IBM PC／AT
硬件和XENIX系统资料汇编之八

系统调用和子程序

胡先祥
梁赓 译
孙玉方 校

北京科海总公司培训中心
中国科学院软件研究所

一九八七年四月

IBM PC/AT硬件和XENIX系统资料汇编

（之八）

系统调用和子程序

胡先祥
梁赓 译
孙玉方 校

北京科海总公司培训中心
中国科学院软件研究所
一九八七年四月

编辑：科海培训中心教材部
发行：科海培训中心资料组
地址：北京2725信箱 科海培训中心
资料组
（北京海淀路332路黄庄站旁）
印刷：河北省蔚县印刷厂

编 者 序

IBM PC已从PC、PC/XT推进到PC/AT。PC/AT以Intel80286为主CPU，具有丰富的硬件资源。鉴于目前DOS系统基本上是一种单用户系统，许多硬件资源未得到充分利用，许多用户都要求在PC/AT上配备多用户多任务的XENIX系统。

XENIX系统是UNIX系统在以Intel为主CPU的微机上的实现，该系统由Microsoft公司开发。目前在PC/AT上运行的XENIX相当于UNIX的System Ⅲ或System Ⅴ。

为了更好地在国内推广PC/AT及其兼容机，中国科学院软件研究所在其雄厚的技术力量基础上，积多年研究、开发UNIX系统之经验，开发成功多种XENIX中西文信息处理系统并移植到几乎所有PC/AT的兼容机和部分386机上。为了更好地推广XENIX中英文信息处理系统，科海培训中心和中国科学院软件研究所组织了一批专家和技术人员，收集并编译整理了有关XENIX及IBM PC/AT的全部技术资料。

这些资料包括以下几类：

Ⅰ. IBM PC/AT硬件资料，包括硬件安装及组装手册、技术手册和维护手册。

Ⅱ. XENIX基本系统的安装、基本用户指南、命令参考手册、系统管理手册和直观shell手册。

Ⅲ. XENIX开发系统方面的软件开发手册、库函数程序员手册、系统调用和子程序手册。

Ⅳ. XENIX系统上运行的汇编语言和各种高级语言（C、Fortran、Cobol、Basic）的用户指南和参考处理手册。

Ⅴ. XENIX正文格式化手册。

Ⅵ. XENIX系统上配备的最新版本INFORMIX和UNIFY数据库管理系统用户及参考手册。

Ⅶ. 中西文兼容的C—XENIX系统安装和基本使用手册。

全套资料约400万字，分装成20本。

全书的主要译校任务由中国科学院软件研究所的专家、技术人员承担，科海培训中心负责编辑、印刷和发行工作。

由于时间仓促，本资料汇编中必有不少错漏之处，敬请读者批评指正，以便再版时更正。

主编 孙玉方

董洪皋

目　录

NITRO (S)

名字

intro——介绍系统服务程序，库子程序和错误号。

句法

#include <errno.h>

说明

本节描述了所有系统服务程序，包括在操作系统核心中可用的全部子程序和系统调用。这些子程序可以看作是C程序调用的标准库libc的一部分。其它的子程序可以在别的库中得到。在8086/88和286系统中，都提供小，中，大模式程序的版本（即：每个系统都有一个库）。

在程序中用标准库libc之外的子程序，必须要联接相应的库。这一点是通过给编译程序或联接程序说明-lname来实现的，这里name是下面列出的名字。例如-lm 和-ltermcap 说明之后，联接程序则从这些库中查找子程序，联接成目标模块，可用的库名如下：

c	标准库，包括所有系统调用接口，标准I/O子程序，和其他通用目的服务程序。
m	标准数学库。
termcap	存取描述终端特征数据库termcap的子程序库。
curses	屏幕和光标操作子程序库。
dbm	数据库管理子程序库。
x	标准xenix库。

操作系统核心中的大多数服务程序都有一个或多个错误返回。一个错误条件是由返回一个非所希望的值来说明的，通常是-1。详细的说明见下面单项描述。

一个错误值总是在外部变量errno中。成功的调用不清errno，所以只有在表示有错之后才能去测试它。

每个系统调用描述中并没有列出所有可能的错误号，因为许多错误号对大多数调用都一样。下面是错误号及在〈error.h〉中定义的名字的一组完整的清单：

1. EPERM 非文件主

   本错误表示试图以除文件主或超级用户之外的禁止方式修改文件。如果普通用户想做只有超级用户才能做的事，也返回本错误。

2. ENOENT 无此文件或目录

   如果指定了文件名，文件应该存在而又不存在时发生此错误。路径名中的某一目录不存在时也发生此错误。

3. ESRCH 无此进程

   根据kill或ptrace中pid说明的进程号找不到进程时出此错。

4. EINTR 中断系统调用

   在系统调用过程中，用户选择了捕俘之后则发生一个异步信号（如中断或退出）。如果处理了这个信号之后还要继续执行，则出现返回这个错误条件的中断系统调

用。

5. EIO I/O错误

物理I/O错误。在实际用物理设备输入输出的调用时，有可能发生此错。

6. ENXIO 无此设备或地址

在表示子设备(此时它不存在或超出设备限制之外)的特别文件上进行输入/输出。如：磁带驱动器没联机，或磁盘没有放入驱动器都会发生这种错误。

7. E2BIG 变量表太长

表示exec类的变量表长于5120字节。

8. ENOEXEC 执行（exec）格式错

希望执行一个文件，尽管其允许方式正确，但不是以一个合法的幻数开始（见a.out (F)）。

9. EBADF 坏文件号

或是文件描述字指向一个没打开的文件，或是读（写）一个只能写（读）的文件。

10. ECHILD 无子进程

进程执行wait，但又没有子进程，或没有可等待的子进程。

11. EAGAIN 无多个进程

系统进程表满或用户不允许创建多个进程而又执行了fork。

12. ENOMEM 没有足够的空间

执行exec或sbrk时，程序要求的空间大于系统提供的空间，这不是一个临时条件，最大空间大小是一个系统参数。如果正文段、数据段和栈段要求太多的段寄存器，或者在fork时没有足够的对换（SWAP）空间也会发生此错。

13. EACCES 否定允许字

试图以保护系统的禁止方式存取文件。

14. EFAULT 坏地址

在试图用一系统调用的某个参数时系统遇到一个硬件错。

15. ENOTBLK 要求块设备

在要求块设备的地方，如：mount，遇到非块设备。

16. EBUSY 设备忙

试图安装一个已经安装的设备，或试图拆卸一个有活动文件（打开的文件，当前目录，安装有文件，活动正文段）的设备。如果试图开始计帐，而以前已经开始了则也发生错。

17. EEXIST 文件存在

一个存在的文件被不适合的上下文牵涉到，如：link。

18. EXDEV 跨设备联接

试图联接另外设备上的文件。

19. ENODEV 无此设备

试图对一设备用一条不适当的系统调用，如读只写设备。

20. ENOTDIR 没有这样的目录

在要求目录例如：在路径前缀中或作为chdir（s）的参数的地方给出了非目录。

21. EISDIR 是目录

试图写一目录。

22. EINVAL 不可用的参数

某些不可用的参数（如：拆卸一台没有安装的设备；kill用在signal中没有定义的信号；读写文件时lseek生成一个负指针）。本手册中（S）项描述的数学函数也置此错。

23. ENFILE 文件表溢出

打开文件系统表满，暂时不能接收open操作。

24. EMFILE 打开文件太多

同一时刻一个进程不能有多于20个文件描述字。

25. ENOTTY 无打印机

26. ETXTBSY 正文文件忙

试图执行一个正在为读/写打开的纯过程程序。试图对正在执行的纯过程程序作写打开时也会出错。

27. EFBIG 文件太长

文件大小超过规定的最大文件大小（1,082,201,088字节）或ULIMIT，见ulimit（S）。

28. ENOSPC 设备无剩余空间

在写一个普通文件时，设备上没有空闲的空间。

29. ESPIPE 非法定位

对管道执行lseek。

30. EROFS 只读文件系统

试图修改按只读条件安装的设备的文件或目录。

31. EMLINK 联接文件太多

试图联接多于最大值（1000）的文件到一个目标文件。

32. EPIPE 断开管道

试图向一个无进程读数据的管道写。这个条件一般生成一个信号，如果信号被忽略，则返回本错误。

33. EDOM 数学变量超出函数的域

数学包中的函数变量超出函数的域。

34. ERANGE 数学结果不能表示

数学包中函数的值不能用机器的精度表示。

35. EUCLEAN 文件系统需要清理

试图安装（mount（S））一个文件系统，而它的超级块没有标记清理。

36. EDEADLOCK 应该死锁

试图锁住一个文件区时，将引起两个进程为争夺该区的控制权而产生死锁。

37. ENOTNAM 非有名文件

creatsem（S），opensem（S），waitsem（S）或sigsem（S）用非法的信

号量标识符。

38. ENAVAIL 非可用值

opensem (S)，waitsen (S) 或sigsem (S) 试图使用不是由creatsem (S) 初始化的信号量。sigsem使用信号量的顺序不对，即：出现在进程安排waitsem使用信号量之前。nbwaitsem使用信号量来保护正在被另外进程用的资源。

39. EISNAM 有名文件

说明了一个有名文件如信号量，共享数据，等等，而它本身又不是。

定义

进程标识符 (process ID)

系统中的每个活动进程都用一个叫进程标识符ID的正整数标识其唯一性。这个ID的范围是从0到30000。

父进程标识符 (Parent Process ID)

一个新进程是由当前的活动进程创建的，见fork (S)。一个进程的父进程标识符就是创建者的进程标识符。

进程组标识符 (Process Group ID)

每个活动进程都是进程组的一员，进程组也用正整数作标识，这个整数就叫进程组标识符。这个标识符是组头的进程标识符。这种分组允许相关进程之间传输信号。

Tty组标识符 (Tty Group ID)

每个活动进程都可以是一个终端组的一员，终端组也用称之为tty组标识符的正整数标识，这种分组用于依据组中某一进程的终止情况停止一组有关进程。参见exit (S) 和signal (S)。

实际用户标识符和实际组标识符 (Real User ID and Real Group ID)

每个用户都允许在系统中用称之为实际用户标识符的正整数标识。

每个用户也是一个组中的一员，组是用称之为实际组标识符的正整数标识的。

每个活动进程有一个实际用户标识和实际组标识，分别表示用户对进程创建的责任。

有效用户标识符和有效组标识符 (Effective User ID and Effective Group ID)

一个活动进程有一个有效用户标识符和一个有效组标识符，它们用于决定文件的存取权限（见下）。有效用户标识符和有效组标识符分别等于进程的实际用户标识符和实际组标识符，非进程或它的某个组先改变了set-user-ID(调整用户标识符)位或set-group-ID（调整组标识符）位，见exec (S)。

超级用户 (Super-User)

一个进程被认作一个super-user（超级用户）进程并确保有特殊的特权是因为它的有效用户标识符是0。

特殊进程 (Special Processes)

进程标识符为0和1的进程是特殊进程，分别叫proc0和proc1。

proc0是调度进程，proc1是初始化进程 (init)。proc1是系统中所有其他进程的祖先，用于控制进程的结构。

文件名 (File Name)

由最多到14个字符组成的名字可以用来给普通文件、特殊文件或目录命名。

这些字符可以是从除0（null）和ASCII码/（斜线）之外所有字符中选取。

注意：一般不要用*，？，［，或］作文件名的一部分，因为这些字符被shell赋予特殊意义了。同样，字符的高位不要设置。

路径名和路径前缀（Path Name and Path Prefix）。

一个路径名是这样一个null结尾的字符串：以可选择的斜线/开头，后跟用斜线/分开的零个或多个目录名，最后可能跟有文件名。一个文件名是除ASCII斜线/，空null之外的1到14个字符的串，一个目录名也是1到14个字符的串表示的有名目录（字符除ASCII斜线/和空null之外）。

如果路径以斜线/开头，则路径从根(root)开始查找。否则，从当前工作目录开始查找。

只有一个斜线/表示根目录。

除非特别声明，否则空路径当作不存在的文件处理。

目录（Directory）

目录项叫做联接，习惯上，一个目录至少有两个联接，·和··，分别读作点和点点。点（·）表示目录本身，点点（··）表示其父目录。

根目录和当前工作目录（Root Directory and Current Working Directory）

每个进程都有与之相连的根目录和当前工作目录，以便路径名查找。进程的根目录不需要是根文件系统的根目录。见chroot（C）和chroot（S）。

文件存取权限（File Access Permissions）

若如下条件中一条或多条成立，则文件的读、写、执行/查找权限对该进程确保：

进程有效用户标识符是超级用户。

进程有效用户标识符与文件主的用户标识符相匹配，而且文件方式的所有者部分（0700）的相应存取位被设置。

进程的有效用户标识符不匹配文件主的用户标识符，但进程的组标识符与文件的组匹配，而且文件方式的组部分（070）的相应存取位被设置。

进程的有效用户标识符不匹配文件主的用户标识符，进程有效组标识符也不匹配文件的组标识符，而且文件方式的其他部分（07）的相应存取位被设置。

除此之外，相应的权限将被否定。见chmod（C）和chmod（S）。

参见

intro（C）

A64L（S）

名字

a64l，l64a——长整型和基为64的ASCII之间转换。

句法

```
long a64l (S)
char *S;
char *l64a (l)
long l;
```

说明

这些子程序用于保持数按基为64的ASCII方式存放。这是一种把长整型表示成6个字符以内的字符串的记法，串中每个字母表示一位基制是64的“数字”。

表示“数字”的字符是：·表示0，/表示1，0到9表示2到11，A到Z表示12到37，a到z表示38到63。

a64l带入指向以空（null）结尾的基为64的表示形式的指针，返回一个相应的长整型（long）值。l64a带入一个长整型（long）变量，返回一个指向相应基为64表示形式的指针。

注意

l64a返回值是指向静态缓冲的指针，其内容在每次调用时都被重写。

ABORT (S)

名字

abort——生成一个IOT错误。

句法

```
abort ( )
```

说明

abort导致一个I/O捕俘信号（SIGIOT）送到调用程序。通常在core dump时导致进程终止。

如果调用进程被置为捕俘或忽略信号SIGIOT，则abort返回控制权。见signal(S)。

参见

adb (CP)，exit (S)，signal (S)。

诊断

如果失败的进程返回控制权给shell（sh(C)），则shell通常显示“abort——core dump”。

ABS (S)

名字

abs——返回一个整型值的绝对值。

句法

```
int abs (i)
int i;
```

说明

abs返回其整型操作数的绝对值。

参见

fabs (floor(S)中)。

注意

如果给定了硬件支持的最大负整数，则本函数返回值不变。

ACCESS (S)

名字

access——确定文件的存取特性。

句法

```
int access (path, amode)
char *path;
int amode;
```

说明

path指向一个命名了文件的路径，access用实际用户标识符替换有效用户标识符，用实际组标识符替换有效组标识符，再根据amode中包含的位模式（bit pattern）检查指名文件的存取特性。位模式可以通过选用下面的任意组合而形成：

04　读
02　写
01　执行（查找）
00　检查文件的存在性

如果下列条件之一成立，则不能存取文件：

路径前缀的一部分不是目录（ENOTDIR）。
对空路径名要求读、写或执行（查找）权限（ENOENT）。
指定的文件不存在（ENOENT）。
对路径前缀部分否定查找权限［EACCESS］。
对只读文件系统要求写操作［EROFS］。
对正在执行的纯过程（共享正文）文件要求写操作［ETXTBSY］。
文件的权限位不允许有存取要求（EACCESS）。
path指到进程的定位地址空间之外（EFAULT）。

access通过检查“所有者”写，读和执行方式位，检查文件主的权限。对文件的同组成员，检查“组”方式位。对所有其他人，检查“其他”方式位。

返回值

如果要求的存取得到允许，则返回值是0。否则，返回－1或表示错误的errno。

参见

chmod (S)，stat (S)

注意

超级用户（root）可存取任何文件，不论权限如何设置。

ACCT (S)

名字

acct——开始或停止进程计帐。

句法

```
int acct (path)
char *path;
```

说明

acct用于控制系统的进程计帐子程序开始或停止。如果子程序开始计帐，则帐目记录在每个进程终止时都要写到帐目文件中去。一个进程可以调用exit终止，也可以接受不忽视的信号或捕俘来终止，见exit (S) 和signal (S) 。调用进程的有效用户标识符必须是超级用户。

path指向帐目文件的路径名，帐目文件的格式在acct (F) 中给出。

如果path非零，而且在系统调用过程中无错，则计帐子程序开始起作用。如果path是零，而且系统调用无错，则计帐子程序不起作用。

如果下列条件之一成立，则acct失败：

调用进程的有效用户标识符不是超级用户。 [EPERM]

如果计帐已经开始又试图再执行开始。 [EBUSY]

路径前缀不是目录。 [ENOTDIR]

帐目文件的路径名不存在。 [ENOENT]

部分路径前缀否定查找权限。 [EACCESS]

path指的文件不是一个普通文件。 [EACCESS]

对帐目文件设置权限被否定。 [EACCESS]

指定的文件是目录。 [EACCESS]

指定的文件在只读文件系统中。 [EROFS]

path指向一个非法地址 [EFAULT]

返回值

直到顺利完成，才返回0，否则返回－1或返回表示错误的errno。

参见

accton (C) , acctcom (C) , acct (F) 。

ALARM (S)

名字

alarm——设置进程的报警时钟。

句法

```
unsigned alarm (sec)
unsigned sec;
```

说明

alarm设置调用进程的报警时钟为sec秒。sec秒“实际时间”之后。报警钟送一个SIGALRM信号给进程，见signal (S) 。

尽管alarm在设置报警时钟之后不等待信号，但仍可用 pause (S) 使调用进程等待。

报警要求不能嵌套，成功的调用将重设调用进程的报警时钟。

如果sec是0，则以前的报警要求被消除。

返回值

alarm返回以前保留在调用进程的报警时钟的时间总数。

参见

pause (S) , signal (S)

ASSERT (S)

名字

assert——帮助验证程序的正确性。

句法

```
#include <assert.h>
assert (expression) ;
```

说明

这个宏对开发时把诊断放入程序是很有用的。执行时如果表达式expression为假，则打印：

Assertion failed: file name, line nnn

name是源文件名，nnn是assert语句在程序中的行号。

注意

要想挂起对 assert 的调用，请在编译程序时用任选项“-DNDEBUG”，见cc (CP) 。

ATOF (S)

名字

atof, atoi, atol——把ASCII转换为数字。

句法

```
double atof(nptr)
char *nptr;
int atoi(nptr)
char *nptr;
long atol(nptr)
char *nptr;
```

说明

这些函数将一个由nptr指出的字符串分别转换为浮点数、整数和长整数表示。第一个不可识别的字符结束字符串。

atof识别这样形式的串：

[+|-] digits [.digits] [e|E [+|-] digits]

这里digits是一串十进制数字。串之前可以放任意多的制表符和空格符。+和-可有可无。e或E用于标记指数的开始。

atoi和atol识别这样形式的串：

[+|-] digits