# 一种高效的混合编程调试汇编程序的方法

童晓阳　（成都西南交通大学模拟中心　610031）

**摘要**：本文研究出一种高效率的调试汇编程序的方法，它利用 Turbo Debugger 3.1 作为调试工具，采取 C 语言与汇编程序混合编程的方法，解决了一般汇编语言调试过程中出现的问题，大大提高了调试汇编程序的效率。

**关键词**：汇编调试　混合编程　高效率

调试汇编程序，尤其是大中型汇编程序，是一件很困难的事情，汇编语言属于低级语言，人们在调试汇编程序时，常常感到汇编程序的数据和过程表示不直观，不容易把握它们的变化，更不能方便随意地改变变量的值，以便观察和检测出程序内部潜在的错误。针对这些问题，本文利用 Borland C++ 3.1 的实用工具 Turbo Debugger 3.1 采用 C 语言和汇编程序混合编程的方法，较好地解决了汇编调试的问题，提高了汇编程序的准确性和调试效率。

## 1. 混合编程调试程序方法的工作原理

常见的汇编调试工具有 Debug，Microsoft 公司的 Code view 4.1（简称 CV）和 Borland C++ 3.1 的实用工具 Turbo Debugger 3.1（简称 TD）等。Debug 是最早的汇编调试工具，它以命令行的形式运行，对于调试小程序方便，但对于大中型汇编程序，它所提供的信息相对就有限，功能也太弱，这里不采用。CV 和 TD 都是多窗口的集成式的调试环境，能同时在屏幕上以多个窗口显示各方面的调试信息。在窗口信息显示方面，都能显示诸如 register、CPU、stack、memory、watch 等，在程序运行方面，都能实现 trace、step、go to cursor、animate、breakpoints 等，通过比较，我们发现 TD 比 CV 具有更强的功能。CV 主窗口显示的是将标记符转变成相对逻辑地址的程序，TD 则显示的是编程者熟悉的源程序代码，标记符和变量名均未变。TD 优异的地方在于：对于每一种类型的窗口，都有一个本地菜单（Local menu），提供不同的功能。如前所述，调试程序时，最关心的是能否直观地看到源程序所定义的标志符，变量和函数，而且能根据调试需要改变它们，以便观察可能出现的变化情况。TD 主菜单的 View 项的子菜单项 Variables 做到了这点，激活 Variables 项所产生的窗口展现出源程序当前模块定义的所有变量名，选中一个变量，按 ALT + F10（或击右键）弹出如图 1 所示的菜单，通过它观察和改变变量的值，这正是我们所需要的。因此选中 TD 作为混合编程调试汇编程序的工具。

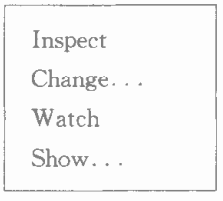


图 1　Variables 的弹出菜单

本文混合编程调试汇编方法的思路是这样的：由于一般的汇编程序不能直接用 TD 调试，从混合编程的角度出发，借助调试工具 TD，将 C 语言和汇编程序制作成一个工程文件，用 BC3.1 的编辑器统一编译连接，利用 C 语言的输出函数辅助调试，汇编程序需根据一定的规则稍加改造，可利用 C 语言程序为汇编程序中的公共变量提供数据。另一方面，从汇编程序中调用 C 和 C++ 子程序，可向屏幕输出，使调试者能够更直观地观察变量值；向文件输出，使调试者能够以后全面系统地观察和分析程序的问题，采用混合编程的方法，借助功能强大的调试工具 TD，使调试者能轻松地找到需要的模块和变量，观察和改变它的值，改变通用寄存器和 PSW 程序状态字寄存器的值，从而设置出各种调试条件，观察程序的变化，才能发现问题所在。

## 2. 混合编程技术

混合编程包括用 C 程序调用汇编语言子程序和汇编程序调用 C 语言子程序两方面。从 BC 3.1 调用汇编程序，需要遵循一定的步骤和规则，在改造原汇编语言子程序时，须遵循以下的规则：

(1)保证连接器可得到所需的信息；

(2)保证文件格式对应 C 程序所用的内存模式。一般来说，汇编语言模块由三个部分组成：代码、初始化数据和非初始化数据。每一种类型的信息使用特定名组织到其自己的段中，所使用的名字依赖于 C 程序所用的内存模式。汇编程序使用由 .MODEL 指定的内存模式所对应的缺省段名，如果希望不使用缺省段名，汇编程序的标志符 code, data 和 dseg 有特定的置换，由表 1 所示。

表 1　标志符置换和内存模式

| 模式 | 替代标志符 | 代码和数据指针 |
|---|---|---|
| Tiny, Small | code = -TEXT<br>data = -DATA<br>dseg = DGROUP | Code: DW-TEXT. xxx<br>Data: DE DGROUP: xxx |
| Compact | code = -TEXT<br>data = -DATA<br>dseg = DGROUP | Code: DW-TEXT: xxx<br>Data: DD DGROUP: xxx |
| Medium | code = filname-TEXT<br>data = -DATA<br>dseg = DGROUP | Code: DDxxx<br>Data: DD DGROUP: xxx |
| Large | code = filename-TEXT<br>data = -DATA<br>dseg = DGROUP | Code: DDxxx<br>Data: DD DGROUP: xxx |
| Huge | code = filename-TEXT<br>data = filename DATA | Code: DDxxx<br>Data: DDxxx |

建立了一个模块后，Borland C + + 程序需要知道它能够调用哪些函数以及能够引用哪些变量，同样，从汇编语言子程序内部调用 Borland C + + 函数，或希望能够引用 Borland C + + 程序中声明的变量。

在进行这些调用时，需要了解 Borland C + + 编译器和连接器的有关内容，当声明一个外部标志符时，编译器在把该标志符保存在目标模块中之前将自动在其前面增加一个下划线(-)，这意味着应该在希望从 C 程序引用的汇编语言模块中的标志符前面加上一个下划线，若源文件中的任何 asm 代码引用了任何 C 标志符(数据和函数)，则这些标志符必须以下划线打头，要使标志符在汇编语言模块外是可见的，需要把它们声明为 PUBLIC。

从汇编语言子程序中调用 C 和 C + + 子程序时，用 EXTRN 语句可使汇编语言模块引用 Borland C + + 程序中声明的函数和变量，对于函数引用，用以下语句声明：

EXTRN vname: size

这里的 vname 是函数名，size 是变量尺寸(near 或 far)。

对于数据引用，可在数据段放置适当的 EXTRN 语句，采用下列格式：

EXTRN vname: size

这里的 vname 是变量名，size 是变量尺寸(BYTE, WORD, DWORD, QWORD 等)

如果调用的 C 和 C + + 子程序有参数时，需将所需参数压入堆栈，然后调用子程序，当它返回时，如果使用 C 调用规范，需要把栈清除。Borland C + + 传递参数有两种方法，缺省情况下，使用 C 调用规范，当该函数被调用时，参数从右到左压栈，然后返回地址被压栈，清除栈内容。另一种方法是标准 Pascal 参数传递方法，参数压栈顺序则相反，并且函数返回时不需清除栈内容。

3. **调试方法的实现和实例说明**

搞清了混合编程调试汇编程序的工作原理，下面简要地叙述它的实现步骤：

(1)先将汇编源程序备份，再将备份后的源程序按照混合编程遵循的规则加以改造，编写 C 语言主程序和所需的 C 子程序；

(2)在 BC 3.1 编辑器中制作工程文件，将 C 程序和汇编程序包括在内，进入主菜单项 Options 的 Transfer，设置汇编编译器 TASM 的命令行参数及其他参数；

(3)编译连接好此工程，保证无错，若有错则及时修改；

(4)选用主菜单的≡(system)中的工具 Turbo Debugger，进入 TD 的调试环境进行调试，若发现错误，则返回 BC 3.1 编辑器重新编辑后，再执行步骤(3)；

(5)调试成功后，对照改造的汇编程序，修改备份前的源程序。

下面举一实例来说明这种调试方法的实现步骤。

C 语言程序：

```
#include<stdio.h>
#include<string.h>
extern void far opercmd(void);//被调试的汇编程序
extern unsigned char cmd-buf[10][81]//汇编程序中的变量
void out-put(int i);
void main()
{ // 1: type:1, nodes:G1, X1, G2, D29F, X3, G4
   strcpy (cmd-buf [0],"/001G/001X/001G/002D/035FX/003G/004");
```

```
    printf("/ncmd-buf[0]: %s/n", cmd-buf[0]);
    opercmd();      //调用汇编程序
    printf("/ncmd-buf[0]: %s/n", cmd-buf[0]);}
void out-put(int i)
{FILE * fp;
    printf("/ncmd-buf[%d]: %s/n", i, cmd-buf[i]);//
向屏幕输出
    if(fp = fopen("test.dat","a")) = = NULL)//向文
件输出
        {printf("cannot open file/n"); return;}
    if(fwrite(&cmd-buf[i], 81, 1, fp)!  = 1) printf("
file write error/n");
    fclose(fp);}
```

改造后的汇编程序：

```
.MODEL SMALL
EXTRN -out-put:near //引用 C 语言函数
dseg Group -DATA, -BSS
-DATA segment
  PUBLIC -cmd-buf //声明变量，可被 C 程序使用
  element struc
    type db 0
    nodes db 80 dup(0)
  element ends
  -cmd-buf label byte
  buffer element 10 dup(□)
    i db 0
-DATA ends
-BSS segment
    ......
-BSS ends
-TEXT segment
    assume cs:-TEXT, ds:dseg
-opercmd proc far
    mov ax, dseg
    mov ds, ax
    ......
    mov i, 1
    mov ax, i //以 C 调用规范调用 C 子程序 out-put
(int)，参数是 i
    push ax
    call -out-put
    add sp, 2 //清除栈
    ......
-opercmd endp
-TEXT ends
    end
```

4.总结

使用本文介绍的这种调试汇编的方法进行调试工作，大大地减少了调试工作的枯燥和繁琐程度，增加了直观性，提高了调试的准确性和效率。它已成功运用于南宁机务段大型汇编语言控制软件的开发和调试，实践证明，确实起到了良好的效果。混合编程调试汇编程序不失为一种高效的调试汇编程序的方法。

参考文献

[1] 张昆藏，微型机(PC 系列)系统功能调用教程，清华大学出版社，1992

[2] Borland C++3.1 程序员指南，电子工业出版社，1992

(来稿时间：1997 年 8 月)