HOPE

Turbo系列语言混合编程

- Turbo Prolog 与其它语言的接口
- Turbo C与其它语言的接口
- Turbo Pascal与其它语言的接口
- Turbo Basic与其它语言的接口
- 混合编程序的调试
- 混合编程的参考资料

中国科学院希望高级电脑技术公司

Turbo 系列语言混合编程

李 文 编

中国科学院希望高级电脑公司

序 言

Borland 公司推出众多风靡世界的 Turbo 系列语言，Turbo Pascal、Turbo C、Turbo Prolog、Turbo Basic、Turbo C++。这些年轻的语言以其优良的性能和用户界面，很快获得世界各地用户的欢迎。Borland 率先使用的集成环境、联机帮助与热键驱动已当今软件的用户界面的标准。独有的内部调试器使编程和调试的效率倍增，开发周期骤缩。此外，Borland 辅助开发实用程序 Turbo Debugger、Turbo Profiler、MAKE、TLINK、TLIB、TCREF、GREP、TOUCH、OBJXREF 和 TC(P)HELP 等，更使 Turbo 系列语言大放光彩。形成了编程(集成环境)、调试(内部源级调试器和 Turbo Debugger)、运行(集成环境中模拟)、剖视(Turbo Profiler)和工程管理(MAKE、Project 性能)一体化的良好环境。

Turbo 系列语言都有其解决问题的方向，如果综合各自的优点，进行混合编程，那么编程水平就将更上一层楼。

现在世界软件市场异彩纷呈，Microsoft 公司开发的语言 Microsoft C、QuickC、Microsoft Pascal、Quick Pascal、Microsoft Fortran 和 Microsoft Basic、Quick Basic 也很受用户欢迎。如果能超越 Borland 和 Microsoft 公司的障碍，把两者的优点结合在一起编程，那将无往而不胜。

Turbo C、Turbo Pascal、Turbo Prolog 等高级语言与汇编语言接口能直接使用 BIOS、DOS 的功能、直接对串行口、视频、游戏棒与鼠标等硬件进行存取、能进行内存驻留程序、设备驱动程序编程，提高了处理的速度(特别是图形处理速度)，扩展了控制。

高级语言之间的接口能充分发挥各种语言的独特优势，使编程更具灵活性；高级语言与数据库管理语言的接口能减少繁琐的数据项管理(数据项插入、删除、检索、排序)；高级语言与 BIOS 和 DOS 的接口扩展了高级语言的低级功能，减少使用汇编带来的繁琐的负担。

顾名思义，接口就是联络、搭起一座进行资源共享的桥梁。

接口的要素在于参数传递协议、函数返回协议、寄存器协议、数据的内部格式的差异、不同编译器生成的 OBJ 的兼容性(特别是相应的汇编语言的兼容性)、生成的 OBJ 文件所用的汇编级的选项的差别、启动代码的主次、重复的屏幕输入/输出和文件操作等相同功能的扬弃和内存管理的处理等。

本书针对以上难题进行探讨，介绍了混合编程的基础，混合编程的技巧，混合编程的调试，混合编程的工程管理等。虽然如此，但是接口论属于高级编程的领域，加上此专题的资料不多，因此本书只是结合长期经验的一次尝试。

编者

目　录

第一部分　Turbo Prolog 与其它语言的接口

Turbo Prolog 是基于逻辑程序设计的语言。从 1.0 版、1.1 版到 2.0 版，已有了很大的发展，成为开发专家系统等知识库系统的有力工具。

尽管 Turbo Prolog 在许多方面是一个非常好的工具，但是仍然需要使用其它语言。例如用 Pascal 或 Fortran 语言很容易完成数值积分运算，用汇编语言实现中断处理及低层操作更好。另外，如果用其它语言开发的程序已经解决了某些问题，这些工作不应被抛弃和否定。因此，Turbo Prolog 允许 Turbo Prolog 程序与其它语言连接。下面将介绍 Turbo Prolog 与其它语言的接口约定、Turbo Prolog 与 Turbo C 的接口、Turbo Prolog 与 MS-Fortran 4.0 的接口、Turbo Prolog 与汇编语言的接口。此外，还介绍了 Turbo Prolog 访问 dBASE Ⅲ 文件的方法，以及 Turbo Prolog 与 DOS 系统级的接口谓词。

第一章　Turbo Prolog 与其它语言接口的约定

在调用其它语言写的子程序和函数前，需要在 Turbo Prolog 中把它们说明成外部谓词，还应了解正确的调用约定和参数压栈顺序，以及如何命名外部谓词的不同流变体。下面三节分别叙述这三个问题。

注意：当提到 Turbo Prolog 时都是指 1.0 或更高版本。

§ 1.1　声明外部谓词

Turbo Prolog 允许通过使用一个全局谓词(global predicates)声明与其他语言进行接口。在声明后面附加一个语言说明，以便使 Turbo Prolog 能知道这个全局谓词是用哪一种语言实现的。

```
global predicates
    add(integer,integer,integer) - (i,i,o),(i,i,o) language asm
    scanner(string, token) - (i,o) language Pascal
```

Turbo Prolog 使接口语言明确化能简化活动记录和参数格式、调用和返回约定、段定义、链接和初始化等问题。

§ 1.2　调用约定和参数压栈顺序

8086 系列处理器使程序员能够对近子例程调用和远子例程调用进行选择。Turbo Prolog 创建大存贮模式程序，并要求所有对子例程的调用和从子例程返回都是远调用。

Turbo Prolog 支持多种调用约定，包括 C、Pascal 和 Assembler 等等。当使用 C 调用约定与子例程接口时，参数以反序压入堆栈，返回后栈指针自动调整；当与其他语言接口时，参数以正常次序入栈，被调函数负责从栈中取走参数。

在许多 8086 系列的语言编译程序中，可以选择 16 位指针或 32 位指针，16 位指针指

向的是默认段，而 Turbo Prolog 总是使用 32 位指针来访问所有内存。

Turbo Prolog 的类型以下列方式实现:

整型	2字节
实型	8字节(IEEE格式)
字符型	1字节(压入堆栈时用2字节)
字符串型	4字节　双字指针指向一个以NULL结尾的串
符号型	4字节　双字指针指向一个记录。

输出参数以指向某地址的 32 位指针形式压入堆栈，返回值必须赋给指针指向的单元。对输入参数，其值直接入栈，并且参数的大小取决于它的类型。

§1.3　命名约定

在 Turbo Prolog 中，一个谓词可以有多个类型的变体和多个输入输出流变体，每个类型和流变体有其各自的子程序。为了调用这些不同过程，必须为每一过程赋以唯一的名字，这是通过从 0 开始对具有相同谓词名的不同子程序向上进行编号来实现的。

例如，给出如下声明:

```
global predicates
    add(integer, integer, integer) - (i,i,o),(i,i,i) language asm
```

第一个变体（流模式为(i,i,o)）被命名为 add_0，第二个变体（流模式为(i,i,i)）命名为 add_1。

Turbo Prolog 还允许程序员为全局谓词声明一个明确的名字，这是通过在声明后跟"as public name"来实现的。下例中，全局谓词 Pred 将用 my_pred 而不是 pred_0 来声明:

```
global predicates
    pred(integer, integer) - (i,o) language asm as "my_pred"
```

这种方法在用户命名只有一种流模式的谓词时采用。如果存在多种流模式，则用户只能为每个变体声明一个名字。以 add 谓词为例，谓词定义可能如下:

```
global predicats
    add(integer,integer,integer) - (i,i,o) language asm as "doadd"
    add(integer,integer,integer) - (i,i,i) language asm as "add_check"
```

第一个变体（流模式为(i,i,o)）被命名为 doadd，第二个变体（流模式为(i,i,i)）被命名为 add_check。注意这种命名方法需要分别对各变体进行声明。

第二章　Turbo Prolog 与 Turbo C 的接口

§2.1 Turbo Prolog 调用 Turbo C 过程

§2.1.1 说明外部谓词

在 Turbo Prolog 主模块中，应说明被调用的 C 函数为全局谓词。下列的 Turbo Prolog 程序说明了如何访问一个用 Turbo C 编写的过程 double。可用如下的 Turbo Prolog 语句调用该过程：

```
double(MyInputInteger, MyOutVar)
```

在调用前 MyInputInteger 约束为一个整数值，调用后 MyOutVar 被约束为该值的两倍。

```
global predicates
    double(integer,integer) - (i,o) language C
goal
    double(5,N),
    write("5 doubled is ",N).
```

在包含调用 double 的 Turbo Prolog 程序中，必须在全局谓词段中指明实现该过程的语言。

§2.1.2 建立 C 函数源程序

在 C 函数源程序中，必须按 Turbo Prolog 命名约定为 C 函数命名，即 C 函数名就是 Prolog 的谓词名紧接一下划线和与一流模式对应的整数。

对于上例的 double 有如下的 C 程序：

```
void double_0(int in, int *out)
{
    *out = in + in;
}
```

§2.1.3 Turbo C 编译选项和连接

为了连接 Turbo Prolog 与 Turbo C 模块，必须带有下列选择项编译 C 源程序

(1) 用 TC.EXE 编译：

<1> Options/Compiler/Model/Large

<2> Options/Compiler/Optimization/Jump optimization ... On

<3> Options/Compiler/Code generation/Generate underbars ... Off

(2) 用 TCC.EXE 编译：

```
-ml -o -u- -z -c
```

(3) 连接程序模块：

Turbo Prolog 和 Turbo C 应按下列通用命令行连接：

tlink init<pOBJS><cOBJS><.sym>,[exe],[map],[usr]+prolog[+emu+mathl+cl]

该命令行各个参数的意义如下表所示：

参 数	功 能
tlink	调用TLINK，即Borland的Turbo连接程序
init	Turbo Prolog的初始化文件
pOBJS	Turbo Prolog .OBJ模块
cOBJS	Turbo C .OBJ模块
.sym	Turbo Prolog的符号文件须为第一个逗号前的最后一个文件
exe	（可选）可执行文件名
map	（可选）map文件名
usr	（可选）用户库文件列表
prolog	指PROLOG.LIB
emu	（可选）C浮点仿真库EMU.LIB
mathl	（可选）C大模式数学库MATHL.LIB
cl	（可选）C大模式库CL.LIB

假设 PDOUB.PRO 已经编译成.OBJ，连接 PDOUB.PRO 和 CDOUB.C 的正确命令为：

```
tcc -ml -o -u- -r- -c cdoub
tlink init pdoub cdoub pdoub.sym,pdoub,,prolog
```

如不用 TLINK，可创建一个工程定义文件 CPROJ.PRJ，在工程文件中给出两个模块名：

```
pdoub
cdoub.obj
```

注：需给 C 目标文件一个扩展名，如果在 C 文件名后不加扩展名，Turbo Prolog 的编译程序将给出错误信息，系统不能成功地进行连接。

创建工程定义文件后要做的是将 Prolog 模块说明为工程的一部分，如果 C 模块已经编译成 CDOUB.OBJ，只要选择 Compile/EXE file(程序 PDOUB.PRO 必须已装入 Turbo Prolog 编辑器中)，Turbo Prolog 系统将自动完成所有连接。如果需要用到外部库文件，可以在 O/L/Libraries 菜单项中指明。

```
project "cproj"
global predicates
        double(integer, integer) - (i,o) language c
goal
        double(5,N),
        write("5 doubled is ", N).
```

§2.1.4 动态存贮分配