第12卷 第4期 郑州工学院学报 Vol.12 No.4
1991年 12 月 Journal of Zhengzhou Institute of Technology Dec. 1991

参数传递过程中不相容性的研究*

王文义

(计算机与自动化系)

摘 要: 本文针对C语言与PASCAL语言不同的数据类型定义和存贮管理方式，详尽的探讨了因C编译器功能限制而引起它们之间在参数传递时的不相容性及解决办法，为系统应用程序的高效实现奠定了基础.

关键词: 参数传递，不相容性，UNIX系统

中国图书分类号: TP312

C语言和PASCAL语言是当今最为流行的两种面向系统的程序设计语言，前者高度灵活，后者结构严谨．以C语言为例，它与UNIX系统可说是一对孪生兄弟，它们之间联系紧密、相辅相成．在UNIX系统的13000多行源程序中，有12000行左右是用C语言书写的，只有约1000行是面向硬件的汇编程序．此外，C语言还实现了描述其自身的编译器，C编译器近10000行程序几乎全部由C语言组成．C语言的独特之处还在于它不属于通常所说的"高级语言"，这是由于它可对硬件寄存器进行存取，且又不象汇编语言、机器语言那样涉及具体地址，故有人形象的称它为"中级语言"．C语言非常适应于自顶向下的开发顺序、程序模块化、控制流结构化与数据结构系统化的模式规定，因此它符合现代软件工程的要求并与现代程序设计风格相吻合，受到了人们的青睐，与Ada和LISP语言一起被称为计算机的现代三大语言．

任何事情都具有相对性．正是由于C语言高度的灵活性，因此若从安全性及可靠性的角度考虑，它就要比PASCAL语言逊色些．之所以出现这种情况，是由它们的先天条件决定的，因为PASCAL语言有完整而精确的语法定义，而C语言则只有说明性的定义．

综上所述，对C语言和PASCAL两种语言，在某些系统应用程序的实施设计中，若能取长补短、各取所需的把二者结合起来，就会收到更为理想的效果．但由于两种语言在参数传递中存在着不相容的情况，因此上述想法的实现并非容易．本文从开发和探索的角度出发，揭示出UNIX系统中PASCAL与C语言间参数传递的内在规律以及解决参数(尤其串参数)不相容的办法，并尽可能的提供了调试用例．

---

* 收稿日期: 1990.01.15

1 关于C函数和PASCAL的外部过程

C语言只有函数，当调用具有形参的函数时，函数的参数替换是传值的，即被调用函数的形参所接受的是实参的一个临时副本，而不是实参的地址．因此尽管形参变量的值在不断发生变化，但它却丝毫不能影响对应实参的值．这与PASCAL、FORTRAN、PL/I等语言是不同的．掌握了这个特点，就明白了，为什么通过调用下面C函数不能完成X、Y两变量内容互换的原因．

```
main ()
 {int  x, y;
swap(x, y);
printf( "x = %d * * * y = %d\n" , x, y);
} ;
swap(a, b)
int  a, b;
 {int  temp;
temp = 1;
a = b;
b = temp;
printf( "a = %d####b = %d\n" , a, b);
}
```

若令主程序中$x=1$，$y=2$，则程序的执行结果为:

```
x = 1 * * * y = 2
a = 2####b = 1
```

可见在调用swap后，达到互换的是变量a和b，而不是x和y．该动作仅仅等价于执行下述的复合语句:

```
 {a = x;
  b = y;
}变量参数传送
temp = a;
a = b;
b = temp;
 }
```

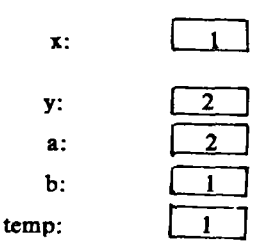


图1

各单元内容如图1所示．

PASCAL语言调用C函数的外部过程形式为:

```
  :
function  examp(var ab: longint; Var name: string[20]): Longint;
external;
begin
```

```
    :
fd:  = examp(x, ne);
    :
end.
```

其中，由于 PASCAL 与 C 语言对函数值的空间存贮方案处理不同，还必需设计一个汇编接口 wrapper（参看[3]）以实现二者之间信息的正确传输.

2　参数为整型变量时的传递

由于 C 语言采用的是传值方式，而 PASCAL 语言一般采用的是传地址方式，(采用变量参数)，因此，若想在 C 函数中重现 PASCAL 语言（调用程序）中整型实参的真实值，就应在 C 函数中把对应形参说明为整型指针形式，(当然，若 PASCAL 语言采用的是值参，那么对应 C 函数参数就应是整型形式)，否则 C 函数中得到的只是一个地址（八进制数字串）. 如:

```
调用程序: program exp1;
var i, j: integer;
fuction fd(var x: longint): longint;
external;
begin
i:  = 6;
j:  = fd(i);
    :
end;
```

C函数: fd(a)

```
int * a; (* 说明为指针形式 *)
 {printf( "a = %d\n" , * a);
}
```

程序的执行结果为:

```
a = 6
```

这是正确的. 若C函数改为如下形式:

```
fd(a)
int a;
 {printf( "a = %d\n" , a);
}
```

则程序执行结果为:

```
a = xxxxxxx
```

其中*xxxxxxx*为七位八进制串（如$a = 2061642$），这显然是错误的.

因此，若要通过 swap 调换前面例中 x，y 两变量的内容，必需正确的设计出下面的 C 函数:

```
swap(a, b)
int *a, *b;
 {int temp,
temp = (*a);
*a = (*b);
*b = temp;
}
```

注意，这里函数体中出现在赋值号"="右边的 *a、*b 之所以要加上括弧，是为了防止两运算符相连成为"=*"时产生的二义 (ambiguous) 性，因为 C 语言的表达式属于算符优先文法.

3 串变量作为参数时的传递

实现 PASCAL 与 C 语言之间串传递的相容性，情况较为复杂.

3.1 为什么要传送串参数?

我们知道，计算机系统中的各种资源都是由操作系统统一管理的. 因此，在操作系统的外层软件 (语言，应用程序等) 或一般的用户程序中，凡是涉及到与系统资源有关的操作，都必需通过某种方式向操作系统提出请求，并由操作系统中的某几个程序代为完成这个请求. 此外，操作系统还常常为用户提供一些方便有效的手段，如查询时间、日期等信息. 为此操作系统必需提供某种接口，以便让外层软件和用户程序通过这种接口，方便地使用操作系统提供的各种功能，这种接口称之为系统调用. 当外层需要时，只要在程序中安排类似于高级语言中语句形式的系统调用，就能完成外层软件或用户程序提出的各种请求.

系统调用是 unix 操作系统面向用户状态下运行程序的界面. C 语言提供有多条系统调用，如:

```
fd = creat(filename, mode);
fd = open(filename, mode);
n = read(fd, buf, count);
n = write(fd, buf, count)
```

等. 另外C语言还提供有系统调用函数，如:

```
system(string)
```

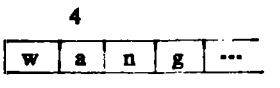
等. 所有这些系统调用语句及系统调用函数对 PASCAL 语言来说是缺省的. 因此当在很多情况下需用 PASCAL 语言调用 C 语言以实现与系统传递信息时，如要执行某条系统命令或需要在外部存贮器上进行扫描以决定某一文件是否存在时，都需要在 PASCAL 语言中能把上述命令或文件名作为串参数准确无误地传递到 C 函数中去以完成其使命.

3.2 PASCAL 的串变量与 C 语言字符数组的不一致性

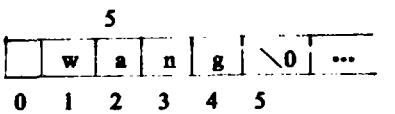
PASCAL 语言中的串（string）变量与 C 语言中的字符数组（char A[20]）基本可划为同一数据类型．但应注意到前者对串进行扫描时，下标是从 1 到 N（假定串长为 N），而后者的下标却是从 0 到 N−1，且由于 C 语言对处理字符数组的需要，编译器会在串尾自动追加一个标志符"＼0"（即 ascII 码转意的 NULL）．于是 PASCAL 中长度为 N 的串传递到 C 函数中成了 N+1，且其第一个下标 0 中的内容处于无定义的状态．如：

欲传递的串为"wang"，那么

在 PASCAL 中：



传到 C 语言中：

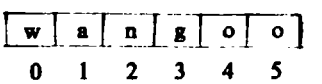


由此可以看出，被传到 C 函数中的串"走样"了，它已不是原来的串"wang"，而是成了"?wang"，其中，"?"表示无定义状态．若该串原为一系统命令或一个文件名，那么通过 C 函数中的系统调用语句或是系统调用函数执行时，就不能得到期望的结果．

对于这种情况，我们可以在 C 函数中对被传递的字符数组进行适当处理，以使"走样了"的串恢复其本来面目．方法如下：

```
fd(name)
char name〔 〕;
 {int i, j;
for(i=0; name〔i〕 != '\0'; i++)
name〔i〕 =name〔i+1〕;
j=open(name, mode); /*系统调用*/
}
```

经过这样的左移处理，C函数中的字符数组就变成了：



虽然在串尾多出一个符号 '\0'，便由于 C 编译器是以扫描到的第一个 '\0' 为作串尾识别标记的，故第二个 '\0' 已无实质意义．这样，左移后的系统调用句

```
j=open(name, mode)
```

可以正确的执行．

3．3　从 PASCAL 向 C 函数传递串变量时，C 函数对该串的复盖特性

C 函数在接受由 C 语言调用段传递的串变量和接受由 PASCAL 调用段传递的串变量时具有不同的特性．

3．3．1　C 语言程序调用 C 函数时字符数组的赋值特性．如下程序：

```
main ()
 {char name〔20〕;
int i;
yy: scanf("%s", name);
printf("***name=%s***\n", name);
i=fd(name);
```

```
goto yy;
} ;
fd(ne)
char ne〔 〕
 {int i;
printf( "\n####ne = %s####\n" , ne);
i = open(ne, mode);
close(i);
returen(i);
}
```

在对该程序的执行过程中；反复对同一字符数组名name进行键盘输入，如第一次输入：　"wnag"

则输出为：* * * name = wang * * *

ne = wang

第二次输入：　"ZZZ"

则输出为：* * * name = ZZZ * * *

ne = ZZZ

……。

由输出结果知，当前输入的字符数组 name 内容传到函数 fd 中是，是先对上次 ne 内容清除之后再赋给 ne 的．因为 ne 内容与 name 相同，故系统调用句的执行结果是正确的．

3．3．2　PASCAL 程序调用外部过程 C 函数时字符数组的"复盖"特性（C 语言编译的不足）

若调用 C 函数的不是 C 语言程序而是 PASCAL 程序，就会产生根本不同的结果，如

```
program exps;
label 100;
type str = string〔20〕 ;
var i: integer;
name: str;
function fd(var name: str): longint;
external;
begin
10:readln(name);
i: = fd(name);
goto 10;
end.
```

C函数：fd(ne)

```
char ne[ ]
 {int i, j;
for(j=0; ne[j] ! = '\0' ; j++)/ *左移* /
ne[j] =ne[j+1] ;
printf( "\n####ne=%s####\n" , ne);
i=open(ne, mode);
close(i);
return(i);
}
```

本程序执行结果：

PASCAL 输入 I: "wang"

C 函数输出: #### ne = wang ####

PASCALl 输入 II: "ZZZ"

C 函数输出: "#### ne = ZZZg ####

PASCAL 输入 III: "ABCDEF"

C 函数输出: #### ne = ABCDEF ####

PASCALl 输入 IV: "$ $ $"

C 函数输出: #### ne = $ $ $ DEF ####

由上述结果，我们发现，当字符串作为参数由 PASCAL 向 C 语言传输时的特点为

①当前输入串长度小于前次输入串长度时，C 编译器不再向当前输入串尾自动追加尾识别标志'\0'，但把前次串尾标志符作为当前串的结束标志．如上例

上次 ne:

w	a	n	g	\0	…

当前 ne:

z	z	z	g	\0	…

而不是 ne:

z	z	z	\0	…

②当前输入串长度大于前次输入串长度时，则 C 函数中的 ne 将"复盖"前次输入内容，而以本次输入内容为 ne 的内容，且 C 编译器在当前串尾加上结束标志'\0'．如上例

前次 ne:

z	z	z	g	\0	…

当前 ne:

A	B	C	D	E	F	\0	…

③为防止① 中的情况而在 C 函数中对 ne 采取任何形式的置空措施，如在

i=open(ne, mode)

后加上 ne='␣␣␣…\0'（20 个空格）都无法消除前次长串对当前短串尾部的影响．

正是由于上述① 中的"复盖"特点，造成了在 C 函数中 ne 的第二种"走样"，故不能使系统调用句

i=open(ne, mode)

执行正确的信息传递．

3．3．3 纠正措施

为了纠正因 ne 新的"走样"而产生的失真，有两种解决办法．其一，在 PASCAL 调

用程序中完成检测当前输入串长度 q 的任务，然后把 q 连同串本身作为两个不同的参数同时传递给 C 函数。在 C 函数中由程序员在字符数组 ne 下标为 * q+1 处人工（而不是由 C 编译器）加上串尾标志'＼0'，即加上：ne〔* q+1〕='＼0'。这就弥补了因 C 编译器有时功能不全导致的缺陷。其二，在 PASCAL 中给当前输入串追加一尾部符'¦'，而在 C 函数中不以'＼0'作为串尾的判断标志而改以'¦'为判断标志。该方法与前一种方法等效，但可读性不如前者好。上述两种方法都可以保证系统调用句的正确执行。仅提供第一种方法的调试程序如下：

```
program exps;
label 10;
type str = string[20]
var i: integer;
name, name1: str;
len: longint;
function fd(var name: str; var int: longint): longint;
external;
begin
10: readln(name):
name1: = name;
len: = length(name1);
i: = fd(name1, len);
goto 10;
end.
C函数: fd(q, ne)
int * q;
charne〔 〕;
 {int i, j;
ne〔* q + 1〕 = '/0';
for(j = 0; ne〔j〕 ! = '/0'; j + +)
ne〔j〕 = ne〔j + 1〕;
j = open(ne, mode);
close(i);
return(i);
}
```

4 结 束 语

不同数据类型的参数由一种语言传递给另一种语言，必需具备有一定的软件环境。上述研究是在 unix 系统下实现的。尽管在 PASCAL、C 之间传递串参数的情况比较复杂，

但一旦找出了它们的内在规律，设计出了相应的接口，使得参数能在空间存贮分配上及其它一些方面取得相容，那么对利用这两种语言的特点设计系统应用程序（如语言处理、编辑等）将具有十分积极的意义。因为它使完成的应用软件既有高度的灵活性及能对硬件寄存器进行存取的优点，又具有安全、可靠性系数高的长处，这正是计算机用户所期望的。

参　考　文　献

〔1〕 尤晋元. UNIX操作系统教程. 西北电讯工程学院出版社. 1985.

〔2〕 孙玉方. 孟庆昌. C语言及其程序设计. 计算机研究与发展编辑部. 1985.

〔3〕 王文义. 共享程序包在unix系统中的实现方法. 郑州工学院学报, 1988年第4期

〔4〕 ALFRED V.AHO, JEFFREY D.ULMAN. Principle of Compiler Design. 1977

〔5〕 Silicon Velley Software Inc. SVS PASCAL / 300. C Reference Manual. 1982

Research into Incompatibility of the Parameter Transmission

Wang Wen Yi

(The department of computer and automation)

Abstract: Taking into account the fact that the C language and PASCAL language have different data type definition and storage administration scheme respectively, the auther of this article in detail explores the incompatibility of the parameter transmission between the two languages, which is caused by the limitation of the limitation of the C compiler's function. The author puts forward the solution to this pfoblem and lays the foundation for the realization of the high efficiency of the program in the sysytem application.

Keywords: Parameter transmission, Incompatibility, Unix system.