



IBM PC 译丛

C 语 言

武高郭	占庆光	峰茂汉	译
武	占	峰	校

辽宁省电子计算机学会

编 者 的 话

这本资料说明Optimizing = C86编译程序，有关的程序和支持程序库。它不是语言参考手册，它将帮助用户准备C程序，根据CPM-86，MPM-86，IBM PC-DOS，MS-DOS，SB-DOS和86DOS进行编译和执行。

黄乙本

C 语言 V2.0 用户手册

目 录

概述

导引	(1-1)
C68 2.00版特征	(1-2)
安装指南	(1-3)
DOS文件	(1-3)
文件系统	(1-6)
标准文件	(1-7)
语言信息	(1-8)
转换到V2.0	(1-9)
V2.0 I/O程序库	(1-10)
汇编语言功能块	(1-10)
汇编语言支撑的代码	(1-10)
复盖	(1-11)
须知和其它注解	(1-11)
运行时错误信息	(1-12)
删除标准函数	(1-12)
在DOS下的存贮器配置	(1-14)

程序

CC 1	—予处理程序	(2-1)
CC 2	—语法分析程序	(2-2)
CC 3	—代码生成	(2-3)
CC 4	—优化处理程序	(2-3)
arch	—源程序归档管理	(2-3)
marion	—库管理	(2-4)

程序库

导引	(3-1)
_default	(3-2)
\$entry	(3-2)
_exit (value)	(3-2)
_main ()	(3-3)

abort (format, args...)	(3—4)
abstoptr (address)	(3—5)
alloc (size)	(3—5)
atof (string)	(3—6)
atoi (string)	(3—7)
basicget (stream, buff, buflen, fieldptr, fieldcnt)	(3—7)
bdos (fcode, dx)	(3—8)
calloc (nelem, elsize)	(3—9)
Ceil (arg)	(3—10)
chdir (pathname)	(3—11)
Chmod (filename, mode)	(3—11)
Clearerr (stream)	(3—11)
Close (fd)	(3—12)
Coreleft ()	(3—13)
Creat (filename, mode)	(3—13)
Crt_cine (X1, Y1, X2, Y2, Color)	(3—15)
Crt_mode (mode)	(3—15)
Crt_rdor (row, Column)	(3—15)
Crt_srcp (row, Column, page)	(3—15)
Crt_wdot (row, column, color)	(3—16)
exit (value)	(3—16)
exp (arg)	(3—16)
fabs (arg)	(3—17)
farcall (offset, segment, srv, rrv)	(3—17)
fclose (stream)	(3—18)
ferror (stream)	(3—18)
fflush (stream)	(3—19)
fgetc (stream)	(3—20)
fgets (buffer, bufleng, stream)	(3—20)
floor (arg)	(3—21)
fopen (filename, fomode)	(3—22)
fprintf (stream, format, args...)	(3—23)
fputc (byte, stream)	(3—25)
fputs (string, stream)	(3—25)
fread (where, elsize, nelem, stream)	(3—26)
free (pointer)	(3—27)
frexp (val, eptr)	(3—28)
fscanf (stream, format, args)	(3—28)

fseek (stream, offset, base)	(3—30)
ftell (stream)	(3—31)
ftoa (value, buffer, iplaces, fplaces)	(3—32)
fwrite (where, elsize, nelem, stream)	(3—32)
getc (stream)	(3—33)
getchar ()	(3—34)
getw (stream)	(3—34)
index (string, cc)	(3—35)
inportb (portno)	(3—36)
inportw (portno)	(3—36)
intrinit (func, stack, vecno)	(3—36)
isalnum (cc)	(3—37)
isalpha (cc)	(3—37)
isascii (cc)	(3—37)
isctrl (cc)	(3—37)
isdigit (cc)	(3—37)
islower (cc)	(3—37)
isprint (cc)	(3—37)
ispunct (cc)	(3—37)
isspace (cc)	(3—37)
isupper (cc)	(3—37)
iswap (inta, intb)	(3—39)
itoa (n, buffer)	(3—39)
itoh (n, buffer)	(3—40)
longjmp (envp, value)	(3—40)
ldexp (mantissa, exponent)	(3—41)
loadexec (filename, param, funcode)	(3—41)
log (val)	(3—42)
log 10 (val)	(3—42)
lower (string)	(3—42)
lseek (fd, offset, base)	(3—42)
ltell (fd)	(3—44)
ltoa (value, buffer)	(3—45)
ltoh (value, buffer)	(3—45)
ltos (n, buffer, base)	(3—46)
main (argc, argv)	(3—47)
makefcb (filename)	(3—48)
makefnam (input, default, output)	(3—49)

malloc (size)	(3—50)
mkdir (pathname)	(3—50)
modf (val, iptr)	(3—50)
movblock (soffset, sseg, doffset, dseg, count)	(3—51)
movmem (source, dest, count)	(3—51)
open (filename, mode)	(3—52)
outportb (portno, byte)	(3—54)
outportw (portno, word)	(3—54)
peek (offset, seg)	(3—55)
pokeb (offset, seg, byte)	(3—55)
pokew (offset, seg, word)	(3—55)
pow (x, y)	(3—56)
printf (format, args, ...)	(3—57)
ptrtoabs (address)	(3—57)
putc (c, stream)	(3—57)
putchar (c)	(3—57)
puts (string, stream)	(3—58)
putw (w, stream)	(3—58)
qsort (array, number, width, cmpf)	(3—59)
read (fd, buffer, count)	(3—60)
realloc (oldp, size)	(3—61)
rename (form, to)	(3—62)
rindex (string, cc)	(3—62)
rmdir (pathname)	(3—63)
sbrk (size)	(3—63)
scanf (format, args)	(3—64)
segread (rv)	(3—65)
setjmp (envp)	(3—65)
setmem (addres, count, value)	(3—66)
sprintf (string, format, args)	(3—67)
sqrt (val)	(3—68)
sscanf (string, format, args)	(3—68)
strcat (strto, strfrom)	(3—70)
strcmp (str 1, str 2)	(3—70)
strcpy (to, from)	(3—71)
strlen (string)	(3—72)
strncat (string 1, string 2, n)	(3—72)
strncpy (to, from, n)	(3—73)

sysint (vector, sreg, rreg)	(3—74)
sysint 21 (sreg, rreg)	(3—75)
tolower (c)	(3—75)
toupper (c)	(3—76)
trigonometric functions	(3—76)
ungetc (c, stream)	(3—78)
ungetch (c)	(3—78)
unlink (filename)	(3—79)
upper (string)	(3—79)
utoa (a, buffer)	(3—79)
wqsort (n, cmpf, xchgf, date)	(3—80)
write (fd, buffer, count)	(3—82)

OPTIMIZING C86 用户手册

导引

欢迎您使用“C”和OPTIMIZING C 86 编译程序。我们希望，你将发现这是一个有用的和令人满意的产品。如果你有什么问题或疑难，请打电话、电传或写信。

资料

这本手册为把 Computer Innovations公司的Optimizing C 86 编译程序用作使用C语言的工具提供所需要的资料。它假定你具备了C语言，机器和操作系统的基本知识。你还必需有 Kernighan 和Ritchie 的 C 语言付本。

如果你正在学习C语言，你将发现K&R中的各例不修改通常就能运行。虽然 K&R 是一本好书，但我们建议你也要有一些目前所能得到的该语言的其它书，这些书提供有价值的技巧。

语言特点

C86支援所有的语言特点，Kernighan和Ritchie 中的所有程序都可运行。我们的程序库包括K&R中所记载的所有的标准程序库的函数，精选的UNIX V 7 例行程序，以及一系列机器和操作系统相关的功能块。同时它们允许你发挥你的计算机的最大效能和多数 C 代码的可移植性。

操作系统

Optimizing C86编译程序在两个不同操作系统下在8086或8088处理机上运行。在这本手册中我们自始至终用“CPM”表示CPM86，CONCURRENT—CPM86和MPM86，用“DOS”表示MS—DOS，IBM PC—DOS或SB—DOS。

随操作系统所提供的文献是全面了解某些程序库函数所必需的。

硬件

要使编译程序运行，需要128Kb的内存。它包括你的操作系统所用的16Kb的存储容量。用于编译程序，实用程序和一些工作区的磁盘空间至少需要 256Kb，推荐你采用两个磁盘驱动器。编译程序可以在一个驱动器的 IBM DOS的配置下运行，但对头脑清醒的人来说这是冒险的。

技术保障服务

为得到技术保障服务你可以打电话，电传或写信。如果问题不是很复杂的话，打电

话你将得到最好的效果。如果写信，请写明电话号码及前去的方便时间，我们对所写来的问题的答复慢得多，但对海外用户给予特殊的优先答复。从星期一到星期五上午9点到下午6点之间我们正常营业。但是星期六，星期天和晚上的另星时间也常常上班。

如果你需要报告问题，请你编一个显示问题的小的实例程序。我们需要下列资料：

※你所配置的磁盘上的编译程序版本和顺序号。

※编译时所用的开关符和在联接步骤中所用的程序库。

※你所使用的硬件。

※你的操作系统的名字和版本。

如果你打电话，尽量把你的机器准备好。有时一些简单的检查可节省我们两方面工作时间。

如果你是一种故障的第一个报告人，我们将给你免费更新（因故障修理），通常在24小时内修完。

如果你要我们回电话，请记住，由于电话号码出错或无人接，则大约有10%接不通。如果在48小时内我们没给你打电话，再给我们打一次。

我们的用户小组是获得援助、代码和帮助的另一条途径。我们将使用告示板，它能使用户们互相交往、共享代码。这时我们第一台BBS样机在运行，但不令人满意。很幸运，在84年一月底以前它将全部运行。

C86 2.00 版特征

如果你的代码在C86的早期版本（V1.33）下编译过，并遵循C语言的一切规则，那么它可以在V2.00版本下编译并运行。

这一版的主要变化是：

※代码数量比前版小10%到20%，而速度快一倍。注意速度增加达4倍。

※大型内存开关符允许程序达到存储器代码区和数据区的内存极限。性能试验（到目前为止很有限）表明，具有相当多指针的典型大存储量的程序，在我们的旧的1.33版编译程序下以大约相同的速度运行。

※现在这个编译程序产生IBM（或微软）型目标文件。这些文件可以用正规的DOS联接程序链接。因此我们不再提供我们自己的联接程序，即CVTOBJ程序。

※本编译程序有一任选功能，这个功能可以使编译程序产生能用MASM汇编的汇编源代码。

※浮点运算使用内部8087代码，实际上运行较快。

※为最大限度地使用8087代码，8087的三角和数学程序库已用汇编语言重新编码。

这一领域速度的增长真令人惊奇。

※包括有具备DOSV2.00全部优点的I/O程序包。这就完全可能存取其他目录中的文件。

※对于DOS和IBM PC的机器相关的支持已扩展到具有基本绘图支持。许多其他函数已加到程序库中。

※我们提供DOS格式目标程序库和一个用以保养这些程序库的程序（marion）。

向Optimizing C86 输入代码

许多用户向optimizing C86输送了在UNIX和其他C语言程序下编译的代码。除随同某些编译程序所提供的一些很不标准的I/O程序库外,通常很少或没有包括转换能力。

我们设置某些编译时的开关符的企图是减少移植中的这种问题。欲知详情,请参看程序说明部分。

安装指南

Optimizing C86编译程序包已在一片或若干片带写保护的软盘上提供。没写在给你配置的软盘上。如果软盘每一面上都有标签,那就是双面软盘。翻过来可读另一面。

你应该格式化一片或几片磁盘,这些磁盘用来保存编译程序的不进行紧缩的主付本。这要遵照操作系统命令去做。

然后,把已格式化的软盘放入驱动器X中,并把录有编译程序的主盘放入驱动器Y中。在下列指令中你应该用你的系统上的实际驱动器的字母代替X和Y。通常他们是A和B。

要设X做为缺省驱动器就打入“X”:

要取回驱动器Y中的配置磁盘的目录就打入“dir Y:”你应该看看文件名,这些文件名与在标题“DOS FILES”下面列出的文件名相似。对于所有的文件(除了前面两个文件),其文件扩展名的第二个字符将是“q”的,它表明该文件为“压缩”格式。这种格式减少装载文件所需要的大量的磁盘空间,并且也提供检验文件内容的检查和。

请寻找含有“read.me”文件的磁盘。可用打入“type Y: read.me”来显示这个文件。如果我们需要提供任何附加信息的话,就记在这个文件中。请遵照本说明做。

通过打入“Copy Y: usq.exe/v”可以把不压缩程序“usq.exe”复制到工作盘上。对于每一片“非压缩”格式主软盘都需这样做。

然后对于主软盘上其文件扩展名的第二个字母为“q”的每个文件都运行非压缩程序。例如,为了不进行压缩主软盘上的文件“stdio.hq”应打入“usq Y: stdio.hq”。本文件的非压缩版将被放置到驱动器“X”上,并带有文件名“stdio.h”。对于每一个已压缩的主文件都要这样做一遍。请注意,“非压缩”文件通常要比“压缩”文件大得多。据我们已收到的某些问题的报告表明,不是某台机器的主要功能出了问题,就是配置的磁盘错了。如果你断定,配置的磁盘错了,请与你的供应者或与我们直接联系。

最后,在你的非压缩格式的主软盘上加写保护与标签。

DOS文件

MS-DOS, IBM PC-DOS和SB-DOS配置的磁盘包括如下的文件。请注意,只有前两个文件是按“非压缩”格式装载的。

read.me	最终版本的说明和注释
usq.exe	非压缩程序

※ <code>stdio. h</code>	标准的头部文件
※ <code>cc1. exe</code>	编译程序第一趟
※ <code>cc2. exe</code>	编译程序第二趟
※ <code>cc3. exe</code>	编译程序第三趟
※ <code>cc4. exe</code>	编译程序第四趟
※ <code>cslib. lib</code>	可重定位的目标程序库 (小内存型的)
<code>clib. arc</code>	源程序库 (基本支援和DOS—ALL)
<code>alib. arc</code>	源程序库 (基本汇编支援)
<code>alibb. lib</code>	适用大内存型的汇编过了的alib
<code>2clib. arc</code>	源程序库 (DOS 2 支援)
<code>2alib. arc</code>	源程序库 (DOS 2 支援)
<code>2alibs. lib</code>	适用小内存型的汇编过了的 2 alib
<code>2alibb. lib</code>	适用大内存型的汇编过了的 2 alib
<code>8087. arc</code>	源程序库 (8087数学)
<code>8087s. lib</code>	适用小内存型的汇编过了的8087
<code>8087b. lib</code>	适用大内存型的汇编过了的8087
<code>arch. exe</code>	源程序库维修程序
<code>marion. exe</code>	可重定位的程序库维修程序

传送文件

从非压缩主盘把你所需要的文件复制到工作盘上去。初次使用时你将需要前面标有星号 (“※”) 的文件, 其余文件这次不需要。

你也应该把其它必要的实用程序 (如文本编辑程序) 复制到你的工作磁盘上去。

建立测试程序

建立一个源程序。虽然任何一个建立标准文本文件的编辑程序都应该是令人满意的。但我们建议你使用随操作系统所提供的编辑程序。我们以 “hello.c” 为例, 其内容如下:

```
# include "stdio.h"
main ( )
{
    printf ("hello.world\n");
}
```

编译

要编译程序, 打下列四行命令:

```
×: cc1 Y: hello
```

```
X: cc2 Y: hello
```

```
X: cc3 Y: hello
```

```
X: cc4 Y: hello
```

这里磁盘驱动器“X:”含有编译程序。而磁盘驱动器“Y:”含有源程序。你的系统中的正确的指示符代入上述命令行。这些指示符通常是“A:”和“B:”对于驻留在缺省驱动器（通常是驱动器“A:”）上的文件可以省略它们。

这一过程的结果将是“Y: hello.obj”文件。

联接

联接该程序以便得到可执行的程序，则应打：

```
X: link Y: hello, con/map, X: cslib
```

我们假设程序库（“cslib.lib”）同程序在同一磁盘上。参阅DOS手册可以得到使用联接程序的更多信息。

执行

执行程序应打“Y: hello”（没引号），接着按回车键（或enter键）。然后程序将在控制台上打印信息“hello, world”。

批文件

为减少运行编译程序所必需打入命令的次数，可以把若干条运行编译程序的命令置入“批”文件中。所有程序都回送结束状态，从而DOS2.0在及其以后版本的控制下，出现错误而终止时，可以调整批文件。

如果你正在用DOS pre-V2.0 版本编译程序，就要去掉所有以“if”开始的命令行以及“:done”行。因为这个编译程序在早期版本的DOS上不能自动停止。而必须手动停止批文件。对于小内存型，在没有跟踪，内部8087，或者汇编源输出的情况下进行编译，我们所采用的批文件为：

```
cc1 %1
if errorlevel 1 goto dohe
cc2 %1
if errorlevel 1 goto done
cc3 %1
if errorlevel 1 goto done
cc4 %1
if errorlevel 1 goto done
Link %1, con/map, cslib
: done
```

为得到更多的信息，请参看你的操作系统文献。

文件系统

给出 I/o 软件包的目的是把 UNIX 的类似接口提供给程序员。这一节提供的信息帮助你了解设计的含义。

标准的 UNIX 系统有两组输入/输出函数。第一组通过操作系统提供 I/o 缓冲几乎不为程序员使用。第二组提供程序内部缓冲和大批服务。

我们已提供两套函数。它们两者都使用程序内缓冲，因而这种服务不是我们所支援的操作系统提供的。否则它们同 UNIX 基本程序所要求的相反。下面各节提供更多资料。

基本服务

下述函数提供基本服务：

open	打开现有文件。
creat	清除现有文件，然后打开一个新（空的）文件。
close	关闭文件。
read	读一个文件的字节。
write	把字节写到文件上。
lseek	在一个文件中定位。
ftell	报告文件中现行位置。

所有上述函数都通过文件描述符识别文件，描述符由 **open** 和 **creat** 函数回送，并送给所有其它函数。按 UNIX 约定，文件描述符是小的非负数，而 **open/creat** 必须回送尽可能小的文件描述符以供调用。你可以完全相信这一点。因此你可以预测由 **open/creat/close** 调用序列而回送的文件描述符。

流的服务

所有其它的输入/输出函数都使用一种流的标识符。它可提供种种有效的服务，包括格式化输入和输出。在可移植性和通用性方面，这些函数可为你的程序使用。

流的标识符可由函数 “**fopen**” 回送。根据 UNIX 约定，流的标识符是对类型 “**FILE**” 的指针，这里的 “**FILE**” 在 “**stdio.h**” 中已有意义。实际的意义是与实现相关的，同时与我们的 DOS-ALL 程序库的区别在于出版了 2.0 版程序库。你不要假设你的程序中的文件描述符和流的标识符之间的相互关系。

DOS 2.00 和以后的版本

在这一版中我们已提供了两个 I/o 系统。第一种系统可以在各种版本的 DOS 上运行并称为 DOS-ALL。除校正在 DOS 的任何版本下重新定向处理外，它提供了在 DOS 1.1 下能得到的全部服务。它不支持 DOS 2.0 下的路径名。可是你将发现我们大量分别销售的 ALLAS 软件包缓和了这些问题。

在 2 clib和 2 alib中配置的 i/o 软件包代替了基本软件包中某些函数,并可完全使用 DOS 2.0的 I/O 系统,包括路径名。可是在以前的DOS 2.0 操作系统上不能运行该软件包。精选在于你。

文件打开方式

由于DOS 和UNIX具有不同的行的结尾和文件结尾的约定,我们必需设计出二进制和ASCII数据之间的区别。在文件打开逻辑中已这样做了,以便最大限度地减少程序转换的操作。因此对文件来说我们有ASCII和BINARY打开方式,一旦选定了某种方式,除非你在一个文件中把ASCII与二进制数据混杂在一起,否则程序毫无问题地运行。

DOS 字符设备

我们为“CON:”,“PRN:”和“AUX:”提供特殊的处理。“CON:”,“PRN:”和“AUX:”分别指的是控制台,打印机和辅助通信设备口。这些文件不能以更新的方式打开。但在一个程序中可以不止一次地打开。Bdos调用 1 至 5 用于以缓冲的方式传送这种数据。当前在DOS 2 程序库下它们没有得到支持。

控制台输入

如果控制台(“CON:”)以ASCII输入方式打开,那么为它提供特殊处理。在这种情况下,我们总是从键盘上输入一个完整的输入行。因此用户可以采用编有字符的行。请注意,对键盘打开的文件不共享一个公共缓冲区,因此交错读各个对键盘打开的文件可以产生意外的结果。在将来的版本中可能改变这个问题。

这种机理保证了控制台输入是每次读一行。因此你的程序将等待,直到出现回车才开始处理。如果你想单个地输入字符,而不要回车,你可以从二进制方式打开通往控制台的另一条通道,或使用bdos()调用,它们也会响应你的控制。

标准文件

在程序初始化时打开三个文件,文件名分别为:stdin, stdout 和 stderr。这些文件可以根据它们的名字所表示的功能而被程序采用。按照缺省,这些文件是以ASCII方式对控制台(“CON:”)打开。

通过一种方法可以改变stdin和stdout的缺省值,这种方法,即人们熟知的在引用程序的命令行上重定。在命令行中规定重定向如下:

programme <newin> newout 其它参数

该命令行使“programme”从文件“newin”而不是从键盘读它的输入,并且向文件“newout”而不是向控制台写它的输出。文件的名字可以是任何合法的名字或任何一个字符设备文件名。当然你不能打开打印机作输入。

如果stdin和stdout两者都重定向到同一个磁盘文件,通常在读所有的输入之前,输入文件将被破坏。

你也可以把数据附加到现有的ASCII文件末端,使用命令如下:

programme»appout 其它参数

如果文件appout不存在, 该命令会建立它。

磁盘文件

所有的磁盘 i/o 都是缓冲的。在CPM下缓冲区为1024字节的块, 在DOS下为 512 字节的块。缓冲区长度与你的操作系统相匹配, 则可得到最佳性能。通过修改程序库中“FILEIO.H”文件内的“SECSIZE”和相关的符号以及重新编译整个程序库的函数(包括文件“FILEIO.H”)可以改变磁盘缓冲区长度。我们正在研究使得这个值在运行时成为动态可变的。

语言信息

下列信息对于C86程序设计环境的所有定义是必要的。它应与K&R结合起来使用。

数据类型

已支持的数据类型和它们的长度是:

char	8 位
unsigned char	8 位
short	16位
unsigned short	16位
int	16位
unsigned int	16位
pointer	16位或32位
long	32位
unsigned long	32位
float	32位 (8087格式)
double	64位 (用于各种计算)

在小型内存中的指针长16位, 在大型内存中的指针长32位。在一个程序中的所有指针长度必需相同。

即使你正使用浮点软件, 浮点数据存贮也要用8087格式。浮点数据有一个 8 位的指数, 允许数最大约为 $1e + 38$ 。双精度有 11位指数, 允许数最大约为 $1e + 308$ 。在使用或传递给函数之前, 浮点数总是要转换成双精度。所有的浮点计算都是双精度的。

存贮类型

auto, extern和static与在K&R中定义相同。目前寄存器类的转换成 auto, 但是当我们实现更多的最优化逻辑时, 这一点也将改变。如果你现在在写新代码, 那么你就必须应用适合新版编译程序的寄存器类。

EXTERN定义

关键字“extern”的用法依编译程序的不同而有所变化。我们的定义取自 K&R206 页第11.2项。定义为:

在没有“extern”这个关键字的情况下,任何外部数据项必须出现一次。这种项目可以具有初始值。这就导致为该数据项保留存贮区。

所有其它项目必须包括关键字“extern”。因而可以不含有初始值。它们不会导致保留存贮区。

程序调试

绝大多数的调试都可通过增加打印语句并重新编译程序来做。你可以用你的标准调试包调试程序。你可以从联接程序得到存贮器映象。然后你可以把断点放到一个函数的第一条指令上,而检查局部变量和参数。我们已发现,在大内存型下使用DOS调试包比在小内存型下困难得多。因此如果可能的话,请在小内存型下调试。

无论什么时候调试失败了,那么由编译程序输出的汇编源程序能帮助你找到故障。请注意,由于汇编程序中的问题,汇编了的汇编源代码可能与直接目标代码不那么准确地匹配(虽然后者与前者在功能上相同)。典型的例子,汇编程序可以为同一指令选择不同操作代码。例如,它可以用常规的三个字节的转移指令,而汇编程序将使用二个字节的短的转换指令。

加上几个打印语句和重新编程序,通常是较容易的。不久可得到符号调试程序。

转换到V2.0

对程序库进行的各种改变的目的是使Optimizing C86符合K&R的标准。如果你发现,有异常情况,你应该检查程序库文件乃至检查程序库源代码定义的改变。几乎没有什么改变,但是2.0和DOS-ALL程序库有小差别是可能的。

指针

如果你正在把运行的代码转换成大内存型,你最可能出现的错误是误用指针。由于大内存型使用16位整数和32位指针,指针和整数不再是等价的。你必须说明指针是所产生的正确代码的指针。你也必须说明回送指针的函数应回送指针。由于编译程序在检查这样的错误方面不提供很多帮助,这是一个值得注意的问题。

8087支持

本版编译程序产生内部8087代码,也包括一个直接使用8087的三角和数学软件包。请注意,这个程序库与在供浮点软件使用的我们的C语言的三角和数学函数截然不同。这样做的结果是使你的浮点代码运行大约快三倍。由于这套代码是新的,它可能产生意想不到的结果。我们建议,使用软件浮点程序库运行相同的代码,以便在最后检查输出阶段时校验结果。

2.0 I/O 程序库

本程序库是为代替标准 DOS-ALL 程序库而提供的。它提供了附加功能，例如，改变目录、编制目录和支持全路经名。这将是最新公布的标准程序库，唯有这个程序库增强了大多数 I/O 的性能。如果有可能，请使用这个程序。

汇编语言功能块

这一节提供写汇编语言函数所需要的基本资料。这些资料可能随着编译程序新版本的发行而有所改变。如果你必须使用汇编代码，要使功能块简短扼要。大多数问题在 C 语言中可以解决。

函数调用规定

在 G 语言或汇编语言中要调用函数，采用下列规定。

全部汇编语言代码内应包括文件 PROLOGUE.P。它定义了一组为使联接程序准确工作所必需的标准段名。如果你构造任何附加的段名，一定要确保它们被联接到适当的位置。运行时系统构成在存储器中关于段的次序的假定。

调用函数则把最左边的参数后压入堆栈，而把最后边的参数先压入堆栈。因此最左边的参数在堆栈的“顶部”。所有字符参数在被放入堆栈之前都被转换成整数。

寄存器 ax, bx, cx, dx, si 和 di 可为调用函数使用。并且没有保存或恢复。在将来的版本中可能改变这一点。

小内存型通过“近”（“near”）调用引入被调函数，大内存型通过“远”（“far”）调用引入被调函数。

必须预先保留寄存器 cs, ss, ds, es 和 bp。寄存器 bp 是帧指针。

回送结果

回送 char, short 和 int 型值于寄存器 ax 中。回送 long 型值于寄存器 ax 和 ux 中，ax 保存最低有效数字。

回送双精度的数于 ax, dx, bx 和 cx 中，其中 ax 保存最低有效数字，cx 含有指数。最近的新版本将有所改变。

因为被调函数并不知道提供了多少参数，所以参数是通过调用函数从栈中弹出参数。

所有名字以 \$ 符号开始的函数是内部的编译程序认识的函数，不必遵循这些规定。这些规定是可以改变的，要仔细读现存的代码以确定所使用的规定。它们因功能不同而异。

汇编语言程序的例子，请参阅汇编语言程序库。

汇编语言支持代码

配置磁盘含有对于汇编语言程序员来说是有价值的三个文件。文件“model.h”应