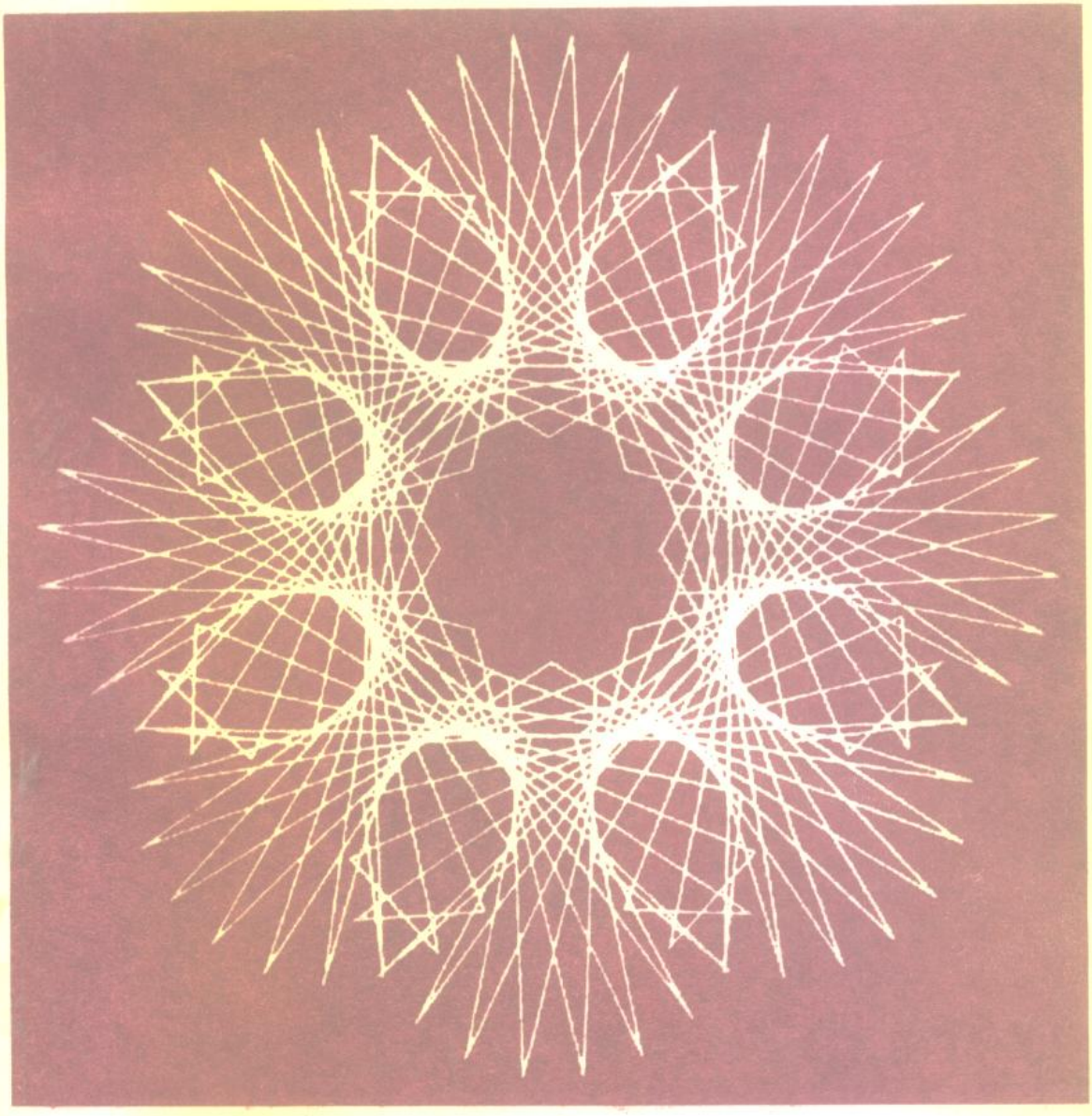
Turbo系列
实用工具软件
用法详解

●尤晓东　潘旭燕　编　●亦　鸥　审校



電子工業出版社

Turbo 系列实用工具软件
用 法 详 解

尤晓东　潘旭燕　编
亦　鸥　审校

电子工业出版社

(京) 新登字 055 号

内 容 简 介

本书全面详细地介绍了与广泛使用的 Turbo C、Turbo Pascal、Turbo [illegible]g、Turbo Assembler、Turbo BASIC、Turbo C++以及 Borland C++等 [illegible]o 系列程序设计语言配套的二十个实用工具程序的功能与用法，使用本书，能够大大方便程序的开发工作，提高编程的效率，加快我国软件开发的质量和速度。

本书可供从事计算机软件开发工作的程序设计人员使用。

Turbo 系列实用工具软件用法详解

尤晓东　潘旭燕　编

亦　鸥　审校

责任编辑　崔国荣

*

电子工业出版社出版

电子工业出版社发行　各地新华书店经售

北京顺义李史山胶印厂印刷

*

开本：787×1092 毫米 1/16　印张：8.5　字数：203.8 千字

1993 年 3 月第 1 版　1993 年 3 月第 1 次印刷

印数:8000　册　定价:6.50 元

ISBN 7-5053-1893-4/TP·453

前 言

自 PC 机问世以来，微机软件在微机市场上占据越来越大的比重，而程序设计语言对微机应用软件的产生是不可缺少的。

美国的 Borland 公司是一家著名的软件公司，它已陆续推出了一序列广受欢迎的微机程序设计语言。在 1991 年以前，它的语言都是以 Turbo 开头来命名的，因此，习惯上被称为 Turbo 系列语言。自 1991 年 2 月推出了 Borland C++之后，这种情况才开始改变。

在我国，广泛使用的 Turbo C、Turbo Pascal、Turbo Prolog、Turbo Assembler、Turbo Debugger、Turbo BASIC、Turbo C++以及 Borland C++都是 Borland C++公司的产品。虽然这些程序设计语言的使用面已经很广，但广大程序设计人员并非已经全面掌握了这些语言的功能和特点。特别是与这些语言配套的还有一系列实用的工具程序，如果使用得当，将大大方便程序的开发工作，提高编程的效率。遗憾的是，大部分程序开发人员只注意到语言本身的使用，对大量的工具程序很少使用，即使使用也只是使用其中的很少的几个（如 MAKE、TLINK）。这一方面是程序员本身没有培养使用工具程序的习惯；另一方面是目前我们能读到的资料对这些工具程序的介绍太少，许多人甚至都不知道与语言配套的还有大量的工具软件，即使知道有工具软件可用，但怎么用又找不到可以参考的资料。

鉴于这种情况，为了向我国的程序开发人员介绍这些工具软件，使程序开发人员能全面、系统地了解 Turbo 系列工具软件的功能和用法，尽快开始使用或更好地使用这些实用工具软件，加快我国软件开发的质量和速度，我们根据有关软件的文档、资料、联机帮助，编著了这本《Turbo 系列实用工具软件用法详解》，希望能对我国的计算机软件事业作出一点贡献。

编 者

一九九二年八月于北京

目 录

概　论

Borland 公司推出的 Turbo 系列语言 Turbo BASIC、Turbo C、Turbo Pascal、Turbo Prolog、Turbo Assembler、Turbo Debugger、Turbo C++和 Borland C++是一系列功能强大、速度很高的程序设计语言。在这些语言软件包中，还配套提供了一系列实用工具软件。使用它们，可以大大方便程序的开发工作，提高编程的效率。本书以某个版本的实用工具为蓝本，介绍这些工具软件的功能和用法。其他语言软件包和其他版本软件包中的同名实用程序的功能和用法，与本书介绍的可能略有差别，读者可以参考本书，具体用法可以查阅与软件配套的手册。下面是这些工具程序的功能简介及我们所依据的该实用程序的版本。

(1) BGIOBJ：图形驱动程序和字模的转换工具程序（2.0 版）。

(2) BINOBJ：将其它格式文件转换为.OBJ 文件（6.0 版）。

(3) CPP：C 语言程序预处理程序（2.0 版）。

(4) GREP：文件搜索工具（3.0 版）。

(5) IMPDEF：为动态连接库（DLL）建立一个模块定义文件。

(6) IMPLIB：为动态连接库（DLL）建立一个输入库（import library）。

(7) MAKE：独立的程序管理器。该工具程序能帮助程序员维护当前版本上的程序，保证它总是最新的（3.5 版）。

(8) OBJXREF：目标模块交叉参考器（3.0 版）。

(9) PRJ2MAK：将 Borland C++的工程文件转换为 MAKE 文件（2.0 版）。

(10) PRJCFG：从配置文件中修改一个工程文件中的选项，或将一个工程文件转换为一个配置文件（2.0 版）。

(11) PRJCNVT：将 Turbo C 的工程文件转换为 Borland C++的格式（2.0 版）。

(12) TCREF：源模块交叉引用实用程序（2.0 版）。

(13) THELP：Turbo Help 联机帮助工具程序（2.0 版）。

(14) TLIB：Turbo 库管理程序（3.01 版）。

(15) TLINK：Turbo 连接程序（4.0 版）。

(16) TOUCH：改变文件的日期和时间的工具（3.0 版）。

(17) TPUMOVER：Turbo Pascal 的单元管理程序（6.0 版）。

(18) TRANCOPY：从一个工程向另一个工程拷贝传送项（2.0 版）。

(19) TRIGRAPH：字符转换工具（1.0 版）。

另外，还有一个功能强大的可用于处理编辑器的宏语言，它称为 TEML（2.0 版）。

本书将详细地介绍每个工具程序，对每个工具程序都给出使用的例子，包括代码和命令行，还说明如何使用它们。

第一章 BGIOBJ

可以使用 BGIOBJ 来将图形驱动程序文件和字符集（笔划式字模文件）转换为目标(.OBJ) 文件。在将它们转换后，就可以将它们连接到程序中，使它们成为可执行文件的一部分。这是对图形软件包动态装入方法的补充，在动态装入方法中，是在运行时从磁盘上装入图形驱动程序和字符集（笔划式字模）的。

将驱动程序和字模直接连接到程序的优点是很明显的，因为这样可执行文件就包含了所有（或大部分）需要的驱动程序和字模，而不必在运行时再访问磁盘上的驱动程序和字模文件了。但是，将驱动程序和字模连接到可执行文件中也将增加该文件的长度。

要将驱动程序或字模文件转换为一个可连接的目标文件，可以以如下语法使用实用工具程序 BGIOBJ.EXE:

BGIOBJ source_file

这里，source_file 是要转换为目标文件的驱动程序或字模文件。生成的目标文件具有与源文件相同的名字，但扩展名为.OBJ。例如，EGAVGA.BGI 将生成 EGAVGA.OBJ，SANS.CHR 生成 SANS.OBJ，依此类推。

1.1 将新的.OBJ 文件加到 GRAPHICS.LIB 中

可以将驱动程序和字模目标模块添加到 GRAPHICS.LIB 中，以使连接器在连接图形例程时能找到它们。如果不将这些新的目标模块添加到 GRAPHICS.LIB 中，就要在工程(.PRJ) 文件、BCC 命令行或在 TLINK 命令行中将它们添加到文件表中。要将这些目标模块添加到 GRAPHICS.LIB 中，可以以如下命令使用 TLIB:

tlib graphics +object_file_name [+object_file_name ...]

这里，object_file_name 是由 BGIOBJ.EXE 建立的目标文件名（如 CGA、EGAVGA、GOTH，等等）。扩展名.OBJ 是隐含的，因此不必写出。可以在一个命令行中写上几个文件，以节省时间。详见下节的例子。

1.2 登录驱动程序和字模

在将驱动程序和字模目标模块添加到 GRAPHICS.LIB 后，就要登录需要连接的所有驱动程序和字模，这可以通过在程序中调用 registerbgidriver 和 registerbgifont 实现，它们通知图形系统已经给出了所需的文件，以保证连接程序在建立可执行文件时能将它们连接到程序中。

每个登录例程带一个参数，这是一个在 graphics.h 中定义的符号名。如果成功地登录了驱动程序或字模，登录程序就返回一个非负值。

下面列出了 registerbgidriver 和 registerbgifont 使用的名字，它是 Borland C++中包

含的所有驱动程序和字模的完整列表。

驱动程序文件 (*.BGI)	registerbgidriver 符号名	字模文件 (*.CHR)	registerbgifont 符号名
CGA	CGA_driver	TRIP	triplex_font
EGAVGA	EGAVGA_driver	LITT	small_font
HERC	Herc_driver	SANS	sansserif_font
ATT	ATT_driver	GOTH	gothic_font
PC3270	PC3270_driver		
IBM8514	IBM8514_driver		

1.3 例　子

假设要登录 CGA 图形驱动程序、gothic 字模和 triplex 字模时，将文件转换为目标模块，然后将它们连接到程序中。下面是执行的步骤:

(1) 用 BGIOBJ.EXE 将二进制文件转换为目标文件，使用以下命令行:

```
bgiobj  cga
bgiobj  trip
bgiobj  goth
```

这将建立三个文件: CGA.OBJ、TRIP.OBJ 和 GOTH.OBJ。

(2) 可以用以下 TLIB 命令行将这些目标文件添加到 GRAPHICS.LIB 中:

```
tlib  graphics  +cga  +trip  +goth
```

(3) 如果没有将这些目标文件添加到 GRAPHICS.LIB 中，就需要将目标文件名 CGA.OBJ、TRIP.OBJ 和 GOTH.OBJ 加到工程列表中（如果使用 Borland C++的集成环境)，或者加到 BCC 的命令行中。例如，BCC 的命令行应象下面这样调用:

```
BCC  niftgraf  graphics.lib  cga.obj  trip.obj  goth.obj
```

(4) 在图形程序中以下面的方法登录这些文件（注意: 如果在连接某些驱动程序或字模时出现连接错误“Segment exceeds 64K-段超过 64K”，就参见下一节):

```
/* 声明 CGA_driver, triplex_font & gothic_font 的头文件 */
#include <graphics.h>

/* 登录并检测错误 */

if (registerbgidriver (CGA_driver) < 0) exit (1);
if (registerbgifont (triplex_font) < 0) exit (1);
if (registerbgifont (gothic_font) < 0) exit (1);
```

```
/* ...* /

initgraph (....);                          /* initgraph 应在 registering 之后调用* /

/* ...* /
```

1.4 / F 选项

本节说明在将几个驱动程序和字模文件连接到程序中后（特别是在 samll 和 compact 模式下），在出现“Segment exceeds 64K（段超过 64K）”或相似的错误时，应该怎么处理。

默认情况下，由 BGIOBJ.EXE 建立的文件都使用同一个段（称为__TEXT）。但是如果程序中连接了许多驱动程序和字模，或者在使用 small 或 compact 模式时，就会出问题。

为了解决这个问题，可以用 BGIOBJ 的/ F 选项来转换一个或多个驱动程序和字模。该选项使 BGIOBJ 使用一个名为“文件名__TEXT”的段，这样，默认段就不会对连接的所有驱动程序和字模（在 small 和 compact 模式的程序中为所有的程序代码）负担太重。例如，下面的两个 BGIOBJ 命令行让 BGIOBJ 使用名为 EGAVGA__TEXT 和 SANS__TEXT 的段。

```
bgiobj  / F  egavga
bgiobj  / F  sans
```

当选择/ F 选项时，BGIOBJ 还将 F 拼到结果目标文件名中（即 EGAVGAF.OBJ、SANSF.OBJ，等等），将__far 拼加到要被 registerfarbgidriver 和 registerfarbgifont 使用的名中（例如，EGAVGA__driver 变成了 EGAVGA__driver__far）。

对用/ F 建立的文件，必须使用远程登录例程，而不能使用常规的 registerbgidriver 和 registerbgifont。例如：

```
if (registerfarbgidriver (EGAVGA__driver__far) < 0) exit (1);
if (registerfarbgifont (sansserif__font__far) < 0) exit (1);
```

1.5 高级特征

本节讨论 BGIOBJ 的一些高级特征和 registerfarbgidriver 和 registerfarbgifont 例程。这一节是专门为熟练的用户准备的。

下面是 BGIOBJ.EXE 命令行的完整语法：

```
BGIOBJ  [/ F] source destination public—name seg—name seg—class
```

下表说明 BGIOBJ 命令行的每个组成部分。

组成部分	说　明
/F 或 -F	本选项让 BGIOBJ.EXE 使用另一个段名，而不是使用__TEXT（默认段名），并改变全局名和目标文件名。参见前一节的说明
<source>	这是要转换的驱动程序和字模文件。如果文件不是 Borland 公司提供的驱动程序或字模文件，则需要指定完整的文件名（包括扩展名）
<destination>	这是生成的目标文件名。默认的目标文件名为 source.OBJ，或 sourceF.OBJ（如果使用了/F 选项）
public-name	这是程序中在调用 registerbgidriver 或 registerbgifont （或对应的远程版本）来连接目标模块时将使用的名字。 公用名是被连接器使用的外部名，因此它应该是在程序中使用的名字前缀以下划线。如果程序使用了 Pascal 调用约定，就只能使用大写字母，并且不再加上下划线
seg-name	这是一个可选的段名；默认段为__TEXT（如果指定了/F 选项，则为"文件名__TEXT"）
seg-class	这是一个可选的段类；默认值为 CODE

除了 source 外的所有参数都是可选的。但是，如果要指定一个可选的参数，就必须指定它前面的所有参数。

如果选择使用自己的公用名，就要向程序中加上声明，形式如下：

```
void public__name(void);                    /*如果没有使用/F，就使用默认的段名*/
extern int far public__name[];              /*如果使用了/F，或使用了不是__TEXT 的段*/
```

在这些声明中，public__name 匹配在 BGIOBJ 转换时用的公用名。头文件 graphics.h 中包含了默认的驱动程序和字模的公用名的声明；如果使用这些默认公用名，就不必如刚才所说的声明它们了。

在这些声明后，需要在程序中登录所有的驱动程序和字模。如果没有使用/F 选项，没有改变默认的段名，就应用 registerbgidriver 和 registerbgifont 来登录驱动程序和字模；否则，就使用 registerfarbgidriver 和 registerfarbgifont。

下面是一个将字模文件装入到内存中的例子程序。

```
/*将字模文件装入到内存中的例子*/

#include <graphics.h>
#include <io.h>
#include <fcntl.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <process.h>
#include <alloc.h>

main()
{
```

```
void        *gothic_fontp;                    /*指向内存中的字模缓冲区*/
int         handle;                           /*用于输入输出的文件句柄*/
unsigned    fsize;                              /*文件和缓冲区的大小*/

int errorcode;
int graphdriver;
int graphmode;

/*打开字模文件*/
handle = open ("GOTH.CHR", O_RDONLY|O_BINARY);
if (handle == -1)
{
    printf ("unable to open font file 'GOTH.CHR'\n");
    exit (1);
}
/*找出文件的大小*/
fsize = filelength (handle);
/*分配缓冲区*/
gothic_fontp = malloc (fsize);
if (gothic_fontp == NULL)
{
  printf ("unable to allocate memory for font file 'GOTH.CHR'\n");
  exit (1);
}
/*将字模读入内存*/
if (read (handle, gothic_fontp, fsize) != fsize)
{
    printf ("unable to read font file 'GOTH.CHR'\n");
    exit (1);
}
/*关闭字模文件*/
close (handle);
/*登录字模*/
if (registerfarbgifont (gothic_fontp) != GOTHIC_FONT)
{
    printf ("unable to register font file 'GOTH.CHR'\n");
    exit (1);
}
/*检测和初始化图形系统*/
graphdriver = DETECT;
initgraph (&graphdriver, &graphmode, "..");
errorcode = graphresult();
if (errorcode != grOk)
{
    printf ("graphics error: %s\n", grapherrormsg (errorcode));
```

```
        exit (1);
    }
    settextjustify (CENTER_TEXT, CENTER_TEXT);
    settextstyle (GOTHIC_FONT, HORIZ_DIR, 4);
    outtextxy (getmaxx() /2, getmaxy() /2, "Borland Graphics Interface (BGI) ");

    /*按任一键结束*/
    getch();
    /*关闭图形系统*/
    closegraph();
    return (0);
}
```

第二章　BINOBJ

在 Turbo Pascal 中，有一个名叫 BINOBJ.EXE 的实用工具程序，它的作用与上一章介绍的 BGIOBJ 相似。它可以将其他文件转换为.OBJ 文件，以使该文件能作为一个“过程”连接到 Turbo Pascal 程序中。如果有一个必须驻留在代码段的二进制数据文件，或者有一个太大的无法放到类型常量数组的二进制数据文件时，这是非常有用的。例如，可以将 BINOBJ 用于 Graph 单元，以将图形驱动程序或字模文件直接连入.EXE 文件中。然后，在使用图形程序时，就只须有.EXE 文件即可。

BINOBJ 带三个参数，如下所示：

BINOBJ <source[.BIN]> <destination[.OBJ]> <public name>

这里，source 是要转换的二进制文件名，destination 是产生的.OBJ 文件名，而“public name”是在 Turbo Pascal 程序中被声明的过程的名字。

在下面的例子中，过程 ShowScreen 带一个指针参数，它将 4000 个字节的数据移到屏幕内存中。名为 MENU.DTA 的文件包含主菜单屏幕的映像（80×25×2 = 4000 字节）。

以下是 MYPROG.PAS 的一个简单版本（无错误检测功能）：

```
program MyProg;

procedure ShowScreen (var ScreenData : Pointer);
{ 显示全屏数据，但无错误检测！ }
var
  ScreenSegment: Word;

begin
  if (Lo (LastMode) = 7) then                          { 单色？ }
    ScreenSegment := $B000
  else
    ScreenSegment := $B800;
  Move (@ ^ScreenData ^ ,                              { 从指针 }
    Ptr (ScreenSegment, 0) ^ ,                         { 移到视频内存 }
    4000);                                             { 80 × 25 × 2 }
end;

var
  MenuP : Pointer;
  MenuF : file;
begin
  Assign (MenuF, 'MENU.DTA');                          { 打开屏幕数据文件 }
```