Lattice 8086/8088
C编译程序功能说明手册

郭 洪、卢 环、相建序 译
郭 洪 校

中国计算机技术服务公司
1986

Lattice 8086/8088
C编译程序功能说明手册

郭 洪、卢 环、相建序 译
郭 洪 校

中国计算机技术服务公司
1986年

目　录

第一章　在MS-DOS下的执行过程

Lattice 8086/8088C编译程序在Microsoft公司的MS—DOS操作系统下运行时，接受用C编程语言编写的程序，然后生成可重定位的、具有8086目标模块格式的、可使用Microsoft连接程序的机器代码。程序库定义了一系列在MS—DOS下可运行的与多数Unix标准函数兼容的子程序，在Kernighan和Ritchie的书中曾给予过描述。

8086指令组适用于象C这样的高级语言，Lattice编译程序产生的机器代码充分保留了8086的特点。8086可支持一兆字节的寻址空间，但却缺乏对存贮器直接和有效的寻址能力，这就将C程序的程序部分限制在最大64K字节，数据部分（包括静态数据，自动数据或堆栈数据以及可自动分配的存贮单元）最大64K字节。虽有这些限制，仍可开发相当复杂和功能强的程序（包括编译程序本身）。

1.1　操作命令

在MS—DOS下运行的Lattice编译程序由以下文件组成：

LC1.EXE	C编译程序阶段1
LC2.EXE	C编译程序阶段2
FXU.EXE	函数抽取实用程序

C.OBJ	C程序进入/退出模块
LC.LIB	运行时I/O程序库
LC.BAT	执行阶段1和阶段2的批文件
STDIO.H	标准I/O首部文件
CONIO.H	控制台I/O首部文件
CTYPE.H	字符串宏首部文件
FTOC.C	华氏到摄氏程序举例
CAT.C	文件衔接程序举例
IO.ASM	汇编程序例题

所有这些磁盘文件占用160K字节的磁盘空间。每个编译阶段程序本身有50K字节，另要求最少14 K字节数据区。因此除MS—DOS占用的存贮区外，编译程序需要至少64K字节的工作区。若编译大型源文件，可能需要更大的存贮区。（MS—DOS操作系统占用略多于16K字节的存贮区）。

LC1和LC2是编译程序的主体，他们分别完成编译处理的一部分。必须用不同的命令去启动他们，在LC1完成编译任务后并不能自动装入LC2。批处理文件LC.BAT就用于连续执行LC1和LC2，以便在LC1运行后如果源文件无错就立即执行LC_2。其编译过程如下：

文件.C→LC1→文件.Q

文件.Q→LC2→文件.OBJ

LC1读入一个带.C扩展名的C源文件，如果没有发现严重错误就生成一个文件名相同而具有.Q扩展名的中间文件。LC2读入由LC1生成的中间文件并产生一个具有相同文件名带.OBJ扩展名的目标文件。编译一完成，LC2删除.Q文件。通常，在每个阶段中生成的输出文件都放在有输入文件的同

一驱动器中。需注意，当一个源文件中定义有多个函数时，他的目标文件也就包含了同样多的函数。在进行程序连接时，不可能分别将函数由这一目标文件中分离出来，详看1.3.2。

要生成程序可执行文件，必须把.OBJ文件输入到连接程序去。除了用户生成的.OBJ文件外，连接处理还涉及另两个指定的文件。整个连接过程可概括如下：

C.OBJ+user.OBJ+…+LC.LIB→
Link→User.EXE

可见，这两个指定文件是C.OBJ和LC.LIB。有两点提请注意：第一，C.OBJ文件一定要是LINK命令中指定的第一模块，以便定义经Lattice C编译后生成程序的入口和出口。第二，LC.LIB是必须被指定的程序库，他定义所有运行时I/O库函数是Lattice C程序包的一部分。连接时，用户必须给出所有参加连接的.OBJ文件名，并指定连接程序从而生成可执行文件的名称。

在此通过对一个把华氏温度转换为摄氏的程序的编译、连接和运行来说明对一个C源程序的全部处理过程。现在我们认为所有.EXE文件（LC1、LC2和LINK）驻留在同一驱动器中并规定命令用大写字母表示，尽管小写字母命令同样工作。（注意，这儿所说的连接程序是Microsoft连接程序版本1.10。如果你有不同版本，请参考该公司文件。一般来说，以下描述适用于任何版本的连接程序。）

第一步：要执行编译的第一阶段，请键入：

LC1 FTOC 〈CR〉

注：文件名不必带.C扩展名，若带.C亦可正常工作。

第二步：在LC1完成，MS—DOS提示符出现时，键入：

LC2 FTOC 〈CR〉

以执行编译第二阶段。同样不必指定扩展名。

第三步：在LC2完成，系统提示符出现时，键入：

LINK C FTOC 〈CR〉

以执行连接程序。

注：无论连接什么C程序都一定要指定C.OBJ为LINK命令中的第一目标模块。FTOC（刚由LC2产生的FTOC.OBJ）是用户目标模块。然后出现下列提示：

RUN FILE [C.EXE]: FTOC〈CR〉

List file [NUL.MAP]: 〈CR〉

Libraries [LIB]: LC 〈CR〉

由此生成的可运行文件名为FTOC.EXE，不要求生成连接报表，使LINK到LC.LIB去搜寻外部访问。

第四步：键入：

FTOC〈CR〉

来运行由上步生成的可执行文件。一个华氏和摄氏温度值对照表将输出到用户控制台。

注：前两步可用一个命令LC FTOC 〈CR〉来完成，即使用批命令LC.BAT连续执行LC1和LC2。这时仍有FTOC.OBJ文件存在，但应可被擦除。

有关编译、连接和运行程序的详细介绍请看下几节，其中编译各阶段的处理过程参考1.3。

在说明各种命令行格式时，用字段来描述命令行中的一串非空格字符。可供选择的字段用方括号括起来。请学习每

节后面的例子以便清楚命令的实际格式。

1.1.1 阶段1

编译程序的第一阶段读入C源程序而后生成一个叫做“quadruples”或者“quads”的逻辑记录中间文件。启动编译程序第一阶段的命令格式是：

LC1 [=Stack] [>Listfile] filename [Options]〈CR〉

对命令行中的各项将按他们在命令行中出现的顺序予以说明。选择项用方括号括起来。头两个选择项是对所有C程序通用的命令行选择项（请参照1.1.4）。

=stack 第一个选择项用于人为控制为堆栈保留的字节数（参考讨论C程序结构的1.4），缺者值是2048(十进制）字节，这对大多数程序足够了。控制堆栈容量的字段一定要是LC1后的第一个字段，其形式是等号后的一个十进制数值(如，=4096即分配给堆栈的十进制字节数为4096）。由于编译程序用递归来处理C语句，因此多重嵌套语句会使编译占用更多的堆栈空间。若有一个多重嵌套的源程序（如多重If-Else结构）在第一阶段编译中途莫明其妙地锁死、或抱怨有出乎意料的错误出现、或其他反常现象，那末增加堆栈空间可能是解决问题的途径。此外，在你发现此编译程序本身的缺陷时，请参考附录B及时将此类问题报告本公司。在存贮容量有限的系统中，可以削减堆栈容量，但必须力图避免“Not enough memory”的错误发生。但当堆栈容量低于1024字节时，很难保证编译的成功。

>Listfile 第二个选择项用于将第一阶段信息引导到指定文

件中去。这些信息包括编译登录信息、出错信息和警告信息。必须指定包括扩展名在内的文件全名。如果文件名已存在，此文件将被再次修改、使用。这个选择项有助于观察一列长表中的出错信息。

filename　这是命令行中唯一必不可少的字段，他指定被编译的C源文件名，不带.C扩展名，因为第一阶段会自动提供此扩展名。注意，只有.C扩展名的源文件才能被编译，若指定了其他扩展名，编译程序不予理睬，并试图找到具有.C扩展名的同名源文件。（不过*include 后面的文件名必须带有扩展名。）若不用—O选择项指定别的驱动器，中间文件生成在缺省驱动器上，即放有源文件的驱动器上。文件名用大、小写均可。

option　负号后跟单字母用来表示编译时间选择项。键入字母必须是小写，否则无效。必须分别用负号和字母说明各个选择项（而不允许各选择项像在某些unix程序中那样被组合在一起）。现行选择项有：

-a　迫使编译程序取缔基于指针的优化过程。一般来说，编译程序认为通过指针访问的对象与直接访问的程序部分中的对象不同。此选择项可取这种假设。只有当程序员想用指针耍些花招时，否则别轻易使用-a选择项。详看1.3.4。

-b　按字节排列计算所有的地址补偿值。第一阶段将非字符类型的目标码按字编排以保证8086中有效的数据存取（在8086中存取奇地址上的一个字需附加4个时钟周期）。这一选择项使8088处理机可有效地分配程序存贮区，并便于说明某些特殊

记录层次中的结构，在这种结构中字数据项必须置放于奇字节地址补偿值上（例MS—DOS中的FCB结构，他在奇地址上保存长整数）。

-C 注释按无嵌套处理。平常编译程序认为注释可以嵌套从而允许对大部分程序加以注释。这个选择项使用户可以强制编译程序按标准、非嵌套方式操作。

-d 使中间文件包含诊断信息，特别是行分隔符记录。这使第二阶段产生一个输入行号与程序部分地址补偿值的对照表。这个选择项的使用使中间文件不会在第二阶段被删除（注意，早期的编译版本没有此选择项）。

-od 将中间文件输出到驱动器d上，d是大、小写皆可的单字母字符，用于指定磁盘驱动器（a表示A：b表示B：）。驱动器字符必须紧邻“-o”不能插入空格。

-x 将外部说明（在函数体之外所做的说明）的缺省存贮类型由外部定义改变为外部访问。外部说明的一般含义是为说明对象分配存贮区而并不指定存贮类型并使他可被其他文件识别。-x选择项与以“extern”引导的说明作用相同，即说明的对象是在某个其他文件中。BDS C编译程序具有同样的选择项，详看索引C。

举例

1. LC1 XYZ -ob -x

对文件XYZ.C执行第一阶段编译，在B：中生成文件XYZ.Q，不必考虑存贮类型，只把所有外部说明译为

“extern”说明。

2. LC1 =4096>tns.err tns

对输入文件TNS.C执行第一阶段编译，在当前登录的磁盘上生成TNS.Q中间文件，建立4096（十进制）字节的堆栈区，并产生一个TNS.ERR文件以容纳编译程序产生的全部信息。

1.1.2 阶段2

编译阶段2读入阶段1生成的中间文件并生成一个标准MS-DOS格式的目标文件。要详细了解执行过程请参考1.3.2。启动编译阶段2的命令格式是：

LC2 filename [option]〈CR〉

与阶段1命令格式十分相似，分配堆栈容量和列表文件选择项仍可使用，但由于很少使用因而不在此详细介绍。编译程序的任一阶段都不对标准输入进行处理，所以〈选择项在两个编译阶段皆无效（请看1.1.4有关一般C程序运行选择项一节）。

filename 这个不可缺的字段被用于指明在编译第一阶段生成的带.Q扩展名的中间文件。但字段中的中间文件名却不必带.Q扩展名，第2阶段会自动提供。数字字母字符大、小写皆可。如不特意指定其他驱动器，就使用缺省驱动器。换句话说，若不使用-o选择项，目标文件就生成在具有中间文件的驱动器中。

options 编译时间选择项格式为负号后跟一单个小写字母，键入的大写字母无效。各个选择项必须分别用负号后的单字母来说明，而不能象在某些Unix程序中被组合。现有选择项：

-f　　使代码具有8087数字数据处理器的浮点运算格式。注意，这个选择项必须对同一程序中的所有浮点运算函数使用，不允许把使用-f选择项编的函数与没有用-f选择项编译的浮点函数组合在一个程序中（注：在早期的编译版本中无此选择项。）

-od　　把目标文件输出到d驱动器中。在此d是单字母字符，大小写皆可，用以指定磁盘驱动器(a表示A：、b表示B：等）。驱动器字符必须紧邻-o，不能插入空格。

举例

1. LC2 A：NXF -f

对输入文件A：NXF.Q执行编译阶段2，在驾动器A：上生成NXF.OBJ文件，产生适用8087浮点运算处理器的代码。

2 LC2 u790 -oc

对输入文件u790.Q执行编译第二阶段，在驱动器C：上生成u790.OBJ文件。

1.1.3 程序连接

在组成一个程序的所有源模块都被编译之后，必须将他们连接在一起形成最后可执行文件。此步必不可少的理由如下：

1. 编译阶段2生成的目标文件处于不可运行状态。

2. 多数程序都要使用一些不在当前模块中定义的函数，必须将他们连接在一起程序才可运行。这些外部函数在是由用户定义的情况下，必须被编译为.OBJ文件以供使用或在编译程序提供的程序库中定义他们（第3章介绍了可供移

植的函数，1.5节只介绍了那些在MS-DOS下定义的函数。）

3. 一般认为C程序是由被定义为“main”的函数开始执行，实际上在调用“main”函数之前，必须完成大量与系统相关的处理，完成这一处理的模块是在连接时组装到程序中来的。

尽管连接的一般概念是涉及对外部函数调用的处理，但C也允许函数存取在其他模块中定义的数据单元。这种访问之所以可能是由于目标码所支持的外部连接结构建立起外部标记与存贮单元间的联系，这个标记就是用来说明C程序中目标码的标识符。要注意的是，程序员必须在定义一个对象和访问这个对象的不同模块中用同一属性来描述他，因为连接程序并不能核实存取类型而只是根据外部标记访问相应存贮单元。用*include文件来做公共外部说明通常可以避免这类错误。

对一个程序进行连接时要求直接地或间接地把该程序的所有有关部分都输入到连接程序中去。要求三种输入：

1. 文件C.OBJ必须是连接的第一模块，他定义了凡是采用Lattice C编译程序进行编译的C程序在MS-DOS下的入口。

2. 用户生成的函数是必须包括一些附加模块在内。这些模块包括主模块，以及在其他源模块中定义的任何其他函数。

3. 文件LC.LIB必须被指定为连接期间要搜寻的程序库。

MS-DOS下的Microsoft连接程序规定这些输入是：

1. “C”是“Link”命令中的第一模块

2. 在“Link”命令中，用户目标文件在“C”之后，

不需要扩展名。

3. 键入“LC”来响应连接中出现的“Libraries”提示。

别忘记，在“LINK”命令中必须有、只能有一个含有“main”的主模块。

如果连接程序没能找到“LINK”命令中提到的某个.OBJ文件，就会停止处理而没有任何.EXE文件生成。如果连接程序在指定的.OBJ文件中没能找到所有的外部项，就会产生另一种错误信息“Unsatisfied external reference”并跟有一串没被发现的外部项名。（此时，只有当你确信程序不会调用一个短缺的函数时，才可运行一个存在以上错误信息的程序）。

关于外部名请看1.2.2。有关目标代码特性请看1.4。如果你的系统所用连接程序具有与这儿不同的提示，请参考Microsoft手册。一般来说对其他种类提示的缺省响应是正确的。在你的连接程序可以产生公共标记映象的前提下，也许你想生成一个a.MAP文件看看结果模块的成份。

举例

```
LINK  C  XYZ  QRS
      Run  File  [C.EXE]:  XYZ<CR>
      List  File  [NUL.MAP]  <CR>
      Libraries  [LIB]:  LC  <CR>
```

即执行连接程序、产生XYZ.EXE可执行程序。这个程序包含文件XYZ.OBJ和QRS.OBJ。

1.1.4 程序运行

运行一个C程序时，“main”函数是首先调用的函数。但在执行主函数前，需要完成两个重要的服务。

1. 分析运行程序命令，将从命行中得到的信息做为参数送给“main”主函数（关于所进行的分析和参数特性将在后面加以讨论），从而使命令行输入到程序的处理过程更易实现。

2. 打开缓存的文本文件“Stdin”（标准输入）、“Stdout”（标准输出）和“Stderr”（标准错误）以供用户程序使用，一般把这三个文件都分配给用户控制台，但也可以用以下将谈及的命令行选择项把“Stdin”和“Stdout”分配给别的设备从而使利用标准“getchar”和“putchar”宏函数进行文件输入或输出的程序应用更灵活、广泛了。

要运行一个C程序，最简单的方法是键入. EXE文件名（不必带扩展名. EXE）然后回车。因由命令行向程序输入是如此方便，所以程序运行请求常常还包含其他信息。运行C程序命令行的格式是：

pgmname [=stack] [<infile] [>outfile]
[args] <CR>

程序名后方括号内的都是选择项。若需要某个选择项必须按所示顺序加以说明。

pgmname　这个字段说明运行程序名就是程序连接时生成的.EXE文件名，但不带.EXE扩展名。

=stack　选择项的头一个字段用于指定程序运行时需为堆栈保留的十进制字节数，其缺省值是2048。表示堆栈容量的十进制数须紧跟在等号之后。被说明为“auto”的目标都在堆栈分配内存，一旦程序由说明目标的函数返回到调用者，这些内存单元就被释放出来。这种动态分配内存使予测一个特定程序实际所需堆栈空间变得十分困难。而命令行中的堆栈容量选择项使用户不必重新编译程序就可调整其堆栈容