UNIX标准及实现

1. UNIX标准化
2. ISO C

ISO C标准的意图是提供C程序的可移植性，使其能适合于大量不同的操作系统，而不只是适合UNIX系统。此标准不仅定义了C程序语言的语法和语义，还定义了其标准库。因为所有现今的UNIX系统都提供C标准中定义的库函数，所以该标准库非常重要。

ISO C标准定义的头文件(24项)：

|  |
| --- |
| <assert.h>       ----------------------   验证程序断言  <complex.h>    ----------------------   支持复数算术运算  <ctype.h>        ----------------------   字符类型  <errno.h>        ----------------------   出错码  <fenv.h>          ----------------------   浮点环境  <float.h>          ----------------------   浮点常量  <inttypes.h>    ----------------------   整型格式转换  <iso646.h>      ----------------------   替代关系操作符宏  <limits.h>         ----------------------   实现常量  <locale.h>       ----------------------   局部类别  <math.h>         ----------------------   数学常量  <setjmp.h>      ----------------------   非局部goto  <signal.h>       ----------------------   信号  <stdarg.h>      ----------------------   可变参数表  <stdbool.h>     ----------------------   布尔类型和值  <stddef.h>       ----------------------   标准定义  <stdint.h>        ----------------------   整型  <stdio.h>         ----------------------   标准I/O库  <stdlib.h>        ----------------------   实用程序库函数  <string.h>       ----------------------   字符串操作  <tgmath.h>     ----------------------   通用类型数学宏  <time.h>         ----------------------   时间和日期  <wchar.h>      ----------------------   扩展的多字节和宽字符支持  <wctype.h>     ----------------------   宽字符分类和映射支持 |

1. IEEE POSIX

POSIX指的是可移植操作系统接口（Portable Operating System Interface）。

POSIX标准定义的必须的头文件(26项)：

|  |
| --- |
| <dirent.h>      ----------------------   目录项  <fcntl.h>         ----------------------   文件控制  <fnmatch.h>   ----------------------   文件名匹配类型  <glob.h>         ----------------------   路径名模式匹配类型  <grp.h>          ----------------------   组文件  <netdb.h>      ----------------------   网络数据库操作  <pwd.h>         ----------------------   口令文件  <regex.h>      ----------------------   正则表达式  <tar.h>           ----------------------   tar归档值  <termios.h>    ----------------------   终端I/O  <unistd.h>      ----------------------   符号常量  <utime.h>       ----------------------   文件时间  <wordexp.h>   ----------------------   字扩展类型  <arpa/inet.h>  ----------------------   Internet定义  <net/if..h>       ----------------------   套接字本地接口  <netinet/in.h> ----------------------   Internet地址族  <netinet/tcp.h>----------------------   传输控制协议定义  <sys/mman.h>----------------------   内存管理声明  <sys/select.h>----------------------   select函数  <sys/socket.h>----------------------   套接字接口  <sys/stat.h>    ----------------------   文件状态  <sys/times.h>  ----------------------   进程时间  <sys/types.h>  ----------------------   基本系统数据类型  <sys/un.h>      ----------------------   UNIX域套接字定义  <sys/utsname.h>----------------------系统名  <sys/wait.h>    ----------------------   进程控制 |

POSIX标准定义的XSI扩展头文件(26项)：

|  |
| --- |
| <cpio.h>          ----------------------   cpio归档值  <dlfcn.h>         ----------------------   动态链接  <fmtmsg.h>     ----------------------   消息显示结构  <ftw.h>            ----------------------   文件树漫游  <iconv.h>        ----------------------   代码集转换实用程序  <langinfo.h>    ----------------------   语言信息常量  <libgen.h>       ----------------------   模式匹配函数定义  <monetary.h>  ----------------------   货币类型  <ndbm.h>        ----------------------   数据库操作  <nl\_types.h>   ----------------------   消息类别  <poll.h>           ----------------------   轮询函数  <search.h>      ----------------------   搜索表  <strings.h>      ----------------------   字符串操作  <syslog.h>      ----------------------   系统出错日志记录  <ucontext.h>   ----------------------   用户上下文  <ulimit.h>        ----------------------   用户限制  <utmpx.h>       ----------------------   用户帐户数据库  <sys/ipc.h>      ----------------------   IPC  <sys/msg.h>    ----------------------   消息队列  <sys/resource.h>-------------------   资源操作  <sys/sem.h>    ----------------------   信号量  <sys/shm.h>    ----------------------   共享存储  <sys/statvfs.h>----------------------   文件系统信息  <sys/time.h>    ----------------------   时间类型  <sys/timeb.h>  ----------------------   附加的日期和时间定义  <sys/uio.h>      ----------------------   矢量I/O操作 |

POSIX标准定义的可选头文件(8项)：

|  |
| --- |
| <aio.h>            ----------------------   异步I/O  <mqueue.h>    ----------------------   消息队列  <pthread.h>    ----------------------   线程  <sched.h>       ----------------------   执行调度  <semaphore.h>---------------------   信号量  <spawn.h>       ----------------------   实时spawn接口  <stropts.h>      ----------------------   XSI STREAMS接口  <trace.h>         ----------------------   时间跟踪 |

1. Single UNIX Specification

Single UNIX Specification（SUS，单一UNIX规范）是POSIX.1标准的一个超集，它定义了一些附加接口拓展了POSIX.1规范提供的功能。POSIX.1相当于Single UNIX Specification中的基本规范部分。

1. FIPS

FIPS代表的是联邦信息处理标准（Federal Information Processing Standard），这一标准是由美国政府发布的，并有美国政府用于计算机系统的采购。他要求任何希望向美国政府销售符合POSIX.1标准的计算机系统的厂商都应该支持POSIX.1的某些选项。（目前已经撤回，我们不再考虑它）

1. UNIX系统的实现
2. SVR4

SVR4（UNIX System V Release 4）是AT&T的UNIX系统实验室的产品。

1. 4.4BSD

BSD（Berkeley Software Distribution）是由加州大学伯克利分校的计算机系统研究组（CSRG）研究开发和发行的。

1. FreeBSD

FreeBSD基于4.4BSD-Lite操作系统。在加州大学伯克利分校的CSRG决定终止其在UNIX操作系统的BSD版本的研发工作，而且386BSD项目被忽视很长时间之后，为了继续坚持BSD系列，形成了FreeBSD项目。

1. Linux

Linux是一种提供类似于UNIX的丰富编程环境的操作系统，在GNU公用许可证的指导下，Linux是免费使用的。Linux的普及是计算机产业中一道亮丽风景线。Linux经常是支持较新硬件的第一个操作系统，这一点使其引人注目。

1. Mac OS X

与其以前的版本相比，Mac OS X使用了完全不同的技术。其核心操作系统称为“Darwin”，它基于Mach内核、FreeBSD操作系统以及具有面向对象框架的驱动和其他内核扩展的结合。Mac OS X 10.5的Intel部分已经被验证为是一个UNIX系统。

1. Solaris

Solaris是由Sun Microsystems（现为Orcale）开发的UNIX系统版本。它基于SVR4，在超过15年的时间里，Sun Microsystems的工程师对其功能不断增强。它是唯一在商业上取得成功的SVR4后裔，并被正式验证为UNIX系统。

1. 其他UNIX系统

已经通过验证的其他UNIX版本包括：

* AIX，IBM版的UNIX系统
* HP-UX，HP版的UNIX系统
* IRIX，Silicon Graphics版的UNIX系统
* UnixWare，SVR4派生的UNIX系统，现在有SCO销售

1. 标准和实现的关系

前面提到的各个标准定义了任一实际系统的子集。本书主要关注4种实际的UNIX系统：FreeBSD 8.0，Linux 3.2.0，Mac OS X 10.6.8和Solaris 10。再者四种系统中，虽然只有Mac OS X和 Solaris 10能够称自己是一种UNIX系统，但是所有这4种系统都提供UNIX编程环境。

1. 限制

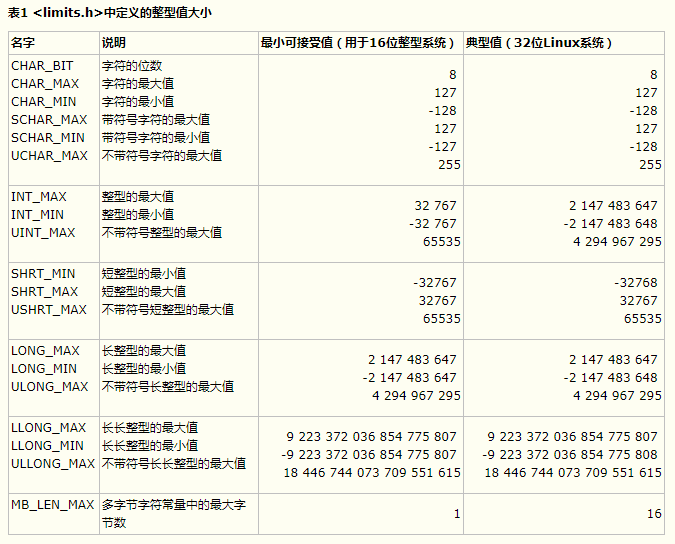
以下两种类型的限制是必需的。

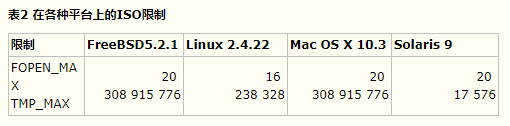
1. 编译时限制（例如，短整型的最大值是多少？）
2. 运行时限制（例如，文件名有多少个字符？）

为了解决这类问题，提供以下三种限制。

1. 编译时限制（头文件）
2. 与文件或者目录无关的运行时限制（sysconf函数）
3. 与文件或者目录有关的运行时限制（pathconf和fpathconf函数）
4. ISO C限制

ISO C定义的限制都是编译时限制。文件<limits.h>中定义的C标准限制如下：



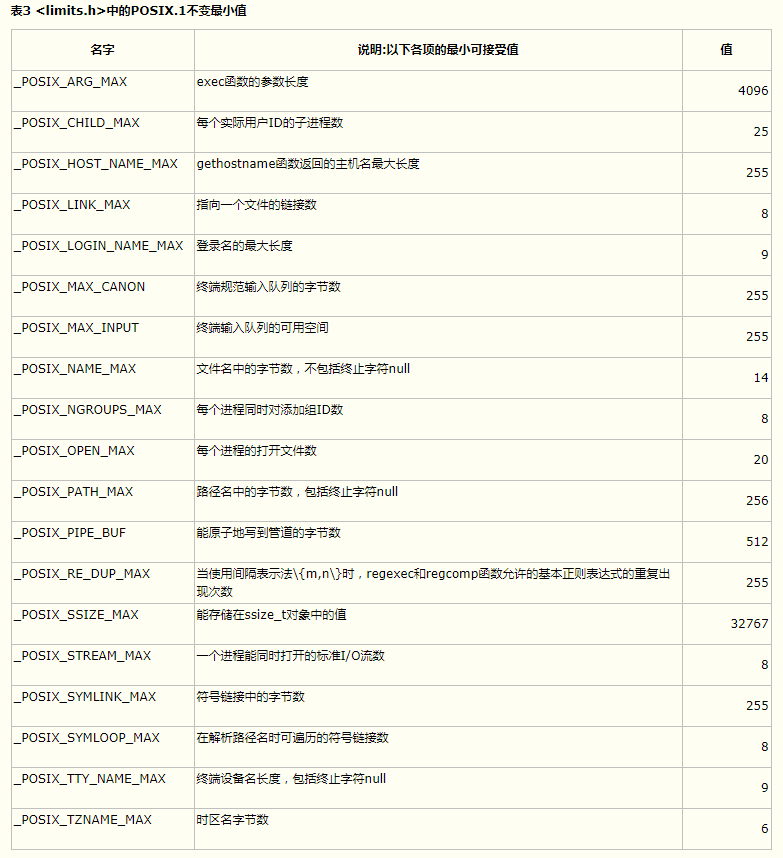


ISO C还定义了常量FILENAME\_MAX，因为某些操作系统实现在历史上将它定义得太小，以至于不能满足应用的需求，所以我们应避免使用该常量。

1. POSIX限制

POSIX.1定义了很多涉及操作系统实现限制的常量，我们只关心与基本POSIX.1接口有关的部分。这些限制和常量被分成下列5类。

1. 不变的最小值：见表3。
2. 不变值：SSIZE\_MAX。
3. 运行时可以增加的值：CHARCLASS\_NAME\_MAX、COLL\_WEIGHTS\_MAX、LINE\_MAX、NGROUPS\_MAX以及RE\_DUP\_MAX。
4. 运行时不变的值（可能不确定）：ARG\_MAX、CHILD\_MAX、HOST\_NAME\_MAX、LOGIN\_NAME\_MAX、OPEN\_MAX、PAGESIZE、RE\_DUP\_MAX、STREAM\_MAXS、SYMLOOP\_MAX、TTY\_NAME\_MAX以及TZNAME\_MAX。
5. 路径名可变值（可能不确定）：FILESIZEBITS、LINK\_MAX、MAX\_CANON、MAX\_INPUT、NAME\_MAX、PATH\_MAX、PIPE\_BUF以及SYMLINK\_MAX。



上述5类共44个限制和常量中，有一些可定义在<limits.h>中，其余的则按照具体条件可定义或不定义。

1. XSI限制

XSI还定义了处理实现限制的下面几个常量：

1. 不变最小值：表4中列出的10个常量。
2. 数值限制：LONG\_BIT和WORE\_BIT。
3. 运行时不变值（可能不确定）：ATEXIT\_MAX、IOV\_MAX以及PAGE\_SIZE。

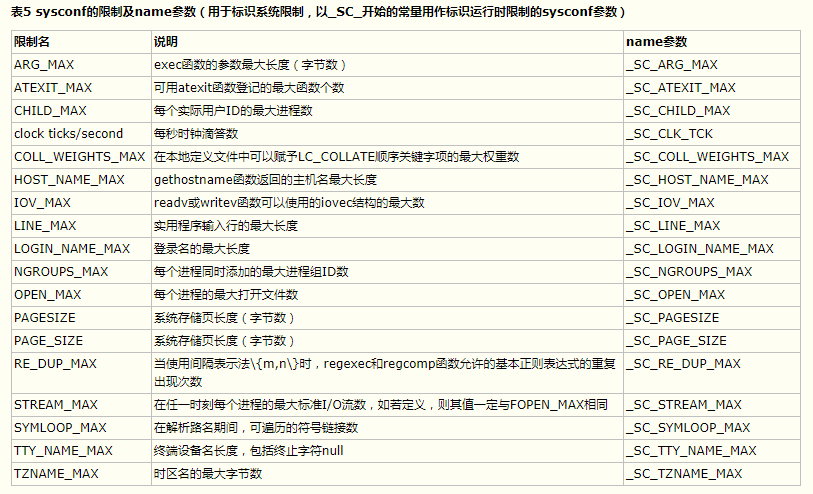


1. 函数sysconf、pathconf和fpathconf

我们已列出了实现必须支持的各种最小值，但是怎样才能找到一个特定系统实际支持的限制值呢？正如前面提到的，某些限制值在编译时是可用的，而另外一些则必须在运行时确定。我们也曾提及在一个给定的系统中某些限制值是不会更改的，而其他限制值则与文件和目录相关联，是可以改变的。运行时限制可通过调用下面三个函数中的一个而取得：

|  |
| --- |
| #include *<unistd.h>*  long sysconf( int name );  long pathconf( **const** char \*pathname, int name );  long fpathconf( int filedes, int name );  所有函数返回值：若成功则返回相应值；若出错则返回-1。 |

后两个函数之间的差别是一个用路径名作为其参数，另一个则取文件描述符作为参数。



实例：构建C程序以打印所有得到支持的系统配置限制prog2-1.awk

|  |
| --- |
| BEGIN{  printf("#include **\"**apue.h**\"\n**")  printf("#include <errno.h>**\n**")  printf("#include <limits.h>**\n**")  printf("**\n**")  printf("static void pr\_sysconf(char \*, int);**\n**")  printf("static void pr\_pathconf(char \*, char \*, int);**\n**")  printf("**\n**")  printf("int**\n**")  printf("main(int argc, char \*argv[])**\n**")  printf("{**\n**")  printf("**\t**if(argc != 2)**\n**")  printf("**\t\t**err\_quit(**\"**usage: a.out <dirname>**\"**);**\n\n**")  FS="**\t**+"  **while**(getline <"sysconf.sym" > 0)  {  printf("#ifdef %s**\n**", $1)  printf("**\t**printf(**\"**%s defined to be %%d**\\**n**\"**, %s+0);**\n**", $1, $1)  printf("#else**\n**")  printf("**\t**printf(**\"**no symbol for %s**\\**n**\"**);**\n**", $1)  printf("#endif**\n**")  printf("#ifdef %s**\n**", $2)  printf("**\t**pr\_sysconf(**\"**%s =**\"**, %s);**\n**", $1, $2)  printf("#else**\n**")  printf("**\t**printf(**\"**no symbol for %s**\\**n**\"**);**\n**", $2)  printf("#endif**\n**")  }  close("sysconf.sym")  **while**(getline <"pathconf.sym" > 0)  {  printf("#ifdef %s**\n**", $1)  printf("**\t**printf(**\"**%s defined to be %%d**\\**n**\"**, %s+0);**\n**", $1, $1)  printf("#else**\n**")  printf("**\t**printf(**\"**no symbol for %s**\\**n**\"**);**\n**", $1)  printf("#endif**\n**")  printf("#ifdef %s**\n**", $2)  printf("**\t**pr\_pathconf(**\"**%s =**\"**, argv[1], %s);**\n**", $1, $2)  printf("#else**\n**")  printf("**\t**printf(**\"**no symbol for %s**\\**n**\"**);**\n**", $2)  printf("#endif**\n**")  }  close("pathconf.sym")  exit  }  END{  printf("**\t**exit(0);**\n**")  printf("}**\n\n**")  printf("static void**\n**")  printf("pr\_sysconf(char \*mesg, int name)**\n**")  printf("{**\n**")  printf("**\t**long val;**\n\n**")  printf("**\t**fputs(mesg, stdout);**\n**")  printf("**\t**errno = 0;**\n**")  printf("**\t**if((val = sysconf(name)) < 0) {**\n**")  printf("**\t\t**if(errno != 0) {**\n**")  printf("**\t\t\t**if(errno == EINVAL)**\n**")  printf("**\t\t\t\t**fputs(**\"** (not supported)**\\**n**\"**, stdout);**\n**")  printf("**\t\t\t**else**\n**")  printf("**\t\t\t\t**err\_sys(**\"**sysconf error**\"**);**\n**")  printf("**\t\t**} else {**\n**")  printf("**\t\t\t**fputs(**\"**(no limit)**\\**n**\"**, stdout);**\n**")  printf("**\t\t**}**\n**")  printf("**\t**}else {**\n**")  printf("**\t\t**printf(**\"** %%ld**\\**n**\"**, val);**\n**")  printf("**\t**}**\n**")  printf("}**\n\n**")  printf("static void**\n**")  printf("pr\_pathconf(char \*mesg, char \*path, int name)**\n**")  printf("{**\n**")  printf("**\t**long val;**\n**")  printf("**\n**")  printf("**\t**fputs(mesg, stdout);**\n**")  printf("**\t**errno = 0;**\n**")  printf("**\t**if((val = pathconf(path, name)) < 0) {**\n**")  printf("**\t\t**if(errno != 0){**\n**")  printf("**\t\t\t**if(errno == EINVAL)**\n**")  printf("**\t\t\t\t**fputs(**\"** (not supported)**\\**n**\"**, stdout);**\n**")  printf("**\t\t\t**else**\n**")  printf("**\t\t\t\t**err\_sys(**\"**pathconf error, path = %%s**\"**, path);**\n**")  printf("**\t\t**} else {**\n**")  printf("**\t\t\t**fputs(**\"** (no limits)**\\**n**\"**, stdout);**\n**")  printf("**\t\t**}**\n**")  printf("**\t**} else {**\n**")  printf("**\t\t**printf(**\"** %%ld**\\**n**\"**, val);**\n**")  printf("**\t**}**\n**")  printf("}**\n**")  } |

将pathconf.sym和sysconf.sym两个文件拷入当前目录下，执行

awk -f prog2-1.awk sysconf.sym pathconf.sym >prog2-2.c

生成prog2-2.c如下：

|  |
| --- |
| #include *"apue.h"*  #include *<errno.h>*  #include *<limits.h>*  **static** void pr\_sysconf(char \*, int);  **static** void pr\_pathconf(char \*, char \*, int);  int  main(int argc, char \*argv[])  {  **if**(argc != 2)  err\_quit("usage: a.out <dirname>");  #ifdef ARG\_MAX  printf("ARG\_MAX defined to be %d**\n**", ARG\_MAX+0);  #else  printf("no symbol for ARG\_MAX**\n**");  #endif  #ifdef \_SC\_ARG\_MAX  pr\_sysconf("ARG\_MAX =", \_SC\_ARG\_MAX);  #else  printf("no symbol for \_SC\_ARG\_MAX**\n**");  #endif  #ifdef ATEXIT\_MAX  printf("ATEXIT\_MAX defined to be %d**\n**", ATEXIT\_MAX+0);  #else  printf("no symbol for ATEXIT\_MAX**\n**");  #endif  #ifdef \_SC\_ATEXIT\_MAX  pr\_sysconf("ATEXIT\_MAX =", \_SC\_ATEXIT\_MAX);  #else  printf("no symbol for \_SC\_ATEXIT\_MAX**\n**");  #endif  #ifdef CHARCLASS\_NAME\_MAX  printf("CHARCLASS\_NAME\_MAX defined to be %d**\n**", CHARCLASS\_NAME\_MAX+0);  #else  printf("no symbol for CHARCLASS\_NAME\_MAX**\n**");  #endif  #ifdef \_SC\_CHARCLASS\_NAME\_MAX  pr\_sysconf("CHARCLASS\_NAME\_MAX =", \_SC\_CHARCLASS\_NAME\_MAX);  #else  printf("no symbol for \_SC\_CHARCLASS\_NAME\_MAX**\n**");  #endif  #ifdef CHILD\_MAX  printf("CHILD\_MAX defined to be %d**\n**", CHILD\_MAX+0);  #else  printf("no symbol for CHILD\_MAX**\n**");  #endif  #ifdef \_SC\_CHILD\_MAX  pr\_sysconf("CHILD\_MAX =", \_SC\_CHILD\_MAX);  #else  printf("no symbol for \_SC\_CHILD\_MAX**\n**");  #endif  #ifdef CLOCKTICKSPERSECOND */\*clock ticks/second\*/*  printf("CLOCKTICKSPERSECOND /\*clock ticks/second\*/ defined to be %d**\n**", CLOCKTICKSPERSECOND */\*clock ticks/second\*/*+0);  #else  printf("no symbol for CLOCKTICKSPERSECOND /\*clock ticks/second\*/**\n**");  #endif  #ifdef \_SC\_CLK\_TCK  pr\_sysconf("CLOCKTICKSPERSECOND /\*clock ticks/second\*/ =", \_SC\_CLK\_TCK);  #else  printf("no symbol for \_SC\_CLK\_TCK**\n**");  #endif  #ifdef COLL\_WEIGHTS\_MAX  printf("COLL\_WEIGHTS\_MAX defined to be %d**\n**", COLL\_WEIGHTS\_MAX+0);  #else  printf("no symbol for COLL\_WEIGHTS\_MAX**\n**");  #endif  #ifdef \_SC\_COLL\_WEIGHTS\_MAX  pr\_sysconf("COLL\_WEIGHTS\_MAX =", \_SC\_COLL\_WEIGHTS\_MAX);  #else  printf("no symbol for \_SC\_COLL\_WEIGHTS\_MAX**\n**");  #endif  #ifdef DELAYTIMER\_MAX  printf("DELAYTIMER\_MAX defined to be %d**\n**", DELAYTIMER\_MAX+0);  #else  printf("no symbol for DELAYTIMER\_MAX**\n**");  #endif  #ifdef \_SC\_DELAYTIMER\_MAX  pr\_sysconf("DELAYTIMER\_MAX =", \_SC\_DELAYTIMER\_MAX);  #else  printf("no symbol for \_SC\_DELAYTIMER\_MAX**\n**");  #endif  #ifdef HOST\_NAME\_MAX  printf("HOST\_NAME\_MAX defined to be %d**\n**", HOST\_NAME\_MAX+0);  #else  printf("no symbol for HOST\_NAME\_MAX**\n**");  #endif  #ifdef \_SC\_HOST\_NAME\_MAX  pr\_sysconf("HOST\_NAME\_MAX =", \_SC\_HOST\_NAME\_MAX);  #else  printf("no symbol for \_SC\_HOST\_NAME\_MAX**\n**");  #endif  #ifdef IOV\_MAX  printf("IOV\_MAX defined to be %d**\n**", IOV\_MAX+0);  #else  printf("no symbol for IOV\_MAX**\n**");  #endif  #ifdef \_SC\_IOV\_MAX  pr\_sysconf("IOV\_MAX =", \_SC\_IOV\_MAX);  #else  printf("no symbol for \_SC\_IOV\_MAX**\n**");  #endif  #ifdef LINE\_MAX  printf("LINE\_MAX defined to be %d**\n**", LINE\_MAX+0);  #else  printf("no symbol for LINE\_MAX**\n**");  #endif  #ifdef \_SC\_LINE\_MAX  pr\_sysconf("LINE\_MAX =", \_SC\_LINE\_MAX);  #else  printf("no symbol for \_SC\_LINE\_MAX**\n**");  #endif  #ifdef LOGIN\_NAME\_MAX  printf("LOGIN\_NAME\_MAX defined to be %d**\n**", LOGIN\_NAME\_MAX+0);  #else  printf("no symbol for LOGIN\_NAME\_MAX**\n**");  #endif  #ifdef \_SC\_LOGIN\_NAME\_MAX  pr\_sysconf("LOGIN\_NAME\_MAX =", \_SC\_LOGIN\_NAME\_MAX);  #else  printf("no symbol for \_SC\_LOGIN\_NAME\_MAX**\n**");  #endif  #ifdef NGROUPS\_MAX  printf("NGROUPS\_MAX defined to be %d**\n**", NGROUPS\_MAX+0);  #else  printf("no symbol for NGROUPS\_MAX**\n**");  #endif  #ifdef \_SC\_NGROUPS\_MAX  pr\_sysconf("NGROUPS\_MAX =", \_SC\_NGROUPS\_MAX);  #else  printf("no symbol for \_SC\_NGROUPS\_MAX**\n**");  #endif  #ifdef OPEN\_MAX  printf("OPEN\_MAX defined to be %d**\n**", OPEN\_MAX+0);  #else  printf("no symbol for OPEN\_MAX**\n**");  #endif  #ifdef \_SC\_OPEN\_MAX  pr\_sysconf("OPEN\_MAX =", \_SC\_OPEN\_MAX);  #else  printf("no symbol for \_SC\_OPEN\_MAX**\n**");  #endif  #ifdef PAGESIZE  printf("PAGESIZE defined to be %d**\n**", PAGESIZE+0);  #else  printf("no symbol for PAGESIZE**\n**");  #endif  #ifdef \_SC\_PAGESIZE  pr\_sysconf("PAGESIZE =", \_SC\_PAGESIZE);  #else  printf("no symbol for \_SC\_PAGESIZE**\n**");  #endif  #ifdef PAGE\_SIZE  printf("PAGE\_SIZE defined to be %d**\n**", PAGE\_SIZE+0);  #else  printf("no symbol for PAGE\_SIZE**\n**");  #endif  #ifdef \_SC\_PAGE\_SIZE  pr\_sysconf("PAGE\_SIZE =", \_SC\_PAGE\_SIZE);  #else  printf("no symbol for \_SC\_PAGE\_SIZE**\n**");  #endif  #ifdef RE\_DUP\_MAX  printf("RE\_DUP\_MAX defined to be %d**\n**", RE\_DUP\_MAX+0);  #else  printf("no symbol for RE\_DUP\_MAX**\n**");  #endif  #ifdef \_SC\_RE\_DUP\_MAX  pr\_sysconf("RE\_DUP\_MAX =", \_SC\_RE\_DUP\_MAX);  #else  printf("no symbol for \_SC\_RE\_DUP\_MAX**\n**");  #endif  #ifdef RTSIG\_MAX  printf("RTSIG\_MAX defined to be %d**\n**", RTSIG\_MAX+0);  #else  printf("no symbol for RTSIG\_MAX**\n**");  #endif  #ifdef \_SC\_RTSIG\_MAX  pr\_sysconf("RTSIG\_MAX =", \_SC\_RTSIG\_MAX);  #else  printf("no symbol for \_SC\_RTSIG\_MAX**\n**");  #endif  #ifdef SEM\_NSEMS\_MAX  printf("SEM\_NSEMS\_MAX defined to be %d**\n**", SEM\_NSEMS\_MAX+0);  #else  printf("no symbol for SEM\_NSEMS\_MAX**\n**");  #endif  #ifdef \_SC\_SEM\_NSEMS\_MAX  pr\_sysconf("SEM\_NSEMS\_MAX =", \_SC\_SEM\_NSEMS\_MAX);  #else  printf("no symbol for \_SC\_SEM\_NSEMS\_MAX**\n**");  #endif  #ifdef SEM\_VALUE\_MAX  printf("SEM\_VALUE\_MAX defined to be %d**\n**", SEM\_VALUE\_MAX+0);  #else  printf("no symbol for SEM\_VALUE\_MAX**\n**");  #endif  #ifdef \_SC\_SEM\_VALUE\_MAX  pr\_sysconf("SEM\_VALUE\_MAX =", \_SC\_SEM\_VALUE\_MAX);  #else  printf("no symbol for \_SC\_SEM\_VALUE\_MAX**\n**");  #endif  #ifdef SIGQUEUE\_MAX  printf("SIGQUEUE\_MAX defined to be %d**\n**", SIGQUEUE\_MAX+0);  #else  printf("no symbol for SIGQUEUE\_MAX**\n**");  #endif  #ifdef \_SC\_SIGQUEUE\_MAX  pr\_sysconf("SIGQUEUE\_MAX =", \_SC\_SIGQUEUE\_MAX);  #else  printf("no symbol for \_SC\_SIGQUEUE\_MAX**\n**");  #endif  #ifdef STREAM\_MAX  printf("STREAM\_MAX defined to be %d**\n**", STREAM\_MAX+0);  #else  printf("no symbol for STREAM\_MAX**\n**");  #endif  #ifdef \_SC\_STREAM\_MAX  pr\_sysconf("STREAM\_MAX =", \_SC\_STREAM\_MAX);  #else  printf("no symbol for \_SC\_STREAM\_MAX**\n**");  #endif  #ifdef SYMLOOP\_MAX  printf("SYMLOOP\_MAX defined to be %d**\n**", SYMLOOP\_MAX+0);  #else  printf("no symbol for SYMLOOP\_MAX**\n**");  #endif  #ifdef \_SC\_SYMLOOP\_MAX  pr\_sysconf("SYMLOOP\_MAX =", \_SC\_SYMLOOP\_MAX);  #else  printf("no symbol for \_SC\_SYMLOOP\_MAX**\n**");  #endif  #ifdef TIMER\_MAX  printf("TIMER\_MAX defined to be %d**\n**", TIMER\_MAX+0);  #else  printf("no symbol for TIMER\_MAX**\n**");  #endif  #ifdef \_SC\_TIMER\_MAX  pr\_sysconf("TIMER\_MAX =", \_SC\_TIMER\_MAX);  #else  printf("no symbol for \_SC\_TIMER\_MAX**\n**");  #endif  #ifdef TTY\_NAME\_MAX  printf("TTY\_NAME\_MAX defined to be %d**\n**", TTY\_NAME\_MAX+0);  #else  printf("no symbol for TTY\_NAME\_MAX**\n**");  #endif  #ifdef \_SC\_TTY\_NAME\_MAX  pr\_sysconf("TTY\_NAME\_MAX =", \_SC\_TTY\_NAME\_MAX);  #else  printf("no symbol for \_SC\_TTY\_NAME\_MAX**\n**");  #endif  #ifdef TZNAME\_MAX  printf("TZNAME\_MAX defined to be %d**\n**", TZNAME\_MAX+0);  #else  printf("no symbol for TZNAME\_MAX**\n**");  #endif  #ifdef \_SC\_TZNAME\_MAX  pr\_sysconf("TZNAME\_MAX =", \_SC\_TZNAME\_MAX);  #else  printf("no symbol for \_SC\_TZNAME\_MAX**\n**");  #endif  #ifdef FILESIZEBITS  printf("FILESIZEBITS defined to be %d**\n**", FILESIZEBITS+0);  #else  printf("no symbol for FILESIZEBITS**\n**");  #endif  #ifdef \_PC\_FILESIZEBITS  pr\_pathconf("FILESIZEBITS =", argv[1], \_PC\_FILESIZEBITS);  #else  printf("no symbol for \_PC\_FILESIZEBITS**\n**");  #endif  #ifdef LINK\_MAX  printf("LINK\_MAX defined to be %d**\n**", LINK\_MAX+0);  #else  printf("no symbol for LINK\_MAX**\n**");  #endif  #ifdef \_PC\_LINK\_MAX  pr\_pathconf("LINK\_MAX =", argv[1], \_PC\_LINK\_MAX);  #else  printf("no symbol for \_PC\_LINK\_MAX**\n**");  #endif  #ifdef MAX\_CANON  printf("MAX\_CANON defined to be %d**\n**", MAX\_CANON+0);  #else  printf("no symbol for MAX\_CANON**\n**");  #endif  #ifdef \_PC\_MAX\_CANON  pr\_pathconf("MAX\_CANON =", argv[1], \_PC\_MAX\_CANON);  #else  printf("no symbol for \_PC\_MAX\_CANON**\n**");  #endif  #ifdef MAX\_INPUT  printf("MAX\_INPUT defined to be %d**\n**", MAX\_INPUT+0);  #else  printf("no symbol for MAX\_INPUT**\n**");  #endif  #ifdef \_PC\_MAX\_INPUT  pr\_pathconf("MAX\_INPUT =", argv[1], \_PC\_MAX\_INPUT);  #else  printf("no symbol for \_PC\_MAX\_INPUT**\n**");  #endif  #ifdef NAME\_MAX  printf("NAME\_MAX defined to be %d**\n**", NAME\_MAX+0);  #else  printf("no symbol for NAME\_MAX**\n**");  #endif  #ifdef \_PC\_NAME\_MAX  pr\_pathconf("NAME\_MAX =", argv[1], \_PC\_NAME\_MAX);  #else  printf("no symbol for \_PC\_NAME\_MAX**\n**");  #endif  #ifdef PATH\_MAX  printf("PATH\_MAX defined to be %d**\n**", PATH\_MAX+0);  #else  printf("no symbol for PATH\_MAX**\n**");  #endif  #ifdef \_PC\_PATH\_MAX  pr\_pathconf("PATH\_MAX =", argv[1], \_PC\_PATH\_MAX);  #else  printf("no symbol for \_PC\_PATH\_MAX**\n**");  #endif  #ifdef PIPE\_BUF  printf("PIPE\_BUF defined to be %d**\n**", PIPE\_BUF+0);  #else  printf("no symbol for PIPE\_BUF**\n**");  #endif  #ifdef \_PC\_PIPE\_BUF  pr\_pathconf("PIPE\_BUF =", argv[1], \_PC\_PIPE\_BUF);  #else  printf("no symbol for \_PC\_PIPE\_BUF**\n**");  #endif  #ifdef SYMLINK\_MAX  printf("SYMLINK\_MAX defined to be %d**\n**", SYMLINK\_MAX+0);  #else  printf("no symbol for SYMLINK\_MAX**\n**");  #endif  #ifdef \_PC\_SYMLINK\_MAX  pr\_pathconf("SYMLINK\_MAX =", argv[1], \_PC\_SYMLINK\_MAX);  #else  printf("no symbol for \_PC\_SYMLINK\_MAX**\n**");  #endif  #ifdef \_POSIX\_TIMESTAMP\_RESOLUTION  printf("\_POSIX\_TIMESTAMP\_RESOLUTION defined to be %d**\n**", \_POSIX\_TIMESTAMP\_RESOLUTION+0);  #else  printf("no symbol for \_POSIX\_TIMESTAMP\_RESOLUTION**\n**");  #endif  #ifdef \_PC\_TIMESTAMP\_RESOLUTION  pr\_pathconf("\_POSIX\_TIMESTAMP\_RESOLUTION =", argv[1], \_PC\_TIMESTAMP\_RESOLUTION);  #else  printf("no symbol for \_PC\_TIMESTAMP\_RESOLUTION**\n**");  #endif  exit(0);  }  **static** void  pr\_sysconf(char \*mesg, int name)  {  long val;  fputs(mesg, stdout);  errno = 0;  **if**((val = sysconf(name)) < 0) {  **if**(errno != 0) {  **if**(errno == EINVAL)  fputs(" (not supported)**\n**", stdout);  **else**  err\_sys("sysconf error");  } **else** {  fputs("(no limit)**\n**", stdout);  }  }**else** {  printf(" %ld**\n**", val);  }  }  **static** void  pr\_pathconf(char \*mesg, char \*path, int name)  {  long val;  fputs(mesg, stdout);  errno = 0;  **if**((val = pathconf(path, name)) < 0) {  **if**(errno != 0){  **if**(errno == EINVAL)  fputs(" (not supported)**\n**", stdout);  **else**  err\_sys("pathconf error, path = %s", path);  } **else** {  fputs(" (no limits)**\n**", stdout);  }  } **else** {  printf(" %ld**\n**", val);  }  } |

1. 不确定的运行时限制

前面已经提及某些限制值可能是不确定的。我们遇到的问题是，如果这些限制值没有在头文件<limits.h>中定义，那么在编译时也就不能使用它们。但是，如果它们的值是不确定的，那么在运行时它们可能也是未定义的！让我们来观察两种特殊的情况：为一个路径名分配存储区，以及确定文件描述符的数目。

1. 路径名

实例：为路径名动态地分配空间

|  |
| --- |
| #include *"apue.h"*  #include *<errno.h>*  #include *<limits.h>*  #ifdef PATH\_MAX  **static** int pathmax = PATH\_MAX  #else  **static** int pathmax = 0;  #endif  #define SUSV3 200112L  **static** long posix\_version = 0;  */\* If PATH\_MAX is indeterminate, no guarantee this is adequate \*/*  #define PATH\_MAX\_GUESS 1024  char\* path\_alloc(int \*sizep) */\* also return allocated size, if nonull \*/*  {  char \*ptr;  int size;  **if**(posix\_version == 0)  posix\_version = sysconf(\_SC\_VERSION);  **if**(pathmax == 0) { */\* first time trough \*/*  errno = 0;  **if**((pathmax = pathconf("/", \_PC\_PATH\_MAX)) < 0) {  **if**(errno == 0)  pathmax = PATH\_MAX\_GUESS; */\* it's indeterminate \*/*  **else**  err\_sys("pathconf error for \_PC\_PATH\_MAX");  } **else** {  pathmax++; */\* add one since it's relative to root \*/*  }  }  **if**(posix\_version < SUSV3)  size = pathmax + 1;  **else**  size = pathmax;  **if**((ptr = malloc(size)) == NULL)  err\_sys("malloc error for pathname");  **if**(sizep != NULL)  \*sizep = size;  **return**(ptr);  } |

1. 最大打开文件数

守护进程（daemon process，是指在后台运行且不与终端相连接的一种进程，也常被称为精灵进程或后台进程）中一个常见的代码序列是关闭所有打开的文件。某些程序中有下列形式的代码序列：

#include <sys/param.h>

for( i=0; i<NOFILE; i++ )

close( i );

这段程序假定在<sys/param.h>头文件中定义了常量NOFILE。另外一些程序则使用某些<stdio.h>版本提供作为上限的常量\_NFILE。某些程序则将其上限值硬编码为20。

我们希望用POSIX.1的OPEN\_MAX来确定此值以提高可移植性，但是，如果此值是不确定的，则仍然有问题。如果我们编写了下列代码：

#include <unistd.h>

for( i=0; i<sysconf( \_SC\_OPEN\_MAX ); i++ )

close( i );

而且如果OPEN\_MAX是不确定的，那么for循环根本不会执行因为sysconf将返回-1。在这种情况下，最好的选择就是关闭所有描述符直至某个限制值（例如256）。如同上面的路径名实例一样，这样并不能保证在所有情况下都能正常工作，但这却是我们所能选择的最好方法。下面的实例使用了这种技术：

|  |
| --- |
| #include *"apue.h"*  #include *<errno.h>*  #include *<limits.h>*  #include *<stdio.h>*  #include *<stdlib.h>*  #ifdef OPEN\_MAX  **static** long openmax = OPEN\_MAX;  #else  **static** long openmax = 0;  #endif  */\**  *\* If OPEN\_MAX is indeterminate,this might be inadequate.*  *\*/*  #define OPEN\_MAX\_GUESS 256  long open\_max(void) {  **if**(openmax == 0) { */\*first time through\*/*  errno = 0;  **if**((openmax=sysconf(\_SC\_OPEN\_MAX)) < 0) {  **if**(errno == 0)  openmax = OPEN\_MAX\_GUESS; */\*it's indeterminate\*/*  **else**  err\_sys("sysconf error for \_SC\_OPEN\_MAX");  }  }  **return** (openmax);  }  int main(void) {  long t;  t = open\_max();  printf("The openmax is %ld**\n**",t);  **return** 0;  } |

1. 选项

如果我们要编写可移植的应用程序，而这些程序可能依赖于那些可选的支持功能，那么就需要一种可移植的方法来判断实现是否支持一个给定的选项。如同对限制的处理一样，POSIX.1定义了三种处理选项的方法：

* 编写时选项定义在<unistd.h>
* 与文件或者目录无关的运行时选项用sysconf函数判断
* 与文件或目录有关的运行时选项通过调用pathconf或fpathconf函数来判断

1. 功能测试宏

如前所述，头文件定义了很多POSIX.1和XSI符号。但是除了POSIX.1和XSI定义外，大多数实现在这些头文件中也加入了它们自己的定义。如果在编译一个程序时，希望它只与POSIX的定义相关，而不与任何实现定义的常量冲突，那么就需要定义常量\_POSIX\_C\_SOURCE。一旦定义了\_POSIX\_C\_SOURCE，所有POSIX.1头文件都使用此常量来排除任何实现专有的定义。

常数\_POSIX\_C\_SOURCE及对应的常数\_XOPEN\_SOURCE被称为功能性测试宏(feature test macro)。所有功能测试宏都以下划线开始。要使用他们时，通常在cc命令行中以下列方式定义：

cc –D\_POSIX\_C\_SOURCE = 200809L file.c

这使得在C程序包括任何头文件之前，定义了功能测试宏。如果我们仅想用POSIX.1定义，那么也可以将源文件的第一行设置为：

#define \_POSIX\_C\_SOURCE 200809L

1. 基本系统数据类型

头文件<sys/types.h >中定义了某些与实现有关的数据类型，它们被称之为基本系统数据类型（primitive system data type）。还有很多这种数据类型定义在其他头文件中。在头文件中，这些数据类型都是用C的typedef来定义的。它们绝大多数都以\_t 结尾。用这种方式定义了这些数据类型后，在编译时就不再需要考虑随系统不同而变的实现细节。

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | | 说明 | |
| clock\_t | | 是在滴答计数器（进程时间）（1.10节） | |
| comp\_t | | 压缩的时钟滴答（POSIX.1未定义；8.14节） | |
| dev\_t | | 设备号（主和次；4.24节） | |
| fd\_set | | 文件描述符集（14.4.1节） | |
| fpos\_t | | 文件位置（5.10节） | |
| gid\_t | | 数值组ID | |
| ino\_t | | i节点编号 | |
| mode\_t | | 文件类型，文件创建模式（4.5节） | |
| nlink\_t | | 目录项的链接计数（4.14节） | |
| off\_t | | 文件长度和偏移量（带符号的）（lseek 3.6节） | |
| pid\_t | | 进程ID和进程组ID（带符号的）（8.2和9.4节） | |
| pthread\_t | | 线程ID（11.3节） | |
| ptrdiff\_t | | 两个指针相减的结果（带符号的） | |
| rlim\_t | | 资源限制（7.11节） | |
| sig\_atomic\_t | | 能原子性地访问的数据类型（10.15节） | |
| sigset\_t | | 信号集（10.11节） | |
| size\_t | | 对象（如字符串）长度（不带符号的）（3.7节） | |
| ssize\_t | | 返回字节计数的函数（带符号的）（read、read，3.7节） | |
| time\_t | | 日历事件的秒计数器（1.10节） | |
| uid\_t | | 数值用户ID | |
| wchar\_t | | 能表示所有不同的字符码 | |

1. 标准之间的冲突

就整体而言，这些不同的标准之间配合得相当好。因为SUS基本说明和POSIX.1是同一个东西，所以我们不对它们进行特别的说明。我们主要关注ISO C标准和POSIX.I之间的差别。它们之间的冲突并非有意，但如果出现冲突，POSIX.1服从ISO C标准。然而它们之间还是存在着一些差别的。

ISO C定义了clock函数，它返回进程使用的CPU时间，返回值是clock\_t类型值，但ISO C标准没有规定它的单位。为了将此值变换成以秒为单位，需要将其除以在<time.h>头文件中定义的CLOCKS\_PER\_SEC。POSIX.1定义了times函数，它返回其调用者及其所有终止子进程的CPU时间以及时钟时间，所有这些值都是clock\_t类型值。sysconf函数用来获得每秒滴答数，用于表示times函数的返回值。ISO C和POSIX.1用同一种数据类型(clock\_t)来保存对时间的测量，但定义了不同的单位。这种差别可以在Solaris中看到，其中clock返回微秒数(CLOCK\_PER\_SEC是100万)，而sysconf为每秒滴答数返回的值是100。因此，我们在使用clock\_t类型变量的时候，必须十分小心以免混淆不同的时间单位。

另一个可能产生冲突的地方是:在ISO C标准说明函数时，可能没有像POSIX.1那样严。在POSIX环境下，有些函数可能要求有一个与C环境下不同的实现，因为POSIX环境中有多个进程，而ISO C环境则很少考虑宿主操作系统。尽管如此，很多符合POSIX的系统为了兼容性也会实现1S0 C函数。signal函数就是一个例子。如果在不了解的情况下使用了Solaris提供的signal函数(希望编写可在ISO C环境和较早UNIX系统中运行的可兼容程序)，那么它提供了与POSIX.1 sigaction函数不同的语义。后面将对signal函数做更多说明。

1. 小结

在过去25年多的时间里，UNIX编程环境的标准化已经取得了很大进展。本章对3个主要标准---ISO C、POSIX和Single UNIX Specification进行了说明，也分析了这些标准对本书主要关注的4个实现，即FreeBSD, Linux, Mac OS X和Solaris所产生的影响。这些标准都试图定义一些可能随实现而更改的参数，但是我们已经看到这些限制并不完美。本书将涉及很多这些限制和幻常量。