C 语言的调用惯例与变参函数设计

范向兵 李莹 王爱学 （大庆石油学院计算机科学系 151400）

摘要：本文深入地讨论了 C 语言的两种不同的函数调用惯例：C 调用惯例和 PASCAL 调用惯例，阐述了 C 调用惯例与变参函数实现的关系，并详细地介绍了 C 语言的可变参数函数设计方法。

关键词：调用惯例 主调函数 被调函数 变参函数 指针 宏

1. 引言

到目前为止，C 语言尽管很流行，其自身的一些特点仍然没有完全地被广大应用开发人员所充分地认识，进而加以灵活地应用于科学实践中。这里我们将要介绍的就是其中的一个方面——C 语言的调用惯例与变参函数设计。

2. C 语言中的调用惯例

对多种调用惯例的支持是 C 语言的一个重要特色，调用惯例上的不同决定着 C 编译系统生成的函数在调用处理方式上的差异。C 语言中支持 C、PASCAL、REGISTER 三种不同的调用惯例。这些调用惯例各自在堆栈的清理、参数的传递顺序、大小写字符和全局标志符前缀（下划线）的处理等方面都是有所不同的。在 C 语言中，函数使用哪种调用惯例可以通过编译命令参数确定，也可以对应地使用关键字--cdecl、--pascal 或--fastcall 来直接为函数指定所采用的调用惯例。

REGISTER 调用惯例可以使函数在调用过程中通过寄存器来实现参数的传递，从而提高效率。但这种调用惯例是否奏效，一般与系统中寄存器的使用情况相关，在一般场合，特别是在微机上使用不多。

现在我们着重来看 C 调用惯例和 PASCAL 调用惯例。PASCAL 调用惯例在传递参数时是将参数列表中的各个参数由左向右置入堆栈，也就是说各参数是从第一个参数开始一个一个按顺序压入堆栈中。C 调用惯例则刚好相反，单看这一点在效率上倒没什么特别的关系。但是堆栈指针的还原，在 C 调用惯例中必须由主调函数来完成，而在 PASCAL 调用惯例中，则由被调函数自己来负责。因此若使用 C 调用惯例，那么对同一个函数的多次调用中，每次调用之后都要由主调函数进行堆栈的清理，于是调用越多，程序代码就越长，执行效率也就越差。

Microsoft 在将 Windows 由 C 调用惯例改为 PASCAL 调用惯例之后，使整个程序规模小了百分之九之多，调用惯例的影响可见之大。

也许你会疑问：既然 PASCAL 调用惯例有那么多的优点，为什么还用 C 调用惯例呢？其实 C 调用惯例也有其特有的长处，它的最大优点就是同一个函数可以接收不同个数和类型的参数，比如大家都很熟悉的标准库函数 printf()和 scanf()就是两个很典型的例子。

为什么 C 调用惯例能够支持可变参数函数的设计呢？其关键就在于 C 调用惯例对所传参数的处理上：由主调函数负责恢复调用前的堆栈现场。因为如果参数不确定，若按 PASCAL 调用惯例在被调函数中恢复堆栈现场，则被调函数根本无法用相同的代码实现不同参数个数和类型的多次调用的现场清理。而这个工作如果由主调函数在每次调用之后来做问题就简单得多了。由于 Windows Function 只接收固定个数和类型的参数列表，因此在声明任何 Windows Function 时都采用 PASCAL 调用惯例，这样既不影响功能的完成，又减小了系统规模，提高了程序的效率。但是毫无疑问，可变参数函数是软件开发中一种非常有用的技术手段，在许多实际开发中都存在着这方面的客观需求。

然而，C 程序开发资料中介绍的多是固定参数函数的设计方法，对于可变参数函数的设计方法一般都没有涉及。那么程序员能否开发自定义的参数可变的函数呢？答案是肯定的。

3. 可变参数函数的开发方法

在 C 系统的标准函数库头文件集合中，有 stdarg.h 这样一个头文件，包含了一些进行可变参数函数设计所必需的定义。其中最主要包括了一个类型和三个宏的定义，它们为在可变参数函数中访问参数列表中的那些可变参数提供了一种便捷的手段。当被调函数不知道所传

来的参数的个数及类型时，可以利用它们来一步步地访问参数列表中的每一个可变参数。下面我们先看一个能够接受可变参数的函数定义的例子：

```
#include <stdio.h>
#include <stdarg.h>
/* sum:计算若干整数之和 */
int sum(int num, int data1, ...)
{
  int total=data1;
  va-list ap; /* 说明变量 ap */
  int arg,i;
  va-start(ap,data1); /* ap 被初始化为指向 data1 后面的参数 */
  for(i=1;i<num;i++)
  {
    arg=va-arg(ap,int); /*把 ap 所指向的 int 类型的参数的值赋给 arg,同时使 ap 指向下一个参数 */
    total+=arg;
  }
  va-end(ap); /* 清除 ap */
  return (total);
}
int main(void)
{
  int value;
  value=sum(5,11,12,13,14,15); /* 第一次 sum 函数调用求五个整数之和 */
  printf("The result is %d\n",value);
  value=sum(7,10,20,30,40,50,60,70); /* 第二次 sum 函数调用求七个整数之和 */
  printf("The result is %d\n",value);
  return (0);
}
```

说明：

(1) C 语言中省略号…用于说明参数可变的函数，所以上例中

```
int sum( int num , int data1 , ... )
```

表明函数 sum 带有两个 int 类型的固定参数，而在固定参数 data1 后面可以带有更多的可变参数。在这里所有的可变参数的类型约定都是 int 类型。

(2) va-list、va-start、va-arg 和 va-end 都是在头文件 stdarg.h 中定义的，va-list 是一种指针类型，后三者都是带参数的宏。

在上例中：

·语句

```
va-list ap;
```

在函数 sum 中定义了一个 va-list 类型的变量 ap。当某个被调函数接收一个可变的参数列表时，这个函数就需要定义一个 va-list 类型的变量用于引用省略号…所代表的可变参数序列。

·语句

```
va-start(ap , data1);
```

使用宏 va-start 初始化 ap，使 ap 指向 data1 后面的第一个参数，也就是省略号所代表的可变参数序列中的第一个参数。宏 va-start 的语法形式如下：

```
void va-start(va-list ap , lastfix);
```

其中 lastfix 代表函数的最后一个固定参数。va-start 必须在 va-arg 和 va-end 之前使用。

·语句

```
arg = va-arg(ap , int);
```

通过宏 va-arg 将 ap 所指向的参数值(宏调用中的第二个参数 int 表明这个参数被解释成 int 类型)置给 arg，同时使 ap 又指向下一个参数。宏 va-arg 的语法形式如下：

```
void va-arg(va-list ap , type);
```

va-arg 中用的变量 ap 必须是用 va-start 初始化的 ap，type 指的是数据类型。要注意：type 不能使用 char 或 float 类型，因为如果在函数调用时可变参数部分使用了这几种类型，对应地系统会自动将之向上转换成 int 和 double 类型。对于固定参数而言，这种转换是没有的，实际定义成什么类型就传递什么类型。

·语句

```
va-end(ap);
```

消除 ap，这表明 ap 不再用于程序的以后部分。此后如需使用 ap，则必须重新调用宏 va-start 进行初始化，否则可能会引起程序奇怪的、不确定的行为。宏 va-end 的语法形式如下：

```
void va-end(va-list ap);
```

它的调用在变参函数设计中也是不可或缺的。

(3) 在前例中，sum 函数有两个固定参数，在调用时编译器会对它们进行相应的参数类型检查与转换，但这些工作对其后的可变参数就不起作用了。省略号…抑制了对可变参数部分的类型检查。这使得变参函数调用中

隐藏的错误不能在编译阶段发现，因此在使用变参函数时一定要小心仔细。

(4) 在进行变参函数设计时，必须有办法告诉变参函数没有指定的参数的个数和类型。前例中 sum 函数由第一参数指出其后参数的个数，而由使用约定规定可变参数的类型也均为 int 类型；我们熟悉的 scanf、printf 函数则是通过第一个参数——格式控制串来传递可变参数的个数和类型信息的。

必须特别强调，进行可变参数函数的定义时，其参数列表中至少要有一个固定参数，同时固定参数必须在前，之后才能是可变参数部分。

4. 结束语

以上我们讨论了 C 语言中的调用惯例问题，并详细介绍了可变参数函数的开发方法。了解与充分认识 C 语言中的调用惯例，无疑对指导我们灵活地选择运用此类特性，以实现更加精练高效的实用软件具有重要的意义。可变参数函数的开发方法，是 C 一个极具特色的语言特点，它为我们开发实现功能灵活、使用方便、接口简洁的程序模块提供了有力的支持，提高了程序中功能模块的通用性，有效地减少了软件开发中不必要的重复，也简化了程序中各模块之间的纷繁的调用关系，从而为增强程序可读性提供了一种有效的保障方法。

参考文献

[1] 谭浩强，C 程序设计，清华大学出版社，1993.7

[2] 张国峰，C++ 语言及其程序设计教程，电子工业出版社，1993.4

[3] 夏洪山、林志坚，Microsoft Windows 软件开发技术基础，海洋出版社，1992.3

[4] 何立起，Borland C++ Windows 程序设计，人民邮电出版社，1994.6

(来稿时间：1997 年 8 月)