

Linux 风暴系列

Linux & Unix C 程序员参考大全

森林图书工作室 编



国防工业出版社



TP316.81

157

Linux & Unix C 程序员 参考大全

森林图书工作室 编

国防工业出版社

·北京·

图书在版编目(CIP)数据

Linux&Unix C 程序员参考大全/森林图书工作室编. 北京:国防工业出版社,2001.9

(Linux 风暴系列)

ISBN 7-118-02559-3

I .L... II . 森... III .C 语言 - 程序设计 IV .TP312

中国版本图书馆 CIP 数据核字(2001)第 029274 号

国防工业出版社出版发行

(北京市海淀区紫竹院南路 23 号)

(邮政编码 100044)

北京奥隆印刷厂印刷

新华书店经售

*

开本 787×1092 1/16 印张 27 $\frac{3}{4}$.631 千字

2001 年 9 月第 1 版 2001 年 9 月北京第 1 次印刷

印数:1—3000 册 定价:38.00 元

(本书如有印装错误,我社负责调换)

出版说明

Linux 是计算机发展历史上的独特现象。虽然它滥觞于一位普通大学生的灵感与才思,却已成为当今最为流行的免费操作系统。对很多人来说, Linux 是一个谜, 免费的东西怎么会变得如此有价值? 事实上 Linux 的确稳定而富有竞争力, 许多大学与研究机构都使用 Linux 完成他们的日常计算任务, 同时, Linux 也逐渐成为各公司服务器的首选操作系统, 许多公司将它用于邮件服务器或 WWW 服务器, 用于 DNS、路由和防火墙等。相信在不久的将来, 人们在家用 PC 上也会广泛使用 Linux。

近年来 Linux 在中国也有了很大的发展, 特别是随着 Internet 的普及, Linux 的发展更是如火如荼, 国内很多城市都成立了 Linux 俱乐部, Linux 发烧友队伍日益壮大, 从而掀起了一场 Linux 风暴。

正是在 Linux 蓬勃发展之际, 我们组织编写了本套丛书, 旨在为一部分读者解开 Linux 成功之谜, 更为 Linux 在中国的普及和发展贡献一份力量。

国防工业出版社计算机编辑室

前 言

近年来,Linux 操作系统的发展非常迅速,特别是随着 Internet 的普及,Linux 的发展更是迅猛如潮,国内的很多城市都成立了 Linux 俱乐部。同时,Linux 操作系统也逐渐成为各公司服务器的首选操作系统。

但是目前还有很多因素制约着 Linux 进入寻常百姓家,首要的是应用程序太少,与 Windows 相比,Linux 应用程序显得品种不全、内容不够丰富。虽说很多公司也开发了一些如 Start Office 之类的办公软件和其他一些工具软件,但是还是显得有点凤毛麟角,而且显得不够成熟。而适合国人实用的软件更是少而又少了。另一个制约的因素就是驱动程序太少,很多新的硬件都得不到及时的支持,使 Linux 使用起来很是不方便。另外就是管理 Linux 需要有很高的技术水平,由于 Linux 是免费的操作系统,没有具体的厂商来维护,必须靠使用者自己,而其源代码是公开的,所以管理人员最好能读懂源代码并能熟练地编程,熟悉并能修改各种配置文件,这样管理起来就非常方便。

Linux 源代码都是 C 语言编写的,C 语言内容丰富、功能强大,是程序人员编程的理想语言。所以无论是阅读 Linux 程序原码还是自己动手编写应用软件,都需要一本全面介绍 Linux 下 C 语言函数库的工具书,由于 C 语言的可移植性,本书介绍的 C 函数大多数在 Unix 系统下也同样适用。

读者对象

有一定编程经验,对用 C 语言编程比较熟悉,尤其是 ISO 标准的 C 语言。

内容提要

第 1 章主要介绍用于监测和报告错误信息的函数和宏,通过这些函数和宏读者可以方便地确定错误所在以及错误原因,以方便程序的调试。

第 2 章介绍对内存进行分配的函数。

第 3 章主要介绍对字符和字符串进行处理的函数,包括字符的判断和转换、字符串的复制比较等。

第 4 章介绍对用于输入和输出的文件和流进行操作的函数。

第 5 章主要介绍执行在文件描述符上的低级输入输出的函数。

第 6 章主要介绍对文件系统内目录和文件的名称、属性等进行操作的函数。

第 7 章主要介绍对用于进程间通信的管道和 FIFO 进行操作的函数。

第 8 章主要介绍对用于进程间通信的 socket 进行操作的函数。

第 9 章主要介绍对终端设备进行操作的函数,可以设置终端的属性、输入模式等。

第 10 章主要介绍用于数学计算的函数,包括三角函数、指数函数以及随机数的创建等。

第 11 章介绍进行搜索、排序和模式匹配等的函数。

第 12 章主要介绍对日期和时间进行操作的函数,包括日期和时间的设置、更改、警报的设置以及系统时间资源的处理等。

第 13 章主要介绍对字符集进行扩展操作的函数。

第 14 章主要介绍信号的类型以及对信号的处理函数等。

第 15 章主要介绍如何创建一个新的进程、终止一个进程以及在进程间传递各种环境变量的函数。

第 16 章主要介绍对作业进行操作的函数。

第 17 章介绍包含系统信息的数据库以及 NSS 的概念。

第 18 章介绍对系统内用户和组进行操作的函数,包括读取或者修改用户和组的 ID 号、查询指定的用户或者组等。

第 19 章主要介绍用于获取系统信息和系统限制参数的函数。

第 20 章主要介绍一些工具,如进行程序的内部检查、接受数目可变的变量以及对数据类型的检测等。

附录 A 函数名和宏索引列出了本书所介绍的所有函数以及定义的宏。

附录 B 变量和常量宏索引列出了本书所介绍的系统定义的当作变量或者常量来处理的宏。

本书由森林图书工作室组织编写,其中主要由程治安主持,另外参加编写工作的还有林振宁、黄建森、李春鹤、冯曙红、罗世兰、袁军、林世永、郑清初、黄重阳、刘浪、岑进华、郑国鸿、黄林、岑进炎、康拥红、郑吉林、许晓春、陈良程、郑宝林、梁任、蒙文荣、梁广钊、刘淑英等。本书在编写过程中,得到了国防工业出版社计算机编辑室的许多支持和帮助,在此表示诚挚的谢意。

由于时间仓促且编写者的经验和水平有限,书中难免有不妥之处,恳请读者批评指正。

目 录

第 1 章 错误报告	1		
1.1 错误检查	1		
1.2 错误代码	1		
1.3 错误信息	7		
strerror()	7		
strerror_r()	7		
perror()	7		
第 2 章 内存分配	10		
2.1 动态内存分配概念	10		
2.2 无限制的分配	10		
malloc()	10		
free()	12		
cfree()	12		
realloc()	13		
calloc()	14		
mallopt()	15		
mcheck()	15		
mprobe()	16		
mallinfo()	17		
2.3 obstack	18		
obstack_init()	19		
obstack_alloc()	19		
obstack_copy()	20		
obstack_copy0()	20		
obstack_free()	20		
obstack_blank()	21		
obstack_grow()	21		
obstack_grow0()	21		
obstack_lgrow()	22		
obstack_ptr_grow()	22		
obstack_int_grow()	22		
obstack_finish()	22		
obstack_object_size()	23		
obstack_room()	23		
obstack_lgrow_fast()	23		
obstack_ptr_grow_fast()	23		
obstack_int_grow_fast()	24		
obstack_blank_fast()	24		
obstack_base()	25		
obstack_next_free()	25		
2.4 变量式自动存储	25		
alloca()	26		
2.5 重新分配	27		
r_alloc()	27		
r_alloc_free()	27		
r_re_alloc()	28		
第 3 章 字符及字符串处理	29		
3.1 字符处理	29		
islower()	29		
isupper()	29		
isalpha()	30		
isdigit()	30		
isalnum()	30		
isxdigit()	30		
ispunct()	31		
isspace()	31		
isblank()	31		
isgraph()	31		
isprint()	32		
iscntrl()	32		
3.2 字符转换	32		
tolower()	33		

toupper()	33	strcspn()	50
toascii()	33	strpbrk()	50
_tolower()	33	strtok()	50
_toupper()	34	Strtok_r()	51
3.3 字符串处理	34	strsep()	52
strlen()	35	第4章 输入/输出	53
memcpy()	35	4.1 文件	53
memmove()	36	clearerr()	54
memccpy()	36	feof()	54
memset()	36	ferror()	55
strcpy()	37	ftell()	55
strncpy()	37	fseek()	55
strdup()	37	rewind()	56
strndup()	38	fgetpos()	57
stpncpy()	38	fsetpos()	57
stpncpy()	38	fflush()	57
strdupa()	39	setvbuf()	58
strndupa()	40	setbuf()	58
strcat()	40	setbuffer()	59
strncat()	41	setlinebuf()	59
bcopy()	41	4.2 标准流输入输出	59
bzero()	42	fopen()	60
memcmp()	42	freopen()	61
strcmp()	43	fclose()	61
strcasecmp()	43	fcloseall()	62
strncasecmp()	43	fputc()	62
strncmp()	44	putc()	63
bcmp()	44	putchar()	63
strcoll()	44	fputs()	63
strxfrm()	45	puts()	64
memchr()	47	putw()	64
strchr()	47	fgetc()	64
index()	48	getc()	65
strchr()	48	getchar()	65
rindex()	48	getw()	66
strstr()	48	getline()	66
memmem()	49	getdelim()	67
strspn()	49	fgets()	67

gets ()	68	dup2 ()	94
ungetc ()	68	5.2 低级输入输出的宏	95
fread ()	69	5.2.1 打开文件操作方式	95
fwrite ()	69	5.2.2 输入输出操作模式	96
4.3 格式化输出	70	5.2.3 文件状态标志	97
printf ()	72	5.2.4 文件锁	97
fprintf ()	72	5.2.5 中断驱动输入	99
sprintf ()	73	第6章 文件系统	101
snprintf ()	73	6.1 目录操作	101
asprintf ()	74	getcwd ()	101
obstack _ printf ()	74	getwd ()	102
vprintf ()	75	chdir ()	102
vfprintf ()	75	6.2 访问目录	103
vsprintf ()	75	6.2.1 目录项的格式	103
vsnprintf ()	75	6.2.2 打开目录流	104
vasprintf ()	76	opendir ()	104
obstack _ vprintf ()	76	6.2.3 读和关闭目录流	104
parse _ printf _ format ()	77	readdir ()	104
4.4 格式化输入	79	readdir _ r ()	105
scanf ()	80	closedir ()	105
fscanf ()	82	6.2.4 目录流的随机进入	106
sscanf ()	82	rewinddir ()	106
vscanf ()	82	telldir ()	106
vfscanf ()	83	seekdir ()	107
vsscanf ()	83	6.2.5 扫描目录内容	107
第5章 低级输入输出	84	scandir ()	107
5.1 低级输入输出函数	85	alphasort ()	107
open ()	85	6.3 硬链接	108
creat ()	86	link ()	108
close ()	86	6.4 符号链接	109
read ()	86	symlink ()	109
write ()	87	readlink ()	110
lseek ()	88	6.5 删除文件	111
fdopen ()	90	unlink ()	111
fileno ()	90	rmdir ()	112
select ()	90	remove ()	112
fcntl ()	92	6.6 重命名文件	112
dup ()	94	rename ()	112

6.7 创建目录	113	pclose ()	133
mkdir ()	113	7.3 FIFO 特殊文件	134
6.8 文件属性	114	mkfifo ()	134
6.8.1 文件属性的意义	114	7.4 微量的管道输入/输出	134
6.8.2 读取文件属性	116	第 8 章 socket	135
stat ()	116	8.1 socket 概念	135
fstat ()	116	8.2 socket 地址	136
lstat ()	116	bind ()	138
6.8.3 查看文件类型	117	getsockname ()	138
6.8.4 文件所有者	118	8.3 Internet 名称空间	139
chown ()	118	Inet _ aton ()	144
fchown ()	119	inet _ addr ()	144
6.8.5 文件模式的访问权限位	119	inet _ network ()	144
6.8.6 访问文件	120	inet _ ntoa ()	144
6.8.7 分配文件权限	121	inet _ makeaddr ()	145
umask ()	121	inet _ lnaof ()	145
getumask ()	121	inet _ netof ()	145
chmod ()	122	inet _ pton ()	145
fchmod ()	122	inet _ ntop ()	146
6.8.8 检查文件的访问权限	123	gethostbyname ()	146
access ()	123	gethostbyname2 ()	146
6.8.9 文件时间	124	gethostbyaddr ()	147
utime ()	124	sethostent ()	147
utimes ()	125	gethostent ()	147
6.9 特殊文件的生成	125	endhostent ()	148
mknod ()	126	getservbyname ()	148
6.10 临时文件	126	getservbyport ()	148
tmpfile ()	126	setservent ()	148
tmpnam ()	127	getservent ()	149
tmpnam _ r ()	127	endservent ()	149
tempnam ()	127	htons ()	149
mktemp ()	128	ntohs ()	149
mkstemp ()	128	htonl ()	150
第 7 章 管道和 FIFO	130	ntohl ()	150
7.1 创建管道	130	getprotobyname ()	150
pipe ()	130	getprotobyname ()	150
7.2 建立子进程的管道	132	setprotoent ()	151
popen ()	132	getprotoent ()	151

endprotoent()	151	9.4.7 本地模式	183
8.4 打开和关闭 socket	153	9.4.8 线速度	184
socket()	153	cfgetospeed()	185
shutdown()	153	cfgetispeed()	185
socketpair()	154	cfsetospeed()	185
connect()	155	cfsetispeed()	185
listen()	156	cfsetspeed()	186
accept()	156	9.4.9 特殊字符	186
getpeername()	157	9.4.10 非规范输入	190
send()	157	cfmakeraw()	191
recv()	158	9.5 线控制函数	191
8.5 数据包操作	166	tcsendbreak()	191
sendto()	166	tcdrain()	192
recvfrom()	167	tcflush()	192
8.6 socket 选项	170	tcflow()	193
getsockopt()	172	第 10 章 数学函数	196
setsockopt()	172	10.1 定义域和值域错误	196
8.7 网络数据库	173	10.2 三角函数	197
getnetbyname()	173	sin()	197
getnetbyaddr()	173	cos()	197
setnetent()	173	tan()	197
getnetent()	174	10.3 反三角函数	198
endnetent()	174	asin()	198
第 9 章 低级终端接口	175	acos()	198
9.1 识别终端	175	atan()	198
isatty()	175	atan2()	198
ttyname()	175	10.4 幂函数和对数函数	199
9.2 I/O 队列	175	exp()	199
9.3 输入的两种模式	176	log()	199
9.4 终端模式	176	log10()	199
9.4.1 终端模式的数据类型	176	pow()	200
9.4.2 终端模式函数	177	sqrt()	200
tcgetattr()	177	cbrt()	200
tcsetattr()	178	hypot()	200
9.4.3 终端模式的正确设置	178	expm1()	201
9.4.4 输入模式	179	log1p()	201
9.4.5 输出模式	181	10.5 双曲函数	201
9.4.6 控制模式	181	sinh()	201

cosh()	201	10.12 整数的除法	210
tanh()	202	div()	211
asinh()	202	ldiv()	211
acosh()	202	10.13 数字的解析	211
atanh()	202	10.13.1 整数的解析	211
10.6 伪随机码	202	strtol()	211
10.6.1 ISO C 随机码函数	203	strtoul()	212
rand()	203	strtoq()	212
srand()	203	strtoll()	213
10.6.2 BSD 随机码函数	203	strtouq()	213
random()	204	strtoull()	213
srandom()	204	atol()	213
initstate()	204	atoi()	213
setstate()	204	10.13.2 浮点数的解析	215
10.7 非数值	205	strtod()	215
10.8 浮点数的判断	205	strtodf()	215
isinf()	205	strtold()	216
isnan()	205	atof()	216
finite()	205	第 11 章 搜索、排序和模式匹配	217
infnan()	206	11.1 搜索和排序	217
10.9 绝对值	206	bsearch()	217
abs()	206	qsort()	217
labs()	206	11.2 模式匹配	221
fabs()	207	fnmatch()	222
cabs()	207	glob()	223
10.10 标准函数	207	regcomp()	225
frexp()	207	regexec()	226
ldexp()	208	regfree()	227
scalb()	208	regerror()	227
logb()	208	wordexp()	228
copysign()	208	wordfree()	229
10.11 取整和求余函数	209	第 12 章 日期和时间	232
ceil()	209	12.1 处理器时间	232
floor()	209	12.1.1 查询基本的 CPU 时间	232
rint()	209	clock()	233
modf()	209	12.1.2 详细的查询流逝的 CPU	
fmod()	210	时间	233
drem()	210	times()	233

12.2 日历时间	234	mblen()	257
12.2.1 单个日历时间	234	mbtowc()	257
difftime()	234	wctomb()	258
time()	235	13.2 地区化和国际化	261
12.2.2 高精度的日历时间	235	setlocale()	264
gettimeofday()	236	localeconv()	266
settimeofday()	236	第 14 章 信号处理	268
adjtime()	237	14.1 标准信号	268
12.2.3 分解的时间	237	14.1.1 程序错误信号	268
localtime()	238	14.1.2 中断信号	270
gmtime()	238	14.1.3 报警信号	271
mktime()	239	14.1.4 异步 I/O 信号	271
12.2.4 格式化日期和时间	239	14.1.5 作业控制信号	272
asctime()	239	14.1.6 操作错误信号	273
ctime()	239	14.1.7 杂项信号	273
strftime()	240	14.1.8 信号信息	274
12.2.5 用 TZ 指定时区	243	strsignal()	274
12.2.6 时区的函数和变量	245	psignal()	274
tzset()	245	14.2 指定信号行为	275
12.3 设置警报	246	14.2.1 基本信号处理	275
setitimer()	247	signal()	275
getitimer()	247	ssignal()	277
alarm()	248	14.2.2 高级信号处理	277
12.4 等待状态	248	sigaction()	277
sleep()	248	14.2.3 sigaction 函数的标志	279
12.5 资源的使用	249	14.3 产生信号	279
getrusage()	249	14.3.1 给自己发信号	282
12.6 限制资源的使用	251	raise()	282
getrlimit()	251	gsignal()	283
setrlimit()	251	14.3.2 给其他进程发送信号	283
12.7 进程的权限	253	kill()	284
getpriority()	253	killpg()	285
setpriority()	253	14.4 阻塞信号	286
nice()	254	14.4.1 信号集	286
第 13 章 扩展字符	255	sigemptyset()	287
13.1 扩展字符串的转换	255	sigfillset()	287
mbstowcs()	256	sigaddset()	287
wcstombs()	256	sigdelset()	288

sigismember()	288	abort()	315
sigprocmask()	288	15.3.5 内部中断	315
14.4.2 检查未决信号	290	_exit()	315
sigpending()	290	15.4 运行一个命令	315
14.5 等待信号	292	system()	316
pause()	292	15.5 生成进程概念	316
sigsuspend()	293	15.6 进程的识别	317
14.6 使用分离的信号栈	295	getpid()	317
sigaltstack()	296	getppid()	317
sigstack()	296	15.7 创建进程	317
14.7 BSD 信号处理	297	fork()	317
sigvec()	298	vfork()	318
siginterrupt()	298	15.8 执行一个文件	318
sigblock()	298	execv()	319
sigsetmask()	298	execl()	319
sigpause()	299	execve()	319
第 15 章 进程的启动和终止	300	execle()	319
15.1 程序变量	300	execvp()	320
15.1.1 程序变量语法惯例	300	execlp()	320
15.1.2 程序选项的解析	301	15.9 进程的完成	321
getopt()	302	waitpid()	321
15.1.3 长选项的解析	304	wait()	322
getopt_long()	305	wait4()	323
15.1.4 子选项的解析	307	15.10 进程完成的状态	323
getsubopt()	308	15.11 BSD 的处理等待函数	324
15.2 环境变量	310	wait3()	324
15.2.1 环境变量的访问	310	第 16 章 作业控制	326
getenv()	310	16.1 作业控制的概念	326
putenv()	311	16.2 可选择的作业控制	327
15.2.2 标准环境变量	311	16.3 进程的控制终端	327
15.3 程序的终止	312	16.4 对控制终端的访问	327
15.3.1 正常中断	312	16.5 被孤立的进程群	328
exit()	312	16.6 作业控制外壳的执行	328
15.3.2 Exit 状态值	313	16.6.1 外壳的数据结构	328
15.3.3 退出时的清除	313	16.6.2 外壳的初始化	329
atexit()	314	16.6.3 作业的启动	330
on_exit()	314	16.6.4 前台和后台作业	331
15.3.4 程序的异常中断	314	16.6.5 停止和中断作业	332

16.6.6 继续停止的作业	336	setuid()	350
16.7 作业控制函数	336	setreuid()	351
16.7.1 识别控制终端	336	18.7 设置组 ID 号	351
ctermid()	336	setgid()	351
16.7.2 进程群函数	336	setregid()	352
setsid()	337	setgroups()	352
setpgid()	337	initgroups()	352
setpgrp()	338	18.8 允许或者禁止 setuid 的访问	353
16.7.3 控制终端的访问函数	338	18.9 编写 setuid 程序的提示	355
tcgetpgrp()	338	18.10 识别登录的用户	355
tcsetpgrp()	339	getlogin()	356
第 17 章 系统数据库和名称服务		cuserid()	356
转换	340	18.11 用户数据库	356
17.1 NSS 基本概念	340	18.11.1 描述用户的数据结构	356
17.2 NSS 配置文件	341	18.11.2 查找用户	357
17.2.1 在 NSS 配置文件内的		getpwuid()	357
服务程序	341	getpwuid_r()	357
17.2.2 NSS 配置中的操作	342	getpwnam()	358
17.2.3 NSS 配置文件的注意点	342	getpwnam_r()	358
17.3 NSS 模块的内部	343	18.11.3 扫描所有用户列表	358
17.3.1 NSS 模块的命名方法	343	fgetpwent()	358
17.3.2 NSS 模块内函数的接口	343	fgetpwent_r()	359
17.4 NSS 的扩展	344	setpwent()	359
17.4.1 向 NSS 增加一个服务		getpwent()	359
程序	345	getpwent_r()	360
17.4.2 NSS 模块函数的内部	345	endpwent()	360
第 18 章 用户和组	347	18.11.4 写入一个用户项	360
18.1 用户和组 ID 号	347	putpwent()	360
18.2 进程的角色	347	18.12 组数据库	361
18.3 更改进程角色的原因	348	18.12.1 描述组的数据结构	361
18.4 应用程序如何更改角色	348	18.12.2 查找组	361
18.5 读取进程的角色	348	getgrgid()	361
getuid()	349	getgrgid_r()	361
getgid()	349	getgnam()	362
geteuid()	349	getgnam_r()	362
getegid()	349	18.12.3 扫描所有组列表	362
getgroups()	349	fgetgrent()	362
18.6 设置用户 ID 号	350	fgetgrent_r()	363

setgrent()	363	19.8 文件系统容量的限制	378
getgrent()	363	19.9 文件支持的附加特征	379
getgrent_r()	364	19.10 文件系统限制的最小值	379
endgrent()	364	19.11 pathconf 的使用	380
18.13 网络组数据库	364	pathconf()	380
18.13.1 网络组数据	364	fpathconf()	380
18.13.2 查找网络组	365	19.12 工具程序的容量限制	381
setnetgrent()	365	19.13 工具限制的最小值	382
getnetgrent()	365	19.14 字符串值参数	382
getnetgrent_r()	365	confstr()	382
endnetgrent()	366	第 20 章 函数库内的 C 语言工具	384
18.13.3 判断网络组的从属关系	366	20.1 内部一致性的检查	384
innetgr()	366	20.2 variadic 函数	385
第 19 章 系统信息和配置参数	369	20.2.1 使用 variadic 函数的原因	385
19.1 主机识别	369	20.2.2 variadic 函数的定义和 使用	385
gethostname()	369	20.2.3 旧类型的 variadic 函数	388
sethostname()	370	20.3 空指针常量	388
gethostid()	370	20.4 重要的数据类型	389
sethostid()	370	20.5 数据类型的检测	389
19.2 识别硬件/软件类型	371	20.5.1 计算一个整型数据类型的 宽度	389
uname()	371	20.5.2 整型数的范围	390
19.3 一般的容量限制	371	20.5.3 浮点型的宏	391
19.4 所有的系统选项	373	20.5.4 结构内元素偏移量的 检测	394
19.5 支持 POSIX 的版本	374	附录 A 函数名和宏索引	396
19.6 sysconf 的使用	374	附录 B 变量和常量宏索引	411
19.6.1 sysconf 的定义	374		
sysconf()	374		
19.6.2 sysconf 参数的常量	375		
19.7 一般容量限制的最小值	377		

第1章 错误报告

有时程序需要检查一些错误情况,在C函数库里有一些函数是用来监测和报告这些错误情况的,例如:当你打开一个输入文件时,需要确认文件打开是否正确,如果调用库函数时出现错误,则需显示错误信息或采取其他相应的措施。

本章将详细介绍错误报告工具的内容。如果你的程序要使用这个工具(facility),必须包含头文件'errno.h'。

1.1 错误检查

多数库函数在调用出错时将返回一个特定值来说明它们出错的情况。这个特定值一般为-1、空指针或一个专门定义的常量(如EOF)。但是这些返回值仅仅只告诉你有一个错误发生了,如果你想查看发生的是何种错误,你需要查看存放在内部变量errno内的错误代码。这个变量在头文件'errno.h'有说明。

变量:volatile int errno

当程序开始执行时,变量errno的初始值为0。多数库函数在检测到错误时,将把他设为非零值。当运行正常时,这些函数不改变errno的值;这样在调用成功后,errno的值是一个无意义的零值,你也不需要(errno)来确定调用是否失败。如果失败,你可以检查(errno)。

一些库函数在调用其他可能存在错误的库函数时,将把(errno)设为非零值。当程序显示出错时,你需要查看有哪些库函数能改变(errno)的值。

 **注意** ISO C 将(errno)定义为“可修改的左值”(modifiable lvalue),允许将它作为一个宏来调用,而非仅作为一个变量。例如,它的扩展可以包括函数调用,如*_errno()。

有少数库函数,如sqrt和atan,在出现错误时仍可以返回一个完美的值,同时对(errno)值进行设置。对于这些函数,如果你想检查是否有错误发生,建议你在函数调用前先将(errno)设置为零,在函数调用后再检查它的值。

所有的错误代码都有其符号名(symbolic name),它们作为宏在'errno.h'内定义。错误代码符号名由'E'开始,由大写字母或数字组成。

除了EWOULDBLOCK和EAGAIN的值是相同的外,其他所有的错误代码值为正整数,且各不相等。对于具有不同值的符号名,可以在判断语句中用作标识符。

1.2 错误代码

错误代码宏定义在头文件'errno.h'内。它们都可以扩展为整型常量值。有些错误代码