UNIX 标准及实现

一、 UNIX 标准化

1. 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

2. 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
```

3. 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。

4. 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）

二、 UNIX 系统的实现

1. SVR4

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

2. 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。

3. 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。

4. 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。

5. 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。

6. 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。

7. 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

三、 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。

四、 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）

1. 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：

表1 <limits.h>中定义的整型值大小

名字	说明	最小可接受值（用于16位整型系统）	典型值（32位Linux系统）
CHAR_BIT	字符的位数	8	8
CHAR_MAX	字符的最大值	127	127
CHAR_MIN	字符的最小值	-128	-128
SCHAR_MAX	带符号字符的最大值	127	127
SCHAR_MIN	带符号字符的最小值	-127	-128
UCHAR_MAX	不带符号字符的最大值	255	255
INT_MAX	整型的最大值	32 767	2 147 483 647
INT_MIN	整型的最小值	-32 767	-2 147 483 648
UINT_MAX	不带符号整型的最大值	65535	4 294 967 295
SHRT_MIN	短整型的最小值	-32767	-32768
SHRT_MAX	短整型的最大值	32767	32767
USHRT_MAX	不带符号短整型的最大值	65535	65535
LONG_MAX	长整型的最大值	2 147 483 647	2 147 483 647
LONG_MIN	长整型的最小值	-2 147 483 647	-2 147 483 648
ULONG_MAX	不带符号长整型的最大值	4 294 967 295	4 294 967 295
LLONG_MAX	长长整型的最大值	9 223 372 036 854 775 807	9 223 372 036 854 775 807
LLONG_MIN	长长整型的最小值	-9 223 372 036 854 775 807	-9 223 372 036 854 775 808
ULLONG_MAX	不带符号长长整型的最大值	18 446 744 073 709 551 615	18 446 744 073 709 551 615
MB_LEN_MAX	多字节字符常量中的最大字节数	1	16

表2 在各种平台上的ISO限制

限制	FreeBSD5.2.1	Linux 2.4.22	Mac OS X 10.3	Solaris 9
FOPEN_MAX	20	16	20	20
TMP_MAX	308 915 776	238 328	308 915 776	17 576

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。

2. 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。

表3 <limits.h>中的POSIX.1不变最小值

名字	说明:以下各项的最小可接受值	值
_POSIX_ARG_MAX	exec函数的参数长度	4096
_POSIX_CHILD_MAX	每个实际用户ID的子进程数	25
_POSIX_HOST_NAME_MAX	gethostname函数返回的主机名最大长度	255
_POSIX_LINK_MAX	指向一个文件的链接数	8
_POSIX_LOGIN_NAME_MAX	登录名的最大长度	9
_POSIX_MAX_CANON	终端规范输入队列的字节数	255
_POSIX_MAX_INPUT	终端输入队列的可用空间	255
_POSIX_NAME_MAX	文件名中的字节数，不包括终止字符null	14
_POSIX_NGROUPS_MAX	每个进程同时对添加组ID数	8
_POSIX_OPEN_MAX	每个进程的打开文件数	20
_POSIX_PATH_MAX	路径名中的字节数，包括终止字符null	256
_POSIX_PIPE_BUF	能原子地写到管道的字节数	512
_POSIX_RE_DUP_MAX	当使用间隔表示法\{m,n\}时，regexec和regcomp函数允许的基本正则表达式的重复出现次数	255
_POSIX_SSIZE_MAX	能存储在ssize_t对象中的值	32767
_POSIX_STREAM_MAX	一个进程能同时打开的标准I/O流数	8
_POSIX_SYMLINK_MAX	符号链接中的字节数	255
_POSIX_SYMLOOP_MAX	在解析路径名时可遍历的符号链接数	8
_POSIX_TTY_NAME_MAX	终端设备名长度，包括终止字符null	9
_POSIX_TZNAME_MAX	时区名字节数	6

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。

3. 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。

表4 <limits.h>中的XSI不变最小值

名字	说明	最小可接受值	典型值
NL_ARGMAX	printf和scanf调用中的最大数字值	9	9
NL_LANGMAX	LANG环境变量中的最大字节数	14	14
NL_MSGMAX	最大消息数	32767	32767
NL_NMAX	N对1映射字符中的最大字节数	未指定	1
NL_SETMAX	最大集合数	255	255
NL_TEXTMAX	消息字符串中的最大字符数	_POSIX2_LINE_MAX	2048
NZERO	默认的进程优先级	20	20
_XOPEN_IOV_MAX	readv或writev可使用的最大iovec结构数	16	16
_XOPEN_NAME_MAX	文件名中的字节数	255	255
_XOPEN_PATH_MAX	路径名中的字节数	1024	1024

4. 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。

表5 sysconf的限制及name参数（用于标识系统限制，以_SC_开始的常量用作标识运行时限制的sysconf参数）

限制名	说明	name参数
ARG_MAX	exec函数的参数最大长度（字节数）	_SC_ARG_MAX
ATEXIT_MAX	可用atexit函数登记的最大函数个数	_SC_ATEXIT_MAX
CHILD_MAX	每个实际用户ID的最大进程数	_SC_CHILD_MAX
clock ticks/second	每秒时钟滴答数	_SC_CLK_TCK
COLL_WEIGHTS_MAX	在本地定义文件中可以赋予LC_COLLATE顺序关键字项的最大权重数	_SC_COLL_WEIGHTS_MAX
HOST_NAME_MAX	gethostname函数返回的主机名最大长度	_SC_HOST_NAME_MAX
IOV_MAX	readv或writev函数可以使用的iovec结构的最大数	_SC_IOV_MAX
LINE_MAX	实用程序输入行的最大长度	_SC_LINE_MAX
LOGIN_NAME_MAX	登录名的最大长度	_SC_LOGIN_NAME_MAX
NGROUPS_MAX	每个进程同时添加的最大进程组ID数	_SC_NGROUPS_MAX
OPEN_MAX	每个进程的最大打开文件数	_SC_OPEN_MAX
PAGESIZE	系统存储页长度（字节数）	_SC_PAGESIZE
PAGE_SIZE	系统存储页长度（字节数）	_SC_PAGE_SIZE
RE_DUP_MAX	当使用间隔表示法\{m,n\}时，regexec和regcomp函数允许的基本正则表达式的重复出现次数	_SC_RE_DUP_MAX
STREAM_MAX	在任一时刻每个进程的最大标准I/O流数，如若定义，则其值一定与FOPEN_MAX相同	_SC_STREAM_MAX
SYMLOOP_MAX	在解析路名期间，可遍历的符号链接数	_SC_SYMLOOP_MAX
TTY_NAME_MAX	终端设备名长度，包括终止字符null	_SC_TTY_NAME_MAX
TZNAME_MAX	时区名的最大字节数	_SC_TZNAME_MAX

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("\tif(argc != 2)\n")
        printf("\t\terr_quit(\"usage: a.out <dirname>\");\n\n")
        FS="\t+"
        while(getline <"sysconf.sym" > 0)
        {
                printf("#ifdef %s\n", $1)
                printf("\tprintf(\"%s defined to be %%d\\n\", %s+0);\n", $1, $1)
                printf("#else\n")
                printf("\tprintf(\"no symbol for %s\\n\");\n", $1)
                printf("#endif\n")
                printf("#ifdef %s\n", $2)
                printf("\tpr_sysconf(\"%s =\", %s);\n", $1, $2)
                printf("#else\n")
```

```
                printf("\tprintf(\"no symbol for %s\\n\");\n", $2)
                printf("#endif\n")
        }
        close("sysconf.sym")

        while(getline <"pathconf.sym" > 0)
        {
                printf("#ifdef %s\n", $1)
                printf("\tprintf(\"%s defined to be %%d\\n\", %s+0);\n", $1, $1)
                printf("#else\n")
                printf("\tprintf(\"no symbol for %s\\n\");\n", $1)
                printf("#endif\n")
                printf("#ifdef %s\n", $2)
                printf("\tpr_pathconf(\"%s =\", argv[1], %s);\n", $1, $2)
                printf("#else\n")
                printf("\tprintf(\"no symbol for %s\\n\");\n", $2)
                printf("#endif\n")
        }
        close("pathconf.sym")
        exit
}
END{
        printf("\texit(0);\n")
        printf("}\n\n")
        printf("static void\n")
        printf("pr_sysconf(char *mesg, int name)\n")
        printf("{\n")
        printf("\tlong  val;\n\n")
        printf("\tfputs(mesg, stdout);\n")
        printf("\terrno = 0;\n")
        printf("\tif((val = sysconf(name)) < 0) {\n")
        printf("\t\tif(errno != 0) {\n")
        printf("\t\t\tif(errno == EINVAL)\n")
        printf("\t\t\t\tfputs(\" (not supported)\\n\", stdout);\n")
        printf("\t\t\telse\n")
        printf("\t\t\t\terr_sys(\"sysconf error\");\n")
        printf("\t\t} else {\n")
        printf("\t\t\tfputs(\"(no limit)\\n\", stdout);\n")
        printf("\t\t}\n")
        printf("\t}else {\n")
        printf("\t\tprintf(\" %%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("\tlong val;\n")
        printf("\n")
        printf("\tfputs(mesg, stdout);\n")
        printf("\terrno = 0;\n")
        printf("\tif((val = pathconf(path, name)) < 0) {\n")
        printf("\t\tif(errno != 0){\n")
        printf("\t\t\tif(errno == EINVAL)\n")
        printf("\t\t\t\tfputs(\" (not supported)\\n\", stdout);\n")
        printf("\t\t\telse\n")
        printf("\t\t\t\terr_sys(\"pathconf error, path = %%s\", path);\n")
        printf("\t\t} else {\n")
        printf("\t\t\tfputs(\" (no limits)\\n\", stdout);\n")
        printf("\t\t}\n")
        printf("\t} else {\n")
        printf("\t\tprintf(\" %%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);
          }
}
```

5. 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>
```

```c
#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);
}
```

2） 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：

```c
#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;
}
```

五、 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

六、 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
```

七、 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

八、 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 1SO 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。

九、 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。