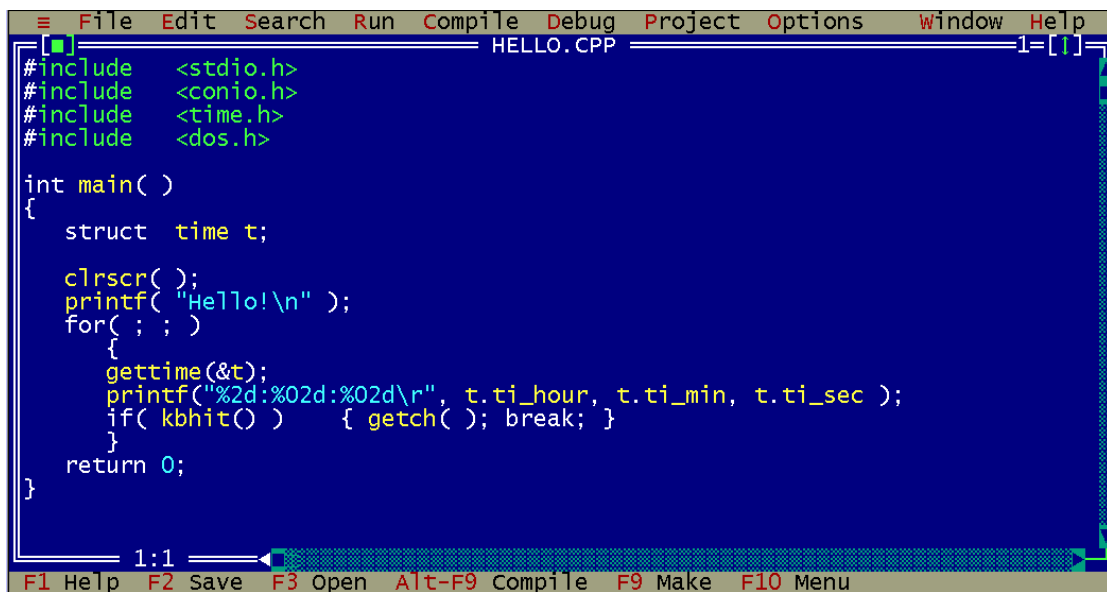
注意在图 20 中 RS232x4.cpp 是经过编译的文件，所以包含有文件的编译信息，而新添加文件 232demo1.cpp 由于还没有编译，所以就没有编译信息。此时可以打开 232demo1.cpp 进行修改，以实现新的应用功能。

在上面的介绍中，我们提到了通过按 **Alt+W**，然后再按快捷键 **P** 来切换到 **Project** 窗口

的方法，实际上是一个普遍适用的方法，特别在 **Project** 窗口被关闭时，可方便快捷地激活 **Project** 窗口。

§ 4. 如何获取在线帮助

BC 集成开发环境有非常简便有效的在线帮助功能，用户若能掌握其适用方法，定会大大加快应用程序的开发。下面以 **Step1** 目录中的 **Hello** 为例介绍用快捷键 **Ctrl+F1** 进入在线帮助的的具体操作方法。启动 BC 并打开 **Hello.cpp** 文件如图 21 所示：



```

File Edit Search Run Compile Debug Project Options Window Help
HELLO.CPP
#include <stdio.h>
#include <conio.h>
#include <time.h>
#include <dos.h>

int main( )
{
    struct time t;

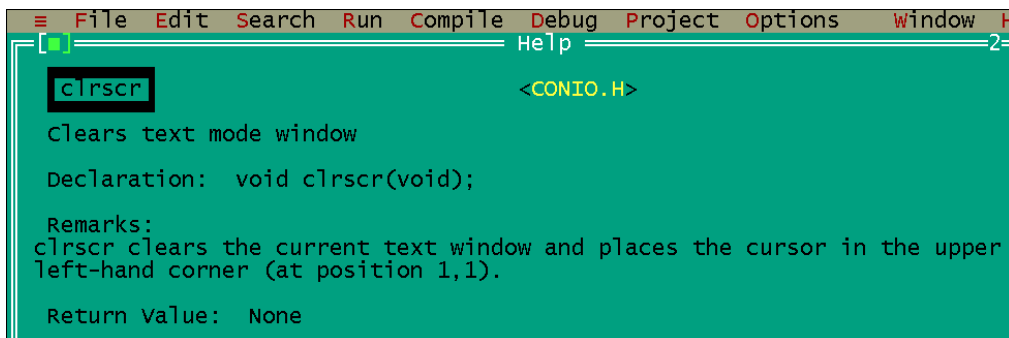
    clrscr( );
    printf( "Hello!\n" );
    for( ; ; )
    {
        gettime(&t);
        printf("%2d:%02d:%02d\r", t.ti_hour, t.ti_min, t.ti_sec );
        if( kbhit() ) { getch( ); break; }
    }
    return 0;
}

1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```

图 21 BC 编辑窗口

Ctrl+F1 是对编辑窗口中的光标指向的关键词提供帮助，这个关键词可以是函数名也可以是变量类型。举例来说，我们需要了解函数 **clrscr()** 的定义及适用方法，则首先把光标移至关键词“**clrscr**”下面，按 **Ctrl+F1**，BC 将立即弹出函数 **clrscr()** 的相关信息，



```

File Edit Search Run Compile Debug Project Options Window H
Help
clrscr <CONIO.H>
clears text mode window
Declaration: void clrscr(void);
Remarks:
clrscr clears the current text window and places the cursor in the upper
left-hand corner (at position 1,1).
Return Value: None

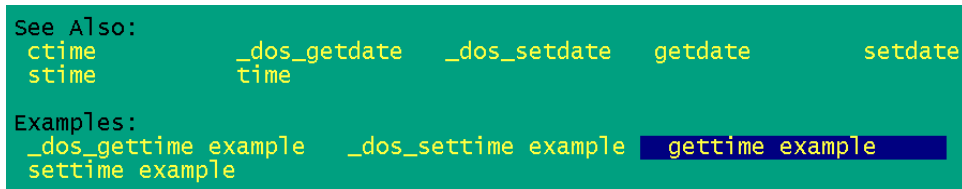
```

图 22 BC 的在线帮助窗口

在线帮助窗口会介绍函数的定义和功能，指出包含该函数的头文件名，在本例中，包含函数 `clrscr()` 的头文件名为 `<conio.h>`，因此在 `Hello.cpp` 中需 `include` 该头文件，才能正确调用 `clrscr()` 函数。除此之外，在线帮助窗口中还有一些用黄色标注的关键词，用 `<Tab>` 键可以在这些关键词间快速跳转，这些关键词包括与该函数相关的头文件、其他函数或变量、该函数的使用范例，其中的使用范例是最有使用价值的功能之一。我们以函数 `gettime()` 来进一步说明范例的使用。

按 `<Esc>` 键即关闭在线帮助窗口。

把光标移至函数 `gettime()` 下面，按 `Ctrl+F1` 打开在线帮助窗口，用 `<Tab>` 键选择黄色的关键词“`gettime example`”如图 23 所示：

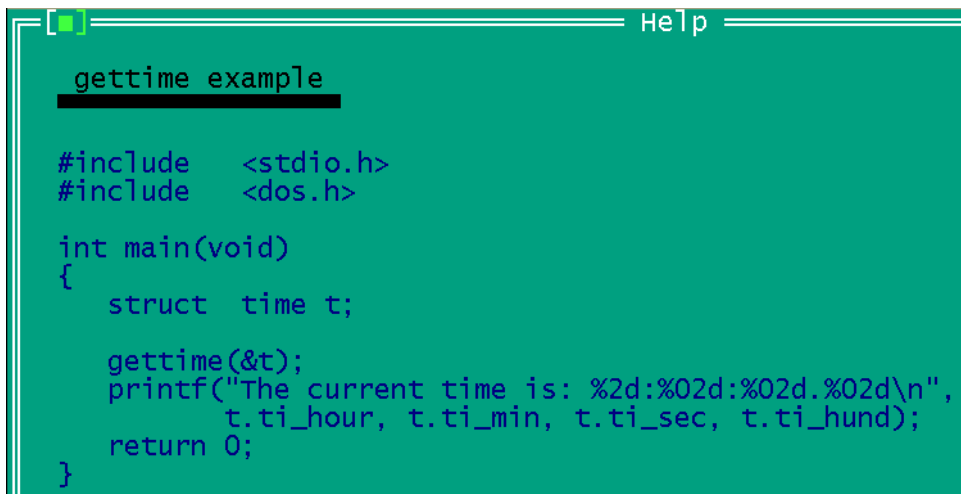


```
See Also:
ctime          _dos_getdate  _dos_setdate  getdate      setdate
stime          time

Examples:
_dos_gettime example  _dos_settime example  gettime example
settime example
```

图 23 函数 `gettime` 的部分在线帮助窗口

直接按 `<Enter>` 确认，BC 弹出范例程序窗口（图 24）。



```
gettime example

#include <stdio.h>
#include <dos.h>

int main(void)
{
    struct time t;

    gettime(&t);
    printf("The current time is: %2d:%02d:%02d.%02d\n",
          t.ti_hour, t.ti_min, t.ti_sec, t.ti_hund);
    return 0;
}
```

图 24 函数 `gettime` 的范例帮助窗口

用户可以通过标准的“拷贝-粘贴”的方法把需要的代码复制到用户自己的 `CPP` 文件中。仔细比较图 24 与图 21，可发现 `Hello.cpp` 其实就是在 `gettime example` 的基础上生成的。

当用户希望全面了解 BC 的运行库某一方面的情况时，可以按快捷键 `Shift+F1` 进入 BC 的在线帮助索引表。直接键入希望的关键词，如 `time`，则索引表会自动显示与 `time` 相关的内容如图 25 所示。

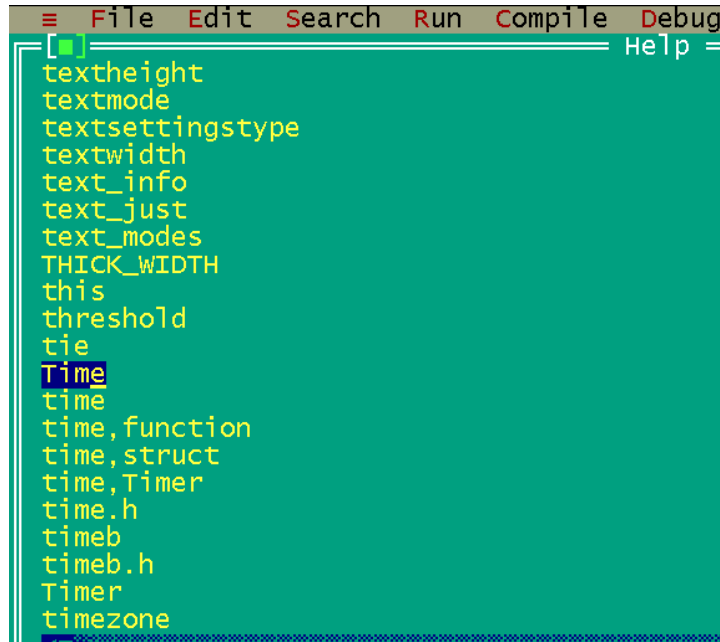


图 25 time 相关的 Help 索引表

这时用户可以方便的查看与 **time** 有关的内容，如“**time,struct**”、“**timezone**”等等。

§ 5. 结束语

美国的 Borland 公司是一家具有传奇色彩的软件开发公司，BC31 是当时公司里一群世界顶尖高手的杰作。我们希望通过本文的介绍以及在英创嵌入式网络模块上的实战操作，用户能感受体验 BC 的快捷与高效、分享大师们的智慧，因为智慧比知识更有力量。

附 1 BC 编程注意事项

编译时的常见错误

- (1) 数据类型错误。此类错误是初学者编程时的常见现象，下面是一些要引起注意的错误
 - a、所有变量和常量必须要加以说明。
 - b、变量只能赋给相同类型的数据。
 - c、不要用 0 除。这是一个灾难性的错误，它会导致程序失败，执行除法运算要特别小心。
- (2) 将函数后面的";"忘掉。此时错误提示色棒将停在该语句下的一行，并显示：

```
Statement missing ; in function <函数名>
```
- (3) 给宏指令如#include, #define 等语句尾加了";"号。
- (4) "{"和"}"、"("和")"、"/"和"/"不匹配。引时色棒将位于错误所在的行，并提示出有关丢掉括号的信息。
- (5) 没有用#include 指令说明头文件，错误信息提示有关该函数所使用的参数未定义。
- (6) 使用了 Borland C 保留关键字作为标识符，此时将提示定义了太多数据类型。
- (7) 将定义变量语句放在了执行语句后面。此时会提示语法错误。
- (8) 使用了未定义的变量，此时屏幕显示：

```
Undefined symbol '<变量名>' in function <函数名>
```
- (9) 将关系符"=="误用作赋值号"="。此时屏幕显示：

```
Lvalue required in function <函数名>
```
- (10) 定义的静态全局变量超过 64Kbyte，在 Large 模式下无法进行编译。

连接时的常见错误

- (1) Borland C 库函数名写错。这种情况下在连接时将会认为此函数是用户自定义函数。此时屏幕显示：

```
Undefined symbol '<函数名>' in <程序名>
```
- (2) 多个文件连接时，没有在"Project/Project name 中指定项目文件 (.PRJ 文件)，此时出现找不到函数的错误。

- (3) 子函数在说明和定义时类型不一致。
- (4) 程序调用的子函数没有定义。

运行时的常见错误

- (1) 路径名错误。在 MS-DOS 中，斜杠(/)表示一个目录名；而在 Borland C 中斜杠是个某个字符串的一个转义字符，这样，在用 Borland C 字符串给出一个路径名时应考虑"\\"的作用。例如，有这样一条语句

```
file=fopen("c:\new\tbc.dat", "rb");
```

目的是打开 C 盘中 NEW 目录中的 TBC.DAT 文件，但做不到。这里"\\"后面紧接的分别是"n"及"t"，"\n"及"\t"将被分别编译为换行及 tab 字符，DOS 将认为它是不正确的文件名而拒绝接受，因为文件名中不能和换行或 tab 字符。正确的写法应为

```
file=fopen("c:\\new\\tbc.dat", "rb");
```

- (2) 在对文件操作时，没有在使用完及时关闭打开的文件。
- (3) 用动态内存分配函数 malloc()或 calloc()分配的内存区使用完之后，未用 free()函数释放，会导致函数前几次调用正常，而后面调用时发生死机现象，不能返回操作系统。其原因是因为没用空间可供分配，而占用了操作系统在内存中的某些空间。
- (4) 使用的局部变量指针，但未进行初始化操作，会造成系统破坏。
- (5) 定义局部变量数组太大，超过 4Kbyte，会引起系统堆栈溢出。