《现代电子技术》增刊

XENIX系统V安装操作使用大全

(下)

王中平　王　毅　编写

陕西电子杂志社

XENIX SYSTEM V 安装·操作·使用大全

下 册

XENIX 系统程序员指南

王中平　　王　毅　编译

陕西电子杂志社

前　言

UNIX 系统 V 是 AT&T 较成熟的商业版本，XENIX 系统是 UNIX 系统在以 Intel 芯片为主 CPU 的微机上的实现，SCO XENIX 系统 V 是目前在我国的高档微机上最为流行的多用户 XENIX 系统、主要运行在 286、386、PS/2 及其兼容机上，较为普遍的有 2.2.3、2.3.1、2.3.2 及 2.3.4 版。一些研究所和软件公司亦对 SCO XENIX 系统 V 内核进行了汉化，但都不很彻底、没有赶上在 DOS 上汉化的水平。

随着我国目前微机应用水平和微机配置的提高，微机网络和多用户系统逐渐成为用户升级换代的选择，XENIX 系统可以是最好的选择之一，因此，SCO XENIX 日趋流行，目前，虽然 SCO UNIX 经 UNIX SVR4.2 等已在中国推出，但由于其硬件配置要求较高，不适合一般配置的用户，因此，大部分用户还是首选 SCO XENIXsystem v' 基于这种考虑，我们编译了此大全，从系统的安装使用、管理、维护、开发等方面详尽地进行了描述，分用户手册、系统管理员手册和程序员手册三分册编排。

本书由王中平、王毅编译，在全书形成过程中，承张忠智先生、孙彩贤女士大力协助，谨致谢意。

编译者

1993.3.15

内容提要

XENIX SYSTEM V 安装·操作·使用大全，包括上、中、下三册，上册为 XENIX 系统用户指南。中册为 XENIX 系统管理员指南。下册为程序员指南。

下册共包括三个部分，分别为：第一章 C 语言概述；第二章 XENIX 系统调用和 C 语言库函数；第三章程序开发工具。

本书是使用计算机 XENIX 系统必不可少的参考书，而且也是一本必不可少的工具书。

目　录

第一章　C语言概述

C语言是一种通用的程序设计语言，它的效率、经济性以及可移值性是众所周知的。这些优点使得几乎上任何一种程序设计都可以把C视为一种选择，而且C在系统程序设计方面特别有用，程序员可以用C写出速度快、结构紧凑的程序，并且可以把这些程序移值到其它系统上。在许多情况下，好的C程序可以在速度上用汇编语言书写的程序相媲美，而且具有易于维护和可读性好的优点。

C不包含执行诸如输入、输出、存贮分配、屏幕操作以及过程控制等任务的内部函数，程序员必须根据运行库存来执行这些任务。

这种设计增加了C的适应性和紧凑性，因为语文是相对地受到某些限制的，所以它不使用一个具体的程序设计模型。运行时的例程提供所需的支持，允许程序员在必需时使用它们，或者对它们进行裁剪以适应特定目的。

这种设计还把语言的特性和具体C语言实现的某些环境分离开来，这对期望写出移植性好的代码的程序员来说是一个帮助。语言的严格定义使得它不赖于任何操作系统或机器，同时，程序员也可以简单地添加依赖于系统的例程以充分地利用特定的机器。

C语言一些重要的性质如下：

.为从逻辑上和有效地控制程序流程以及鼓励结构程序设计方法的使用，C提供了全套的循环、条件控制。

.C提供了一组数量极多的运算符。许多C运算符对应于常用的机器指令，这为直接翻译成机器代码提供了方便。各种各样的运算符使得程序员能够以最少的编码清楚地说明不同类型的运算。

.C数据类型包括几种长度的整数以及单精度和双精度的浮点数。程序员可以定义更为复杂的数据类型，如数组和结构，以适应特殊程序的需要。

.C程序员可以说明指向变量和函数的指针。指向一个实体的指针对应于该实体的机器地址，巧妙地使用指针可以极大地提高程序的效率，因为指针使程序员以物理机器的方式访问实体。C还支持指针的算术运算，允许程序员直接存取和操作内存地址。

.C的预处理程序，即正文处理程序，在编译之前作用于文件的正文。对于C程序来说，其中最有用处的是，程序常数的定义、用更快的类似宏的函数调用的替换以及条件编译。预处理程序不只限于C文件，它也可以用于任何正文文件。

.C是一种灵活的语言，它把许多东西都留给程序员。为了这一目的，C在许多方面，例如类型转换中，使用了很少的限制条件。这常常是一种很好的机制，但是，为理解程序将怎样动作，程序员必须完全熟悉语言的定义。

1.1 C的元素

1.1.1 字符集

C程序中定义了两个字符集,C字符集和可表示字符集。C字符集包括字母、数字和与C编译程序有特殊含义的标点符号,C程序就是由C字符集中的字符组合成有意义的语句而构成的。

C字符集是可表示字符集的一个子集。可表示字符集包括所有字母、数字以及用户能以图形方式用一个字符表示出来的符号。字符集的范围取决于所使用的终端、控制台或字符设备的类型。

除了串直接量、字符常量和注解可以使用任何可表示字符以外,C程序只能使用C字符集中的字符。C字符集中的每个字符对C编译程序来说都有其确切的含义,当编译程序遇到对字符的错误使用或使用了不属于C字符集中的字符时,它将产生错误信息。

1. 字母和数字

C的字符集包括英语的大小写字母和10个阿拉伯数字。

大写英语字母:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

小写英语字母:

a b c d e f g h i j k l m n o p q r s t u v w x y z

10个阿拉伯数字:

0 1 2 3 4 5 6 7 8 9 这些字母和数字可以用来组成常、标识符和关键字。

C编译程序区分大小写字母。如果用小写的"a"代表一给定的项,那么就不能用大写的"A"来替代之,而必须用小写。

2. 空白字符

空格、制表符、制表符、跳行、回车符、换页、垂直制表符和换行符号都称为空白字符,因为输出时它们在字与字之间、行与行之间都起到类似于空格的功能。

除非把空白字符用的字符常量或串直接量中,否则C编译程序是忽略这些字符的,这就是说程序员可以使用额外的空白符白增加程序的可读性。注释也被视为空白字符。

3. 标点和空白字符

C字符集中的标点和特殊字符用于许多目的,从组织程序的正文到定义要求编译程序执行的任务或指定被编译的程序要完成的任务。

标点和特殊字符

字符	名字	字符	名字
,	逗号	!	惊叹号
.	点	|	竖线
;	分号	/	斜线
:	冒号	\	反斜线

?	问号	~	非
'	单引号	—	下划线
"	双引号	#	数字号
(	左括号	%	百分号
)	右括号	&	和号
[	左方括号	^	脱字号
]	右方括号	*	星号
{	左花括号	—	减号
}	右花括号	=	等于号
<	小于号	+	加号
>	大于号		

对于C编译程序来说，这些字符都有其特殊的含意。可表示字符集中那些未出现在该表中的标点字符仅能用在串直接量、字符常量和注解中。

4. 转义序列

转义序列是表示串直接量和字符常量中空白字符和非图形字符的特殊的字符组合。典型的用途是用它们一不指明动作，例如回车或制表，在终端和打印机上移动，并提供有特殊含义的特殊符号的字面表示，例如双引号(")字符。转义序列由反斜线后面跟以一个字母或数字的序列组成。

转义序列

转义序列	名字
\n	换行
\t	水平制表
\v	垂直制表
\b	退格
\r	回车
\f	换页
\'	单引号
\"	双引号
\\	反斜线
\ddd	ASCH 字符的八进制表示
\xdd	ASCH 字符的十六进制表示

如果反斜线之后的字符未出现在上表中，那么反斜线就被忽略过去。而该字符就是它的字面表示。例如，在串或字符中，"\c"就是表示"c"

序列"\ddd"和\"xdd"允许用三位八进制数字或两位十六进制数字给出 ASCII 字符集中的任何字符。例如退格符可以用"\010"和\x08"给出，ASCII 的空字符可以用"\0"或"x0"给出。

在八进制转义序列中只能出现八进制数字，并且至少要有一位数字，但可以小于三位数字，例如，退格字符可以由"\10"给出。类似地，十六进制序列应至少包括一位数字，但第二个数字可以省去，退格字符的十六进制转义字符可以写为"\x8"。但是，在串中使用八进制或十六进制转义序列时完整地给出转义序列则更为完全一些，否则，如果序列后紧跟着的字符是八进制数字或十六进制数字时，该字符就可能被解释成序列的一部分。

转义序列允许把非图形控制字符送到显示设备上，例如"\033"常用作终端或打印机控制命令的第一字符。

非图形字符必须用转义序列表示，这种字符出现在 C 程序中时将产生难以预料的结果。

用于引入转义序列的反斜线还可以在串或预处理程序定义中当作续行符使用。当换行符跟在反斜线之后时它将被忽略，而下一行被视为上一行的继续。

5. **运算符**

运算符是指明如何把值进行传送和赋值的特殊字符的组合，用户必须准确地按表中所列的那样使用运算符，多字符的运算符之间不能插入空白字符。运算符 sizeof 不包括在该表中，它由关键字而不是由符号组成。

运算符

运算符	名字
!	逻辑"非"
~	逐位求"反"
+	加
−	减
*	乘
/	除
%	求余数
<<	左移
>>	右移
<	小于
<=	小于等于
>	大于
>=	大于等于
==	等于
! =	不等于
&	逐位求"与"，求地址
|	逐位求"或"

	逐位求“异或”
&&	罗辑“与”
||	罗辑“或”
,	顺序求值
?:	条件式
++	递增
--	递减
=	简单赋值
+=	加赋值
-=	减赋值
*=	乘赋值
/=	除赋值
%=	求余数赋值
>>=	右移位赋值
<<=	左移位赋值
&=	逐位求“与”后赋值
|=	逐位求“或”后赋值
^=	逐位求“异或”后赋值

1.1.2 常量

常量是在程序中作为值使用的数字、字符或字符串，常量的值不因执行的环境不同而改变。

1 整数常量

整数常量表示一个整数值的十进制、八进制或十六进制数。十进制常量的形式为：

digits

其中 *digits* 是一个或多个十进制数字(从 0 到 9)。

八进制常量的形式为：

0*odigits*

其中 *ookigits* 是一个或多个八进制数字(从 0 到 7)，前导零是必须的。

十六进制常量的形式为：

0*xhdigits*

其中 *hdigits* 是一个或多个十六进制数字(从 0 到 9、以及从 a 到 f 和从 A 到 F)，前导零是必须的，而且必须跟以“x”。

整数常量中，数字间不能有空白字符出现。

整数常量

十进制常量	八进制常量	十六进制常量
10	012	0xa 或 0xA
132	0204	0x84
32179	076663	0x7db3 或 0x7DB3

整数常量总是表示正值。若需要负值，则可以把负号(—)放在常量之前表成一个表示负值的表达式，而负号被视为一个算术运算符。

每个整数常量根据它的值都有一个类型，常量的类型决定了它出现在表达式中或用以形成负常量时将进行的转换，十进制常量被认为是带符号的数值，并给以类型 int 或者 long。

八进制和十六进制常量也可指定为 *int* 或者 *long* 类型，但它们与其它有符号数不同，在类型转换中不进行符号扩充。

程序员可以告诉编译程序让一个常量具有 *long* 类型，方法是在常量后面附加上"1"或"L"。

长整数常量

十进制常量	八进制常量	十六进制常量
10L	012L	0xaL 或 0xAL
79 1	0115 1	0x4fl 或 0x4FL

2. 浮点常量

浮点数常量是表示一个有符号的实数的十进制数值，有符号的实数包括整数、小数和指数三部分。浮点数常量的形式为：

[*digits*][. *digits*][E[-]*digits*]

其中 *digits* 是一个或多个十进制数字从 0 到 9()，E(或 e)是指数的符号。小数点以前的数字(值的整数部分)或小数点以后的数字(值的小数部分)均可以省去，但二者不能同时省去。指数由指数符号后面跟以有可能为负值的整数值构成。仅当给出指数时才能省去小数点。不能用空白字符把常量中数字或字符隔开。

浮点数量总是表示正值，当需要负值时，可以把负号(—)置于常量之前形成一个具有负值的浮点常量表达式，而负号被视为算术运算符。

在上例中，列举了浮点常量和表达式的一些形式：

15.75

1.575E1

1575e—2

—0.0025

—2.5e—3

2.5E—4

浮点数量的整数的部分可以省去，举例如下：

.75

.0075e2

-.125

-.175E—2

所有浮点数常量都具有类型 *double*。

3. 字符常量

字符常量是括在单引号之内的字母、数字、标点符号或转义序列，字符常量的值就是它本身。字符常量中不能出现多于一个的字符或转义序列。

字符常量的形式为：

'char'

其中，char 是可表示字符集中(包括任何转义序列)除了单引号(')或反斜线(\)和换行符以外的任何字符。为了把单引号或反斜线用字符常量，应该在它前面再加一个反斜线，而用转义序列'\n'来表示换行符。

字符常量举例

常数	值
'a'	小写字母 a
'?'	问号
'\b'	退格字符
'*xlb*'	ASCII 的 ESC 字符
'\''	单引号
'\\'	反斜线

字符常量的类型为 *char*，在类型转换时进行符号扩充。

4. 串直接量

串直接量是括在双引号之内的字母、数字和符号的序列。串直接量为一个字符数组，数组的每一个元素是一个字符的值。

串直接量的形式为：

"characters"

其中，*characters* 是可表示字符集中除双引号(")、反斜线(\)和换行符以外的一个或多个字符。为了在串中使用换行符，可以在换行之前加一个反斜线，反斜线使得换行符被略去，这就允许一个串直接量可以占据多行的位置。例如，串：

"Long strings can be bro\

ken into two pieces."

等同于串：

"Long strings can be broken into two pieces."

为了在串中使用双引号或反斜线，可以在它们之前加一个反斜线。如下例所示：

"This is a string literal."

"Enter a number between 1 and 100\n Or press Return."