(Micro soft 4.0版)

C 语 言

编译及开发维护工具

中 卷

北京中国科学院希望电脑公司
一九八八年三月

第一章 嵌 入 文 件

1 引言

C 库的嵌入文件只含有宏定义、常量定义、类型定义和函数说明。有些例程需要用到嵌文件中的定义和说明才能正常工作；对于某些例程，嵌入文件则是可有可无的。本章描述每一个嵌入文件的内容以及用到这些嵌入文件的例程。

有许多例程在不止一个嵌入文件中被说明。例如缓冲区管理函数 **memccpy**、**memchr**、**memcmp**、**memcpy**、**memicmp**、**memset** 和 **movedata** 是在嵌入文件 **memory·h** 和 **string·h** 中说明的。在多个嵌入文件中说明是为了保证与XENIX和UNIX以及ANSI C 标准中的嵌入文件的名字一致。因而保持了与用旧版的C 写的程序兼容，而且也进一步加强了用Microsoft C 所写程序的可移植性。

每个嵌入文件中有两组函数说明。第一组说明了函数的返回值类型以及参数类型表。这一组说明只有在需要进行参数类型检查时才使用，这在第二章“使用C 程序库”中的第2.5节“参数类型检查”中介绍过了。第二组只说明了返回值的类型，这一组用在不需要进行参数类型检查时。

嵌入文件的命名和组织方式力求达到以下目标：

- 保持与XENIX和UNIX系统以及正在发展的ANSI C 标准的嵌入文件的命名相兼容
- 反映例程在逻辑上的分类（例如把所有存储分配函数的说明都放在一个嵌入文件 **malloc·h**中）
- 使用一个给定例程时需要嵌入的文件个数最少

这些目标有时会有冲突。例如函数 **ftime** 要使用结构类型 **timeb**，**timeb** 这个结构类型在XENIX中是在嵌入文件 **sys\timeb·h** 中定义的；为保证兼容性，在MS-DOS中也采用了同名的嵌入文件。为了使用 **ftime** 时嵌入的文件最少，在 **sys\timeb·h** 中说明了 **ftime**。但是大多数时间函数都是在**time·h**中说明的。

2 assert.h

嵌入文件 **assert·h**中定义宏**assert**。使用**assert**时必须嵌入文件**assert·h**。

assert 的定义是放在预处理块 **#ifndef** 中的。如果标识符**NDEBUG**没被定义（其定义是通过 **#define**命令或编译命令行中的选项），则宏 **assert** 被定义，用来测试给定的表达式（“断言”）；如果被测试的断言为假，则打印出信息并终止程序。

如果定义了**NDEBUG**，那么 **assert** 被定义为空，这样就通过从源程序中删去**assert**的所有出现而解除了对任何程序断言的测试。因此通过定义 **NDEBUG** 就可以取消程序断言。

3 conio.h

嵌入文件 conio·h 中包含了所有控制台和I/O端口例程的说明，如下表所示：

cgets	cputs	getch	inp	outp	ungetch
cprintf	cscanf	getche	kbhit	putch	

4 ctype.h

嵌入文件 ctype·h中定义了宏和常量，还说明了一个全局数组，它们用于字符分类和处理。

ctype·h 中定义的宏如下：

isalnum	iscntrl	islower	isspace	toascii	_tolower
isalpha	isdigit	isprint	isupper	tolower	_toupper
isascii	isgraph	ispunct	isxdigit	toupper	

当用到这些宏时就必须嵌入文件 ctype · h，否则宏就没有定义。

宏 tolower 和toupper 是用条件运算定义的。它们要对参数计算两次，故对有副作用的参数会得出意料之外的结果。为解决这个问题，你可以去掉 tolower 和 toupper 的宏定义而改用它们的同名函数。关于细节请参看第四章“库例程分类”中的第4.3节“字符分类和转换”。tolower 和 toupper 的函数说明在 stdlib·h中。

除了宏定义外，文件 ctype·h中还有以下内容：

1. 用作掩码的常量定义。掩码对应于指定的分类测试。例如被定义的常量-UPPER和-LOWER分别用于测试大写字母和小写字母。

2. 全局数组_ctype 的说明。_ctype 数组是一个基于ASCII码的字符分类代码表。

5 direct.h

文件 direct·h 中包含四个目录管理函数的说明（chdir、getcwd、mkdir、rmdir）

6 dos.h

文件 dos·h 包含用在MS-DOS接口函数中的宏定义、函数说明和类型定义。

宏FP-SEG和FP-OFF用于读取或设置远指针的段和偏移部分。用到这些宏时必须嵌入文件dos·h，否则它们没有定义。

dos.h 中说明的函数有：

bdos

dosexterr

int86

int86x

intdos

intdosx

segread

文件dos.h中还定义了结构类型**WORDREGS**和**BYTEREGS**，用于定义字寄存器和字节寄存器组。这两个结构类型结合在联合类型**REGS**中。联合**REGS**作为通用寄存器类型，同时包括两种寄存器类型。

结构类型**SREGS**定义了四个域，以表示ES、CS、SS、DS这四个段寄存器的值。

结构**DOSERROR**用于存放由MS-DOS系统调用59H（在MS-DOS以及更新的版本中可用）所返回的出错值。

注意：**WORDREGS**、**BYTEREGS**、**REGS**、**SREGS**和**DOSERROR**都是成员标志名(tag)，而不是由typedef定义的类型名（关于类型定义、成员标志和typedef名字请参看《Microsoft C语言参考手册》）。

7 errno.h

嵌入文件errno.h中包含系统调用中用来设置变量errno时需要的值。在errno.h中定义的常量被函数perror作为下标用来在全程变量sys_errlist中查寻对应的错误信息。

errno.h中定义的常量以及对应的错误信息都列在附录A“错误信息”中。

8 fcntl.h

文件fcntl.h定义了用于open和sopen中的标志，它们指定对被打开的文件要进行的操作类型并且控制文件的转换方式是正文还是二进制的。用到函数open或sopen时必须嵌入fcntl.h。

9 float.h

文件float.h中定义的常量指定浮点数据类型的值域。例如，double（双精度）类型值的最多数位数（DBL-DIG = 15），或浮点类型值的最小指数（FLT-MIN-EXP = -38）。

文件float.h中还包含数学函数_clear87、_control87、_fpreset以及_status87的说明和这些函数用到的常量。

此外，float.h中还定义了浮点运算例外处理子代码，同SIGFPE一起用于捕捉浮点运算错误（见signal.h）。

10 io.h

文件io.h中包括大多数文件管理和低级I/O函数的说明，如下表：

access	dup2	mktemp	tell
chmod	eof	open	umask

chsize	filelength	read	unlink
close	isatty	rename	write
creat	locking	setmode	
dup	lseek	sopen	

函数fstat和stat是例外，它们两个是在文件sys\stat.h中说明的。

11 limits.h

文件limits.h中包含一些常量定义。它们确定整数和字符类型的数据的值的范围的。例如，类型为char的数据对象的最大值（CHAR-MAX = 127）。

12 malloc.h

文件malloc.h中包含存贮分配函数的说明：

alloca	– fmalloc	halloc	_msize	realloc
calloc	– fmsize	hfree	_nfree	sbrk
_expand	free	malloc	_nmalloc	stackavail
_ffree	– freect	_memavl	_nmsize	

13 math.h

嵌入文件math.h中包括所有浮点运算函数的说明以及 atof 函数的说明，如下表所示：

abs	bessel	fabs	ldexp	sin
acos	cabs	floor	log	sinh
asin	ceil	fmod	log10	sqrt
atan	cos	frexp	matherr	tan
atan2	cosh	hypot	modf	tanh
atof	exp	labs	pow	

文件math.h还定义了两个结构exception和complex。结构exception 用在函数 matherr 中，结构complex用于说明函数cabs的参数。

HUGE值和它在ANSI C标准中的对应者HUGE-VAL都在math.h 中定义，有些数学函数出错时就返回它们，HUGE和HUGE-VAL可用常量来实现，也可以用double类型的全局变量来实现，它们可以互换使用，HUGE或HUGE-VAL的值不能用#define 命令来改变。在整个第二部分中，我们将用HUGE代表HUGE或HUGE-VAL。

math.h中还定义了当数学函数出错时用 exception结构传递的常量。（例如 DOMAIN、SING、EDOM和 ERANGE.）

14、memory.h

嵌入文件memory.h中包括下列七个缓冲区管理例程的说明：

memccpy
memchr
memcmp
memcpy
memicpy
memset
movedata

15 process.h

嵌入文件process.h说明了所有的进程管理函数（在下面列出）；但signal除外，它是在signal.h中说明的。

abort	execvp	spawnlp
execl	execvpe	spawnlpe
execle	exit	spawnv
execlp	_exit	spawnve
execlpe	getpid	spawnvp
execv	spawnl	spawnvpe
execve	spawnle	system

process.h中还定义了一些标志，用于对spawn函数的调用中控制子进程的运行，每当你用到八个spawn族函数之一时，必须嵌入process.h以确保标志有定义。

16 search.h

文件search.h说明函数bsearch，lsearch，lfind和qsort。

17 setjmp.h

文件setjmp.h中说明了函数setjmp 和 longjmp。它还定义了与机器有关的缓冲区 jmp_buf用在setjmp和longjmp，以保存和恢复程序状态。

18 share.h

文件share.h中定义了用在函数sopen中设置文件共享方式的标志，用到函数 Sopen 时就得嵌入这文件，sopen的函数说明在嵌入文件io.h中。**注意**：函数sopen只能用在MS-DOS3.0及更新的版本中。

19 signal.h

文件signal.h中定义信号值，MS-DOS只能识别信号SIGINT 和 SIGFPE（浮点运算的异

常）。信号函数也是在 signal.h中说明的。

20 stdarg.h

文件stdarg.h中定义的宏，使用户可以存取带可变长参数表的函数，如vprintf的参数，这些宏的定义与机器无关，可移值性强并且与ANSI C标准兼容（参见varargs.h）。

21 stddef.h

文件stddef.h中定义了下列常用的变量和类型：

名称	意义
NULL	空指针（也在stdio.h中定义）
errno	存放错误信息号的全局变量（也在errno.h中定义）
ptrdiff_t	表示两指针的差的类型（int）的同义词
size_t	被sizeof返回的值的类型（int）的同义词

22 stdio.h

文件stdio.h中包含用于流式I/O处理函数的常量、宏、类型定义以及函数说明。流式I/O函数如下：

clearerr	fileno*	fseek	putchar*	sprintf
fclose	flushall	ftell	puts	sscanf
fcloseall	fopen	fwrite	putw	tempnam
fdopen	fprintf	getc*	remove	tmpfile
feof*	fputc	getchar*	rename	tmpnam
ferror*	fputchar	gets	rewind	ungetc
fflush	fputs	getw	rmtmp	vfprintf
fgetc	fread	perror	scanf	vprintf
fgetchar	freopen	printf	setbuf	vsprintf
fgets	fscanf	putc*	setvbuf	

其中带*号的是宏。

stdio.h 中还定义了很多常量，较常用的如下：

名称	意义
BUFSIZ	用于流式I/O的缓冲区必须具有不变的大小，该大小是由 BUFSIZ 定义的，BUFSIZ用于确定系统分配的缓冲区的大小，当调用 setbuf 建立用户自己的缓冲区时也必须用它。
_NFILE	常量_NFILE定义的是在某一时刻可以打开的文件的数量。五个文件 stdio, stdout, stderr, stdoux, 和 stdprn 总是打开的，故计算程序所打开的文件个数时要把它们算在内

EOF: 当文件结束（或者有时是在出错时）I/O例程就返回EOF这个值

NULL: NULL是空指针值，它在小型和中型模式的程序中被定义为0，而在大型模式程序中被定义为0L

在用户程序中可以使用上述常量，但不能改变它们的值。

stdio.h中还定义了一些内部标志用于控制流式文件操作。

stdio.h中定义了结构类型FILE，流式例程用一个指向FILE类型的指针来存取某给定的流式文件，系统利用FILE结构中的信息来维护流式文件。

FILE结构值是存放在叫做_iob的数组中的，每个文件占一个元素，因为_iob的每个元素都是对应于某流式文件的FILE结构，当一个流式文件被打开时，就被赋与一个_iob数组元素的地址（FILE指针），此后，该指针就用于引用其对应的流式文件。

23 stdlib.h

文件stdlib.h中包括下列函数的说明：

abort	ecvt	itoa	putenv	swab
abs	exit	labs	rand	system
atof	fcvt	ltoa	realloc	tolower
atoi	free	malloc	srand	toupper
atol	gcvt	onexit	strtod	ultoa
calloc	getenv	perror	strtol	

例程tolower和toupper是库函数，但它们也用宏定义实现了，其宏定义在文件ctype.h中，tolower和toupper的说明被放在#ifndef块中，它们仅在ctype.h中相应的宏定义被取消后才起作用，关于何时使用这两个例程的何种实现方式，请看第四章“库程序分类”中的第.3节“字符分类与转换”。

stdlib.h中还包括类型onexit_t的定义，以及下列全程变量的说明：

_closerrno	_fmode	_psp
environ	osmajor	sys_errlist
errno	_osminor	sys_nerr

24 string.h

文件string.h中包括下列串处理函数的说明：

memccpy	strcat	strerror	strnicmp	strstr
memchr	strchr	stricmp	strnset	strtok
memcmp	strcmp	strlen	strpbrk	strupr
memcpy	strcmpi	strlwr	strrchr	
memicmp	strcpy	strncat	strrev	
memset	strcspn	strncmp	strset	
movedata	strdup	strncpy	strspn	

25 sys\locking.h

文件locking（一般存放在子目录sys中）包含用于loking 例程中的标志的定义，每当使用locking例程时，必须嵌入文件sys\locking，以保证 locking标志有定义。

locking的函数说明包含在io.h中，**注意**：只能在MS-DOS 3.0或者更新的版本中才可以用 locking。

26 sys\stat.h

文件sys\stat.h（通常存放在子目录sys中）包含函数fstat和stat返回的结构的类型和用于维护文件状态信息的标志，还包括fstat和stat的函数说明，当用到函数fstat和 stat时，必须包含文件 sys\stat.h以确保相应的结构类型（名叫stat）有定义。

27 sys\timeb.h

文件timeb.h（通常存放在子目录 sys 中）定义结构类型 timeb，且说明用到该类型的函数ftime，当使用函数ftime时必须嵌入文件timeb.h，以保证timeb结构类型有定义。

28 sys\types.h

文件types.h（通常存放在子目录sys中）中包含用于返回文件状态和时间信息的系统调用中的类型，当包含了文件sys\stat.h，sys\utime.h或返sys\timeb.h时，也必须包含文件sys\types.h。

29 sys\utime.h

文件 utime.h（通常存放在子目录sys中）定义结构类型 utimbuf，且说明用到该类型的函数utime，当使用函数utime时必须嵌入文件utime.h以保证类型utimbuf有定义。

30 time.h

文件time.h说明函数asctime，ctime，difftime，gmtime，localtime，time 和 tzset，（函数ftime和utime分别是在sys\timeb.h和sys\utime.h中说明的）。

time.h还定义了结构类型tm(函数asctime，gmtime和localtime 用它)，和类型 time-t（函数 difftime中用到）。

31 varargs.h

文件varargs.h中定义了存取带可变长参数表的函数如 vprintf 的参数的宏，这些宏定义

与机器无关，可移植性强，而且与UNIX系统V兼容（参见 stdarg.h）。

32 v2tov3.h

文件v2tov3.h是为那些从Microsoft C 编译器2.03版以及更旧的版本改用第3.0版的用戶提供的，有一些2.03版编译器提供的程序库中的例程在3.0版中是用略为不同的方式提供的，嵌入了v2tov3.h这个文件后，这些例程就可以不加修改直接在第3.0版中使用。

文件v2tov3.h与Microsoft C编译器3.0版同其他版本的另外一些区别在《Microsoft C编译器用戶指南》一书的附录F中有详细讨论。

文件 v2tov3.h 中有三个有用的宏，宏abs求取其参数的绝对值，宏min和max 分别计算两个参数的最小值和最大值，详细情况请参看v2tov3.h文件本身。

第二章　C程序库

abort

● 说明

```
# include   <process.h>      用于函数说明
# include   <stdlib.h>       用process.h或stdlib.h
void abort( );
```

● 功能

abort在stderr上输出信息

"Abnormal program termination"

然后终止调用它的进程，把控制权返回创建此调用进程的进程（通常是操作系统）。abort 不将流式文件的缓冲区清仓。

● 返回值

把终止状态 3 返回父进程或操作系统。

● 参见

execl, execle, execlp, execlpe, execv, execve, execvp, execvpe, exit, _exit, signal, spawnl, spawnle, spawnlp, spawnlpe, spawnv, spawnve, spawnvp, spawnvpe

● 范例

```
# include   <stdio.h>
main(argc, argv)
int argc;
char * argv[ ];
{
```

```
FILE *stream;
if((stream = fopen(argv[argc-1], "r")) = = NULL){
    fprintf(stderr, "%s couldn't open file %s\n", argv[0], argv[argc-1]);
    abort( );
    }
    /*注: MS-DOS 3.0及更新版本, argv[0] 包含程序名, 在以前的版本, argv[0]包
    含串 "C" */
```

命令行例子:

```
update employ.dat
```

输出:

```
C:\BIN\UPDATE.EXE couldn't open file employ.dat
Abnormal program termination
```

abs

● 说明

```
#include <stdlib.h>      用于函数说明
int abs(n);
int n;                   整型值
```

● 功能

abs返回它的整型参数n的绝对值。

● 返回值

abs返回它的参数的绝对值，没有错误信息返回。

● 参见

cabs, fabs, labs

● 范例

```
#include <stdlib.h>
int x = -4, y;
y = abs(x);
printf("%d\t%d\n", x,y);
```

输出:

```
-4    4
```

access

●说明

```
# include <io.h>                    用于函数说明
int access(pathname, mode);
char *pathname;                     文件或目录的路径名
int mode;                           权限值
```

●功能

对于文件来说，access首先检查所指定的文件是否存在，以及是否可以用所给的mode 存取。下面给出mode的允许值和在调用access 中的意义：

值	意义
06	检查是否允许读和写
04	检查是否允许读
02	检查是否允许写
00	仅检查是否存在

在 MS—DOS 下，所有存在的文件都具有可读性，因此方式00和04的效果一样；同样，在MS—DOS下，可写性意味着可读性，因此方式06和02是等价的。

对于目录来说，access只检查所指定的目录是否存在；在MS—DOS下，所有的目录都具有可读性和可写性。

●返回值

access返回 0 值，说明文件具有所给的方式；返回 -1值，说明文件不存在或不能以所给的方式存取，这时errno为下面给出的值的其中之一：

值	意义
EACCES	存取方式错误：即文件的权限不允许所指定的存取方式。
ENOENT	文件或路径名不存在。

●参见

chmod, fstat, open, stat

●范例

```
# include  <io.h>
# include  <fcntl.h>
int fh;
...
```

```
if ((access("data", 2)) = = -1){
  perror("data file not writable");
  exit(1);
  }
else
  fh = open("data", O_WRONLY);
```

acos

● **说明**

```
# include  <math.h>
double acos(x);
double x;
```

● **功能**

acos 返回 x 的反余弦值，此值的范围是从 0 到 π。x 值必须在 -1 和 1 之间。

● **返回值**

acos 返回反余弦值。如果 x 小于 -1 或大于 1，那么acos 把 errno置成EDOM，把错误信息 "DOMAIN error" 传送给stderr，并且返回 0。

可以用matherr子程序修改错误处理方式。

● **参见**

asin, atan, atan2, cos, matherr, sin, tan

● **范例**

在下面的例子中，只要输入的值不在 -1和 1 之间，程序就不断给出输入的提示。

```
# include  <math.h>
int errno;
main( )
    {
    float x,y;
    for (errno = EDOM; errno = = EDOM; y = acos(x)){
        printf("Cosine = ");
        scanf("%f", &x);
        errno = 0;
        }
    printf("Arc Cosine of %f = %f\n", x,y);
    }
```

输出举例

```
    Cosine = 3
    acos: DOMAIN error
Cosine = -1.0
Arc Cosine of -1.000000 = 3.141593
```

alloca

● 说明

```
#include <malloc.h>                    用于函数说明
char *alloca(size);
unsigned size;                         从栈中分配的字节数
```

● 功能

alloca从程序的栈中分配size个字节。当调用alloca 的函数退出时，所分配的空间自动被释放。

● 返回值

alloca返回一个指向所分到的空间的 char 类型的指针。返回值指向的存贮空间保证适合于任意类型的对象的存储。要挣到不同于char的某种类型的指针，就要对返回值做强制类型转换。如果没有空间可分配，就返回NULL。

● 参见

calloc, malloc, realloc

● 注释

由alloca返回的指针值不能作为free例程的参数。还有，因为alloca 对栈进行操作，所以它只能用于简单的赋值语句，不能用于作为函数参数的表达式。

● 范例

```
#include <malloc.h>
int *intarray;
/*在栈上分配10个整数*/
intarray = (int *)alloca(10 * sizeof(int));
```

asctime

● 说明

```
#include <time.h>
```

```
char *asctime(time);
struct tm *time; 指向在time.h中定义的结构的指针
```

● **功能**

asctime 把以结构类型存放的时间转变成一个字符串。time 值通常是通过对 gmtime 或 locatime的调用获得，gmtime和localtime均返回一个指向tm结构的指针，tm在time.h中定义。（参看gmtime对tm结构的域的描述）。

由asctime 得到的字符串结果包括26个字符，具有如下形式：

Mon Jan 02 02:03:55 1980\n\0

时钟是以24小时为周期。所有的域具有一定的宽度。换行符('\n')和串结束符('\0')出现在字符串的最后两个位置上。

● **返回值**

asctime 返回指向字符串结果的指针。没有错误返回。

● **参见**

ctime, ftime, gmtime, localtime, time, tzset

● **注释**

asctime和 ctime 用一个静态分配的缓冲区存放返回的字符串。对这些子程序之一的每次调用都要冲掉前一次调用的结果。

● **范例**

```
# include <time.h>
# include <stdio.h>
struct tm *newtime;
long ltime;
...
time(&ltime);                          /*取时间(秒数)*/
newtime = localtime(&ltime);           /*将时间转换成本地时间存入 struct tm中*/
printf("the current data and time are %s\n",asctime(newtime));
                                       /*以串方式打印本地时间*/
```

asin

● **说明**

```
# include <math.h>
double asin(x);
double x;
```