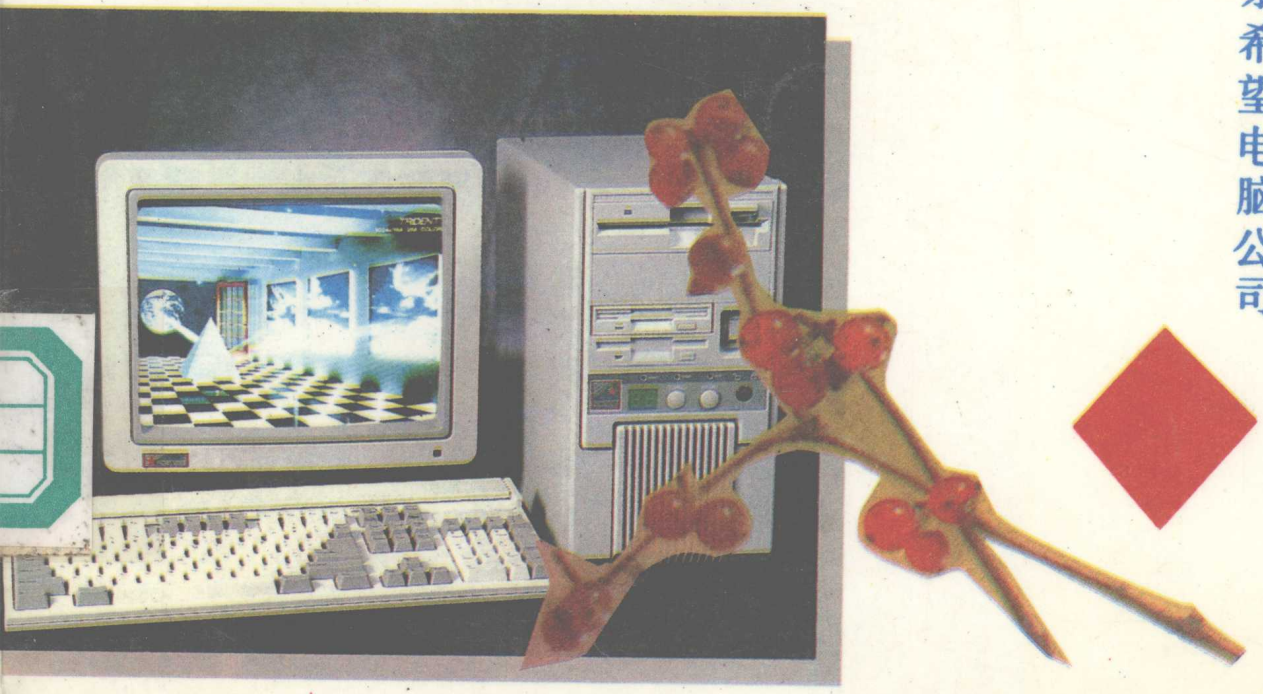
Clipper高级编程技巧与实例

（上册）

廖斌　张林　编著

东岳　木杉　审校



北京希望电脑公司

Clipper 高级编程技巧与实例

(上册)

廖 斌　　张 林 编著

东 岳　　木 杉 审校

北京希望电脑公司

内容摘要

Clipper 是 dBASE 语言的最新的编译型版本，它自问世以来，对微型机数据库管理产生了深远影响。本书对 Clipper 高级编程技巧作了详细介绍，并提供了丰富的实例。全书内容共分十九章：概述、Clipper 基本概念、Clipper 语言及环境、编译和连接应用程序、连接文件的覆盖、Clipper 除错程序、数组、SET KEY TO 指令、用户界面、备注字段的处理及操作、有效的查询方法、直接处理文件、网络、在 Clipper 中使用 C 语言、利用 C 和汇编语言编写用户自定义函数、文件结构、Clipper 公用程序、屏幕和窗口技术、数据驱动技术。在附录部分则对 Clipper 命令、函数、错误信息、警告信息、ASCII 字符集、INKEY()函数返回值、SET 函数等作了介绍。

本书内容新颖、论述全面，不失为计算机软件开发人员的一本难得的参考书。

欲购本书的用户可直接与北京 8721 信箱联系，电话 2562329，邮政编码 100080。

Clipper 高级编程技巧与实例
(上册)

廖 斌　张 林 编著

东 岳　木 杉 审校

北京希望电脑公司　双青印刷厂印刷

开本：787×1092 毫米　1/16　印张：52.9　字数：1083 千字

京准印字：3572—9157 2　内部定价：40.00元/套

前　言

Clipper 自问世以来，对微型机数据库管理产生了深运影响．Clipper 是 dBASE 语言的最新的编译型版本，它是一种开发工具．Clipper 使用 dBASE III Plus 的扩充作为它的标准命令集．Clipper 是第一个提供用户定义函数的 dBASE 语言产品，也是第一个提供用户定义命令的 dBASE 产品．此外，Clipper 的扩充系统已被加强以允许更容易地存取用 C 和汇编语言编制的例程和函数．最后 Clipper 还包括一个强有力的、用户可存取的编译预处理程序，一个新的连接程序和一个新的调试程序．

本书不同于一般的 Clipper 使用手册，它着重于编程技巧和实例介绍．全书内容共分十九章：概述、Clipper 基本概念、Clipper 语言及环境、编译和连接应用程序、连接文件的覆盖、Clipper 除错程序、数组、SET KEY TO 指令、用户界面、备注字段的处理及操作、有效的查询方法、直接处理文件、网络、在 Clipper 中使用 C 语言、利用 C 和汇编语言编写用户自定义函数、文件结构、Clipper 公用程序、屏幕和窗口技术、数据驱动技术．此外在附录部分对 Clipper 命令、函数、错误信息、警告信息、ASCII 字符集、INKEY()函数返回值、SET 函数等作了介绍．

本书内容新颖、论述全面．不失为计算机软件开发人员的一本难得的参考书．

本书的出版得到了北京希望电脑公司的大力协助，在此表示衷心的感谢．此外，由于时间仓促，不当之处在所难免，尚望读者提出批评指正．

作者

目　录

第一章　概述

本章将概要的介绍 Clipper 的功能．包括解释器(Interpreter)与编译器(Compiler)的不同，它们对程序的影响以及 Clipper 比 dBASE Ⅲ多提供的功能．

1.1 解释器(Interpreter)与编译器(Compiler)

如大家所知，dBASE Ⅲ和 Clipper 使用共同的语言，并把程序存在所得的 PRG 文件中，不同的地方是 dBASE Ⅲ是个解释器，而 Clipper 是个编译器．现在让我们看看 Clipper 存在的原因及其价值，首先我们看 dBASE 如何执行一个程序，每当程序在 dBASE 中执行时，dBASE 系统会一次又一次的将每个命令转成计算机可执行的格式，然后再执行．如此的作法，在程序发展的过程或程序经常需要修改时，是非常有弹性的．但当程序发展完成，不再需要修改时，象 dBASE 这样一次次的转换命令，就显得速度太慢，不切实际．

为了解决 dBASE 速度慢的缺陷，Clipper 采用了编译器的结构——一次就将所有命令转成计算机的可执行格式，再执行．它的作法是这样的，它先将 PRG 程序转成机器可接受的中间格式——目标(OBJ)文件，再将目标文件单独或与其它目标文件(可以为 Clipper，C 或组合语言所产生)连结成可执行文件(EXE 文件)．以后不论 dBASE 或 Clipper 是否存在于电脑中，只要计算机使用 DOS 系统，就可以执行这个 EXE 文件．

上面所谈到连成连结目的的程序称为[连结器(linker)]．Clipper 软盘所提供的连结器是由 Phoenix 公司提供的 Plink86 连结器．Plink86 的特色是稍慢，但提供覆盖(overlay)的功能．所谓覆盖的功能是指，它能使比存储体大的程序也能正常的经由执行码的转移而在较小存储上执行．

若你不需要覆盖的功能也可使用其它的连结器；如 DOS 的 link，或 Borland 编译器所提供的 Tlink，……，等．它们的速度将比 Plink86 快，但当你使用其它连结器时，要注释它们所提供的格式不一定与 Plink86 相同．

当你使用 EXE 文件时，可能会发现二件事，一是它的速度较快，另一是它非常大．速度快的原因是它利用编译器及连结器将解释器所需不断转换命令的时间完全省掉了，所以速度较快．同时，因为可执行文件是个计算机可执行的格式，人几乎无法读懂，所以它还可以防止程序外露为别的所知．

程序会因为各种需求而愈来愈大，所以不论如何管理存储体，它总有一天会不够用．因此，Plink86 提供你解决程序比存储体大的问题．在 Summer'87 版的手册中，提供了有关覆盖的文件．

有两种方式可以使用覆盖，一是用 OVL 文件详细的将要覆盖的模块分配好，另外是将这些模块附在 EXE 文件之后．这两种方式都可以正常运作，就算程序超过 640K，计算机也会自动在需要时才输入被覆盖的部分．

接下来让我们看看产生 EXE 文件的两个步骤：编译与连结。

1.2 编译

最简单的编译法为：将程序名交给编译器，然后 Clipper 自动将程序内所有 DO 所呼叫的 PRG 文件包含进来，一起编译产生 OBJ 文件。

假设一个程序 ACCOUNTS 利用 DO 调用 SCREEN 和 REPORT 二个副程序，则

Clipper ACCOUNTS

将如编译 ACCOUNTS 般的编译 SCREEN 和 REPORT，并且在目前的目录下产生 ACCOUNTS.OBJ 文件。若 SCREEN 和 REPORT 中有任何一个不存在，它会显示一个错误信息"cannot open assumed external"，然后继续往下执行。

当程序调用副程序或用户自定函数时，可以用 SET PROCEDURE TO 告诉 Clipper 到何处找它们。编译器可以自动找到并编译它们。和 dBASE 解释器不同的是，Clipper 不需要清楚的列出文件与副程序的关系。所以 SET PROCEDURE TO 在执行完全没有使用作用。

你也可以选用 Clipper 提供的参数来控制编译器的运作，它们详列如下：

• -m 参数指示编译器不要包含其它文件。所以当上例改成：

Clipper ACCOUNTS -m

它将产生只含 ACCOUNTS 的 OBJ 文件，你可以另外编译 SCREEN 及 PEPORT，然后再用连结器产生可执行文件。

• 你也可以用-o 参数指示编译器将产生的 OBJ 文件放到其它子目录内。下例

Clipper ACCOUNTS -m, -o d: \objdir

将 ACCNTS.OBJ 文件放入 d: \objdir 目录中，等待连结。

• -s 参数指示编译器只检查语法，不产生 OBJ 文件

• Clipper 的编译器以行号显示语法错误的行，而调试器(debugger)对程序的追踪与状态显示也以行号为主。-l 参数允许用户除去 EXE 文件中的行号，如此每行可以节省三个字节。

• 为了使 Clipper 更快。它提供-t 参数，让用户能将编译器产生的暂存文件放入 RAM 磁盘中。如下

Clipper ACCOUNTS -t d

使用 D 磁盘机暂存文件。

Clipper 也允许用户将文件名放入 CLP 的批处理文件中，一次编译许多文件。批处理文件的使用法是在 CLP 文件前加上一个@，例如，COMP.CLP 文件包含下面两行：

ACCOUNTS

REPORT

可以用下行编译它

Clipper @ COMP

可此将产生 COMP.OBJ 文件。

上面的编译方式看起来就象选用-m 参数一样。所以 COMP.OBJ 文件不包含

SCREEN．PRG．Clipper 编译器在编译时，程序所使用的变量与常数如果超过一定数量，就可以使用上述的方法产生不同的模块，再让连结器加结各模块．

1.3 连结

连结器的功能是将各 OBJ 文件组合起来，形成一个可执行的 EXE 文件．

Nantucket 公司特别为 Clipper 修改了 Plink86 的功能，使它能自动连结 CLIPPER.LIB．Clipper 磁盘上所提供的 Plink86 就是这个修订版．

Tlink 与 link 具有相同的格式，先列出 OBJ 文件，然后是 EXE 文件，MAP 文件最后才是程序库．当省略 EXE 文件及 MAP 文件时，它会自动有个内定值，但是程序库文件一定要完全列出不可省略．假设前述 accounts.obj，report.obj，及 screen.obj 都已存在，下列命令就是使用 link 产生可执行文件 acct.exe4 方法：

link accounts report screen，acct，＼clipper＼clipper

Think 的格式完全与 link 相同．

若使用到扩增程序库内的功能，就一定要加入命令列中如下：

link accounts report screen，acct，，＼clipper＼clipper＼clipper＼extend

Plink86 需要的格式与它们不同．它需要三个命令指定各种资源：1. files，详列所有的 OBJ 文件，2. library，指示编译器到何处寻找程序库，及 3. output，如果可执行文件的名称与第一个 OBJ 文件不同，则用来指示可执行文件的名称，这些命令也可缩写成 fi，lib，out．它们所提供的信息与其它两个连结器相同．

在 DOS 下使用 Plink86 有两种方法，一是直接在 DOS 提示字下打入 Plink86，它会出现＝号，提示你输入 OBJ 文件，程序库及可执行文件名．另一种更常用的方式，是将 Plink86 与 OBJ 文件，程序库及可执行文件名打成一行．例如，我们将含有 report 及 screen 的目标文件 account 连结成可执行文件，可以打入：

Plink86 file ACCOUNTS library EXTEND

它将会连结 ACCOUNTS.OBJ，EXTEND.LIB 及 CLIPPER.LIB，并产生 ACCOUNTS.EXE

下一行：

Plink86 out ACCT fi ACCOUNTS lib EXTEND

将产生 ACCT.EXE．

大部分的程序设计师会将某个特定用途的所有目标文件放在一个子目录中．因此最方便的方式，是将包含 Plink86 的子目录放入 PATH 中．

Plink86 也可以认出在 PATH 中定义的文件名，它会在 DOS 的环境变量 OBJ 内所设置的目录中，找寻它所需要的目标文件及程序库．例如，下面的 DOS 命令：

SET OBJ＝＼OBJS；＼CLIPPER

指示 Plink86，如果在目前的目录下找不到某些 OBJ 文件，可以到＼OBJ 及＼CLIPPER 两个子目录中找．如此可以将目标文件及程序库放入你所喜欢的子目录中．

在程序发展的过程中，子程序会愈来愈多．因此，在编译及除错时，指令会愈来愈长，且愈来愈复杂．Plink86 提供一个方式，可以象编译时使用 CLP 文件一般的使用连结

器．它是 LNK 文件．

例如，可以建立 ACCOUNTS.LNK．它的内容如下

fi ACCOUNTS

out ACCT

lib EXTEND

然后在连结时打入下述命令即可:

Plink86 @ ACCOUNTS

如果你想多了解如何使用 LNK 文件及覆盖，请参考 Nantucket 公司所提供的文件．

1.4 程序库

程序库是用来存储共用程序的地方，它包含 OBJ 文件及一个指示各 OBJ 文件内有那些程序的索引．使用程序库的好处是，程序库内不被使用的 OBJ 文件，不会被放入 EXE 文件内，当一个 OBJ 文件被使用到，它才会被引入 EXE 文件中．如此，使用到某个程序库内提供的功能时，只要连结一个很小的 OBJ 文件，便可节省很多空间．

在此推荐一个程序制作的好技巧，就是使用 Microsoft 公司提供的工具 LIB 来建立自己的程序库．你可以将 Clipper 编译器产生的 OBJ 文件，结合成一个 LIB 文件．

如你所知 CLIPPER.LIB 是一个程序库，它包含了如索引、网络、排序等各种不同功能的 OBJ 文件及各文件内的各相关函数．那么你可能会问，在如此的细分下，为了像下列只有一行的程序:

x = 1

为什么会产生 160K 的可执行文件？为什么 Clipper 不产生一个只包含所需模块组的 EXE 文件？这是一个顶难回答的问题．首先，必须提示一点：这个例子是经过精心设计的．我们都知道 Clipper 是一个处理数据库的语言，所以就算是最简单的程序最少也会处理一个数据库，也就是说它必须具有数据库处理的函数，甚至必须有索引处理的函数，还有一些函数也是必须的，包括内部堆栈的处理及存储体管理．这些函数是 EXE 文件所必备的．(注：也就是说 Clipper 所产生的 EXE 文件必须包含一个 dBASE Ⅲ PLUS 系统)．更进一步说，如果表达式含有一个宏，编译器在执行时并无法确定它将调用那一个函数，所以必须将所有可能的函数都串入 EXE 文件中．一个实际的程序并不是象从 1 算到 10 那么简单，所以必须把大部分程序库都放进去．

毋庸置疑的 Clipper 可以将可执行文件的大小设置到某个极限．譬如，可用旗号指定宏不可用于函数调用中，或让你定义函数是外建或内含的．但是实际的应用上，程序的大小并不算差．对基本所需的 160K 而言，再加入的程序码实在是很小．况且对目前以千万字节为单位的存储体而言，什么样的大小才叫小？

虽然编译器所产生的 OBJ 文件可能没有使用到程序库中所有的文件及数据，但是 Clipper 程序库中大部分的函数还是要放入 EXE 文件中．这不是和前面所述程序库的特性不同吗？Clipper 是如何完成这一点呢？它的解决方式是在 OBJ 文件中放入一个特殊的函数，此函数在程序库中包含了所有必须的 OBJ 文件．当然，程序库中还是有一些视需要才放入 EXE 文件的 OBJ 文件．例如，SORT，REPORT 和 LABEL FORM 就不一定

要放入 EXE 文件内.

1.5 外界函数

Clipper 将 PRG 文件中没有定义的函数及副程序的名称都放入 OBJ 文件中，称为"外界函数"．它们可能存在其它 OBJ 文件或程序库中，而在连结时才被放入文件中．若连结时没有这些副程序，连结器会指示错误的地方．

当函数或副程序出现在宏，REPORT 或 LABEL FORM 时，编译器便无法知道它的存在，也就无法明确地指明它是外界函数．所以在连结时连结器无法正确的将它放入 EXE 文件中．因此在执行时，如果调用这个函数就会产生执行时段的错误．如果它存在某个 OBJ 文件中，可以在连结时放入 EXE 文件．如果它是 LIB 文件中某个未被引用的 OBJ 文件，可以用 EXTERNAL 命令来连结它．

对宏，REPORT，或 LABEL FORM 中未被连结的函数，可以用 EXTERNAL 将它包含入文件中．最常见的例子是 extend.lib 中提供的各函数．例如，常被称为 HARDCR 的 HARDER.FRM，如果程序没有使用到就必须用 EXTERNAL 声明．

1.6 使用 MAKE

Summer '87 版的 Clipper 比以前的版本快了很多．但是重覆的编译，连结整个应用程序，仍旧要花掉很多时间．大部分节省时间的办法是记录下那些程序有修改那些没有，然后只编译那些修改的程序．

但追踪存储那些程序有修改，那些没有，是件很乏味的事．所以 Summer '87 版提供一个能自动记载程序修改的时间与日期的工具，称为 MAKE．对于熟悉 Boarland 或 Unix 上的 MAKE 的人而言，学习 Clipper 可能有些困难，因为它们有些不同．Clipper 的 MAKE 比较类似于 Microsoft 公司所提供的 MAKE．

Make 执行的原理是利用了依存性的概念，也就是一个目标文件依存于一个特定的程式，而一个可执行文件依存于某些目标文件及程序库．用户只要将这些依存关系与处理方式放入一个反应文件中．Make 即会自动处理它们．

因此如果任何程序文件的修改日期比它相依的目标文件晚，它就依照用户指定的处理命令重新编译；相同的，如果目标文件或程序库的产生日期比它们产生的可执行文件晚，也一样要依指定命令重新连结．本章前面所述的例子要写成如下的 MAKE 文件:

```
accounts.obj: accounts.prg
        clipper accounts -m
screen.obj: screen.prg
        clipper screen -m
report.obj: report.prg
        clipper report -m
accts.exe: accounts.obj screen.obj report.obj
```

第一行指示 accounts.obj 依存于 accounts.prg，若 PRG 文件比 OBJ 文件还新，则执行

clipper accounts−m

下面二个命令指示 screen.prg 与 report.prg 的依存关系与产生方式完全与

accounts.prg 相同。

最后的命令指定 acct.exe 依存于 accounts.obj，screen.obj 及 reprot.obj。若任何一个 OBJ 文件比 acct.exe 新，则执行

link accounts screen report，acct，＼clipper＼clipper＼clipper＼extend。

建议你花点时间，详读 Nantucket3 公司有关 MAKE 的使用手册，你会发现那是非常值得的。

1.7 Clipper 套装软件

和其它语言所提供的精密模块一样，Clipper 也提供了一套完整且丰富的数据库处理工具。而本书及 Clipper 的目的就在协助你写你所需要的应用程序。

下面是 Nantucket Wummer'87 版的 Clipper 软盘中所提供的工具。

第一版软盘是 Clipper 的重心，它包含了编译器 Clipper.exe；连结器，Plink86.exe；还有调试器，debug.obj；包含 Clipper 特殊功能的程序库，extend.lib；覆盖处理程序库，overlay.lib；及一个介绍各软盘内容及特殊功能的文件，READ-ME.1ST。

软盘 2 的大部分空间是被每个应用程序都需要用到的程序库 Clipper.lib 所占据，另外还有两个非常重要的工具。一是 me.prg，它提供一个功能类似 Wordstar 编辑器的原始程序。另一个是允许用户使用 dBASE 索引的工具，ndx.obj。

软盘 3 内包含了 3 个功能很强的工具:

1. DBU，全功能数据库处理器；2. RL，报表及标识编辑器；3. SWITCH，可执行文件连结器。

软盘 4 内包含 Addendum.doc—详列使用手册没有完全介绍的新功能；及许多展示 Clipper 程序技巧的有用程序，副程序与函数。

1.8 安装 CLIPPER

当您拿到原版 CLIPPER 磁盘，绝不可直接用在编译或连结工作。首先，你应该将 CLIPPER 拷贝到硬盘中，接着将它拷贝到一片软盘上做为备份。的后；请在根目录下建立一个子目录以备存放 CLIPPER 文件，例如：下面这个指令将建立一个叫"CLIPPER"的子目录

C＞MD CLIPPER

一旦子目录建立好之后，请依照下列指令进入该子目录

C＞CD CLIPPER

现在请将 CLIPPER 系统磁盘插入 A 驱中，并请键入"A:"以转换到 A 驱上，接着请键入下列指令:

A＞CLIPCOPY(＜drive＞)

上面的指令中的(drive)是磁盘的识别字。若你拿到的 CLIPPER 磁盘，没有 CLIPCOPY.BAT 的文件，则将 CLIPPER 系统磁盘插入 A 驱中，接着请键入下列指