XENIX 库函数程序员手册

郑　蕾　译

孙玉方　校

本 书 概 述

本手册说明了如何使用XENIX系统的C语言库所提供的函数。它特别描述了C语言的两个库函数：

- 标准C语言库；
- 称做curses的屏幕更新和光标移动库。

本手册假定你已懂得C程序设计语言而且熟悉XENIX shell—sh。本书中几乎所有的程序设计实例都是用C写成的，而且所有说明shell的例子都是用sh (Bourne shell)。

如果你以前从未用过C的库函数，那么请首先阅读本手册，然后，如果你需要了解更详细的内容，可参阅《IBM PC XENIX软件命令参考手册》。

如果你对库函数很熟悉，则可参阅《IBM PC XENIX软件命令参考手册》，以了解这些函数和你已经知道的那些有什么不同，而在本手册中，你能查到有关函数的例子。

本书的结构如下：

第一章 引言

给出C语言库的概况并介绍了在本手册中使用的约定。

第二章 使用标准I/O函数

描述了标准输入输出函数，这些函数使得程序能够读、写XENIX文件系统中的文件。

第三章 屏幕处理

描述了屏幕处理函数，这些函数使得程序能够使用用户终端的屏幕处理设施。

第四章 字符和字符串处理

描述了字符和字符串处理函数，这些函数使得程序能够赋值、处理和比较字符及字符串。

第五章 使用进程控制

描述了进程控制函数，这些函数使得一个程序能够执行其它程序并可创建其自身的多个副本。

第六章 创建并使用管道

描述了管道函数，这些函数使得程序能够与另一程序通信而无需建立一个临时文件。

第七章 使用信号

描述了信号函数，这些函数使程序能够处理那些通常是由系统处理的信号。

第八章 使用系统资源

描述了系统资源函数，这些函数使程序能够动态地分配存储、与其它程序共享内存、封锁文件防止其它程序存取以及使用信号装置。

第九章 错误处理

描述了错误处理函数，这些函数使得程序能够在存取文件系统或分配存储空间时进行

出错处理。

附录A　汇编语言接口

描述了汇编语言和C语言之间的接口，并且说明了C函数的调用以及返回值的约定。

附录B　XENIX系统调用

说明如何创建和使用新的XENIX系统调用。

附录C　XENIX和DOS的公用库

列出了构成XENIX和Mircosoft C编译程序的DOS版本的公用C库的XENIX库程序。

与本手册有关的XENIX资料有：

- IBM PC XENIX安装指南
- IBM PC XENIX系统管理手册
- IBM PC XENIX命令参考手册
- IBM PC XENIX软件开发指南
- IBM PC XENIX汇编程序参考手册
- IBM PC XENIX直观shell
- IBM PC XENIX基本操作指南
- IBM PC XENIX C编译程序参考手册
- IBM PC XENIX软件命令参考手册

目　　录

第一章 引　　言

1.1 使用C语言库函数

C的库函数可由任何一个需要使用XENIX系统资源来执行任务的程序调用。这些函数允许程序读、写XENIX文件系统中的文件，读、写诸如终端和行式打印机之类的设备，装配并执行其它程序，接收并处理信号，通过管道与其它程序通信，共享系统资源以及进行错误处理。

为使用C的库函数，你必须在程序中包括适当的程序调用及定义，并且在编译程序时说明相应的库。当你编译一个C语言程序时，包含在文件libc.a中的标准C库被自动说明了，而其它的库，包括存于文件libcurses.a中的屏幕更新和光标移动库，都必须在编译程序时用cc命令的-l任选显式说明（参见《IBM PC XENIX软件开发指南》中“cc：一个C编译程序”）。

1.2 符号约定

本手册使用几个特殊符号来描述库函数调用的格式，下面是这些符号及其意义的说明：

[ ]	方括号指示一个任选的函数参数。
…	省略号表示它前面的参数可以重复一次或多次。
SMALL	表示显示常量，这些是依赖于系统的常量，它们在各种include文件中定义。
斜体字	斜体字表示函数的位置，它们必须由合适的值或变量名代替（译文中用正体英文表示——译注）。

第二章　使用标准I/O函数

2.1　引言

几乎所有的程序都要用到某种形式的输入和输出。有些程序读、写存在盘上的文件，还有些则向行式打印机一类的设备写，许多程序读写用户终端。标准C库提供了几个预定义的输入输出程序设计函数。

本章说明如何使用标准C库中的I/O函数，特别描述了：

- 命令行参数
- 标准输入和输出函数
- 用于普通文件的字符流函数
- 用于普通文件的低级函数
- 随机存取函数

2.1.1　为I/O函数做准备

为了使用标准I/O函数，程序必须包含文件stdio.h，该文件定义了所需要的宏及变量。要包含此文件，只要把下行放到程序的开头即可：

```
#include <stdio.h>
```

实际的函数在库文件libc.c中，在你编译一个程序时，该文件被自动读入，所以，在调用编译程序时，就不再需要特别参数了。

2.1.2　专用名

标准I/O库使用许多有专门用途的名字，一般说来，这些名字可用在任何包含stdio.h文件的程序中。专用名是：

- stdin　标准输入文件的名字。
- stdout　标准输出文件的名字。
- stderr　标准出错文件的名字。
- EOF　读子程序在文件尾或是出错时的返回值。
- NULL　指示一个错误的空指针，它由带有指针值的函数返回。
- FILE　文件类型的名字，被用于说明指向字符流的指针。
- BSIZE　用户提供的适当的I/O缓冲区大小（字节数，通常是512）。

2.1.3　专用宏

函数getc, getchar, putc, putchar, feof, ferror和fileno实际上都是宏，而不是函数，这意味着你不能重新说明它们或者在调试时把它们做为断点。

2.2　使用命令行参数

XENIX系统允许你在执行一个程序的同时把信息传递给该程序，这只要使用命令行参数就行了。

XENIX命令行就是你敲入以引用程序的行，命令行参数则是你在一个XENIX命令行中敲入的任何东西。命令行参数可以是一个文件名、一个任选项或是一个数。命令行中的第一个参数必须是你想要执行的程序的文件名。

在你敲入一个命令行后，系统读入第一个参数并装配相应的程序，它还对其它参数计数，按照它们在行中出现的次序把它们存到内存中，并且把计数和存放位置传送给程序的主函数，于是，该函数就能够通过存取参数所在的内存来存取参数了。

为存取这些命令行参数，主函数须有两个参量：argc，存放参数计数的整型变量；argv，指向参数值的指针的数组。你可以在主程序函数的开头用下列行来定义这两个参数：

```
main (argc, argv)
int argc;
char *argv[ ];
```

当程序开始执行时，argc中含有计数，argv的每个元素则含有指向一个参数的指针。

参数是作为一个以空结束的串（亦即，一个以空字符\0结束的串）存放的。第一个串（在argv[0]中）是程序名字，参数计数永远不会小于1，因为程序名总是被当作第一个参数。

在下例中，先读入命令行参数，然后把它们回应到终端屏幕上。该程序类似于XENIX I的echo命令：

```
main (argc, argv)          /* echo arguments */
int argc;
char *argv[ ];
    int i;
    for (i=1; i<argc; i++)
        printf ("%s%c", argv[i], (i<argc-1) ? ' ' :'\n') ;
```

在上例中，每个参数后面都附加了一个空格以便把它和下一个参数分隔开。需要这样作是因为系统在命令行中读入参数时，自动把参数前后的空白格字符（即空格和制表符）删去了。在最后一个参数后面加一个换行符，只是为了方便，因为这样就可以在程序结束时使得shell提示符出现在下一行上。

当你在命令行上敲入参数时，要确保各个参数间用空白格符分开。如果某一个参数中必须含有空白格符的话，可用双引号将该参数括住。例如，在命令行：

```
display 3 4"echo hello"
```

中，串“echo hello”就被当做一个参数。而且，把参数括在双引号中还可标识出任何含有可由shell认识的字符的参数（例如，<，>，|和^）。

你不能改变argc和argv的值，如果有此必要，可以把参数值赋给另一变量，然后修改那个变量；把这些参数的值赋给外部变量，你就可以把对这些参数的存取权交给程序中的其它函数了。

2.3 使用标准文件

在你执行一个程序时，XENIX系统就自动地创建一个标准输入文件，一个标准输出

文件和一个标准出错文件来处理程序的输入输出需要。因为多数程序的大部分输入输出都是通过自己的终端进行的，所以系统总是把用户的终端键盘和屏幕分别指定为标准输入和输出，而用于接收由程序产生的错误信息的标准出错文件也被指定为终端屏幕。

程序可用函数getchar，gets，scanf，putchar，puts和printf来读、写标准输入和输出文件；标准出错文件则可用字符流函数来存取。关于字符流函数，将在本章后面的“使用字符流函数”一节中描述。

XENIX系统允许你用shell的改向符号（<和>）来改向标准输入和输出，这就允许程序使用其它设备和文件代替终端键盘和屏幕来做为它的主要输入源和输出目的地。

下面各节解释了如何读、写标准输入和输出以及如何改向标准输入和输出。

2.3.1 从标准输入读

你可用函数getchar，gets和scanf从标准输入读。

函数getchar每次从标准输入读一个字符，其调用形式是：

```
c=getchar( )
```

其中，c是用以接收字符的变量，它必须为整型，整型意味着列出的变量是一个整数。正常情况下，该函数返回它读入的字符，而在遇到文件尾或遇到错误时，则返回值EOF。

函数getchar可用在条件循环中从标准输入读入一个字符串。例如，下面的函数从键盘上读入cnt个字符：

```
readn (p, cnt)
char p[ ];
int cnt;
    int i, c;
    i=0;
    while (i<cnt)
            if ( (p[i++]=getchar ( ) ) ==EOF)
                    p[i]=0;
                    return (EOF) ;
            }
    return (0) ;
```

如果getchar正在读键盘，它就在返回之前等待字符的键入。

函数getchar正在读键盘，它就在返回之前等待字符的键入。

函数gets从标准输入读入一个字符串并把它拷贝到给定的内存位置上，其调用形式是：

```
gets (s)
```

其中，s是指向接收串的位置的指针。该函数读入字符直到发现一个换行符（\n）时为止，然后，它用一个空字符(\0)来代替该换行符，并把结果拷贝到内存中。如果遇到文件尾或是错误，该函数就返回空指针值NULL，否则，返回s的值。

该函数可用来从标准输入读入一整行，例如，下程序段从标准输入读入一行，把它存在字符数组cmdln中，而且，如果没有出错的话，调用一个函数（称为parse）：

```
char cmdln[SIZE];
if (gets (cmdln) != NULL)
```

```
parse ( ) ;
```

在这种情况下，假设字符串的长度小于“SIZE”。

函数gets不能检查字符串的长度，因此，可能会发生溢出。

函数scanf从标准输入读入一个或多个值，这里，值既可是一个字符串，也可是一个十进制或八进制或十六进制数，其调用形式是：

```
scanf (format, argptr, . . )
```

其中，format是一个指针，指向定义待读入值的格式的字符串；argptr是一个或多个指针，指向接收值的变量。在format串中给出的每一个格式，都须有一个argptr与之对应。格式可以是：%s，表示一个字符串；%c，表示一个字符；%d，%o，%d，分别表示一个十进制数、一个八进制数和一个十六进制数（其它格式在《IBM PC XENIX软件命令参考手册》中的scanf（s）一节中描述）。正常情况下，该函数返回它由标准输入读入的值的个数，但在文件尾或是在遇到错误时，返回值EOF。

与getchar和gets不同，scanf跳过所有的空白符，只读入那些构成值的字符，然后，如果有必要的话，它把这些字符转换成适当的字符串或数。

在需要做格式输入时，可用函数scanf。所谓格式化输入是指那些必须以特殊方式键入或是有特殊意义的输入。例如，在下面的程序段中，scanf从同一行读入一个名字和一个数：

```
char name[20];
int number;
scanf ("%s %d", name, &number) ;
```

在本例中，字符串“%s%d”定义了要读入的是什么值（一个字符串和一个十进制数），字符串被拷贝到字符数组name中，而数则在整型变量number中。注意，在调用中用的是指向这些变量的指针，而不是实际的变量自己。

在读键盘时，scanf在返回前等待把值键入。各个值之间应用一个或多个空白格符分开，如空格，制表符或是换行符。例如，对函数：

```
scanf ("%s %d %c", name, &age, &sex) ;
```

一个可以接收的输入是：

```
Jeni 22
F
```

如果值是一个数，那么该数应有合适的数字。亦即，一个十进制数必须由十进制数字组成，八进制数由八进制数字组成，十六进制数则由十六进制数字组成。

如果scanf发现了一个错误，它就立即停止读标准输入，在再次使用scanf之前，必须用getchar把引起错误的非法字符从输入中删去。

你可以在同一个程序中使用getchar，gets和scanf，只是要记住，每个函数都是读取下一个可用的字符，从而使该字符对其它函数而言成为不可用的。

当标准输入是终端键盘时，getchar，gets和scanf总是要等到至少键入一个换行符之后，才返回一个值，即使只要读取一个字符也是这样。如果你希望在一次击键之后立即输入，可参阅第三章中“使用标准屏幕”一节。

2.3.2 向标准输出写

你可用putchar，puts和printf向标准输出写。

函数putchar把单个字符写到输出缓冲中，其调用形式是：

```
putchar (c)
```

其中，c是要写的字符。正常情况下，该函数返回它写出的字符，但是在遇到错误时，则返回值EOF。

把该函数用在条件循环中可以写一个字符串到标准输出上。例如，函数：

```
write (p, cnt)
char p[ ];
int cnt;
    int i;
    for (i=0; i<=cnt; i++)
            putchar ( (i! =cnt) ? p[i]:'\n') ;
```

把cnt个字符和一个换行符写到标准输出上。

函数puts把在给定内存位置上找到的串拷贝到标准输出上，其调用形式是：

```
puts (s)
```

其中，s是一个指针，指向含有串的位置。该串可以有任意个数的字符，但必须用一个空字符（\0）结束，该函数把串中的每个字符写到标准输出上，并把串尾的空字符替换成一个换行符。

由于该函数自动地附加一个换行符，所以可在向一标准输出写一个整行时使用它。例如，下面程序段把三个字符串中的一个写到标准输出上：

```
int c;
switch (c) {
        case ('1') :
            puts ("Countinuing. . . ") ;
            break;
        case ('2') :
            puts ("All done. ") ;
            break;
        default:
            puts ("Sorry, there was an error. ") ;
```

写出的串是哪一个取决于c的值。

函数printf把一个或多个值写到标准输出上，这里，值可以是一个字符串，也可以是十进制或八进制或十六进制数。该函数自动把数转换成适当的显示格式，其调用形式是：

```
printf (format[, arg]. . . )
```

其中，format是一个指针，指向描述要写的各个值的格式的串；arg是一个或多个含有待写值的变量。对应于format串中的每个格式，都须有一个arg。格式可以是：%s，表示一个字符串；%c，表示一个字符；%d，%o和%x分别表示一个十进制，一个八进制和一个十六进制数（其它格式在《IBM PC软件命令参考手册》一书的printf（s）中描述）。如果要求的是一个字符串，那么，对应的arg必须是一个指针。正常情况下，该函数返回零，但在遇到错误时，返回一个非零值。

函数printf主要用在要进行格式化输出的情形下，亦即，当输出必须以某种方式显示