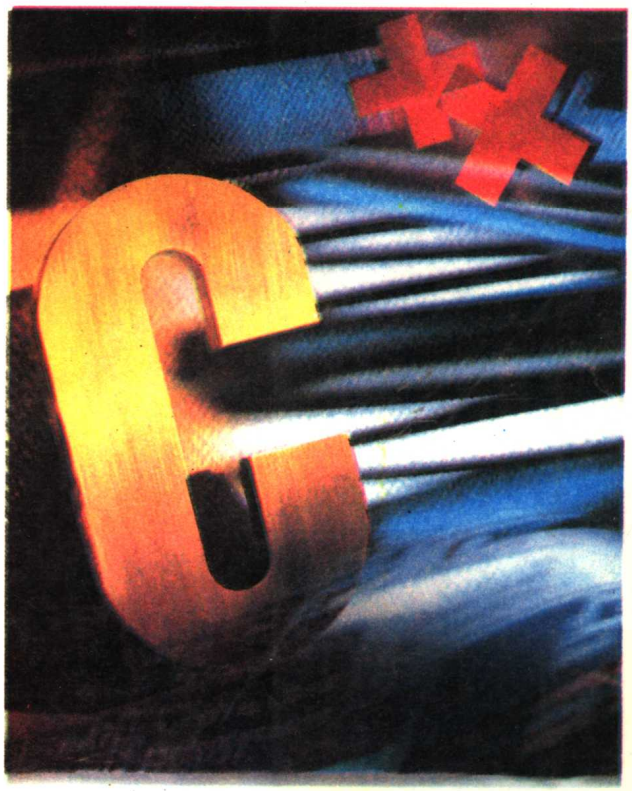
北京希望电脑公司

6

BORLAND® C++ 3.0
Borland C++ 3.0
工具和实用程序指南

张克华 编译
希 望 审校



海洋出版社

北京希望电脑公司 Borland C++ 3.0 系列丛书之六

Borland C++ 3.0

工具和实用程序指南

张克华 编译

希 望 审校

海洋出版社

1992 年 · 北京

内 容 简 介

本书完整地介绍了 Borland C++ 3.0 的实用程序。首先介绍了引入库工具 IMPDEF、IMPLIB 和 IMPLIBW 的功能和用法，然后介绍了程序管理程序 MAKE、库管理程序 TLIB 和连接器 TLINK 的功能和用法，说明了 WinSight、RC 和 HC 等 Windows 应用程序开发工具的功能和用法，在附录的前面部分列出了 Borland C++ 的错误信息，最后列出了几个没有文档的实用程序的功能和用法。

本书是 Borland C++ 3.0 系列丛书之一，需要本书和整套丛书者请与北京 8721 信箱联系，邮编 100080，电话 2562329。

(京)新登字 087 号

责任编辑：阎世尊

Borland C++ 3.0
工具和实用程序指南

张克华　等编译
希　望　审校

海洋出版社(北京市复兴门外大街 1 号)
海洋出版社发行　双青印刷厂印刷
开本：787x1092 1/16　印张：11.625　字数：　283 千字
1992 年 1 月第一版　1992 年 1 月第一次印刷
印数：1-3000
ISBN 7-5027-2609-8/TP・90　定价：11.00 元

郑 重 说 明

Borland(宝兰国际有限公司)授权我中科院九州计算机网络公司为 Borland 程序语言和应用软件之中国代理。

北京希望公司此中文手册的出版，已通过我公司与 Borland 的商讨，得到 Borland 的认可。

该手册仅限于和相应的 Borland 软件配套发行，严禁私自翻印和单独发行。

中科院九州计算机网络公司

1992 年 9 月

地　址：北京海淀白石桥路25号

邮　编：100081

电　话：8311822　8420320

特此函告

中国科学院计算中心

九州计算机网络公司

1992.9.19

目　录

第0章 简 介

Borland C++带有许多功能强大、独立的实用程序，使用户能使用 Borland C++文件和其它的实用程序来减轻 DOS 和 Windows 的程序设计负担。

本书描述了 IMPDEF、IMPLIB、IMPLIBW、MAKE、TLIB、TLINK 和 WinSight，并用代码和命令行的例子说明怎样使用它们。Borland C++实用程序的其它部分放在名为 UTIL.DOC 的文本文件中，它是被放置在 DOC 子目录中的文挡程序。

名字	描述
本书中涉及的程序	
IMPDEF	产生模块定义文件。
IMPLIB	产生引入库。
IMPLIBW	产生一个运用于Windows的引入库。
MAKE	独立的程序管理器。
TLIB	Turbo库管理程序。
TLINK	Turbo链接器。
WinSight	Windows消息监示器。
在联机UTIL.DOC中涉及的程序(在附录B~附录D中介绍)	
BGIOBJ	图形驱动程序和字体文件的转换实用程序。
CPP	预处理器。
GREP	文件搜索实用程序。
OBJXREF	目标文件的交叉参考。
PRJCFG	根据配置文件把映象文件中的选择变为当前的选择，或改变一个映象文件为配置文件。
PRJCNVT	把Turbo C映象文件变成Borland C++格式。
PRJZMAK	把Borland C++映象文件变成MAKE文件。
THELP	Turbo求助功能。
TOUCH	更新文件日期和时间。
TRANCOPY	复制一个映象文件中的项到另一文件中。
TRIGRAPH	字符变换实用程序。

第一章　输入库工具

动态连接库(DLLs)是 Windows 程序设计的重要部分。用户能创建自己通常要使用的代码的 DLLs，能在实际应用程序(自己的程序代码)或来源于别处而用户需要的代码中使用 DLLs。

在创建 Windows 应用程序时，TLINK 使用引入(IMPORT)库，在连接时，TLINK 需要知道函数是什么时间被定义和什么时间从 DLL 中引用的。引入库代替了通常的模块定义文件(.DEF)。

IMPDEF 为在 DLL 中的每一个可被输出的函数创建一个包含 EXPORT 语句的模块定义文件。IMPLIB 为 DLL 创建一个引入(IMPORT)库。IMPLIBW 为 DLL 创建一个引入库，但后者一个 Windows 应用程序。

1.1　IMPDEF：模块定义管理程序

一个输入库提供了对 Windows DLL 中函数的访问。

IMPDEF 是配合 IMPLIB 而工作的，它使用户定制一个引入(IMPORT)库，从而能适合于特别应用程序的要求。其语法是：

```
IMPDEF DestName.DEF SourcName.DLL
```

语句从文件 SourceName.DLL 中产生了一个名为 SourceName 的模块定义文件。模块定义文件将是这样的：

```
LIBRARY    FileName
DESCRIPTION 'Deacription'
EXPORTS
        ExportfuncName          @Ordinal
        ExportfuncName          @Ordinal
```

这里 FileName 是 DLL 的根文件名，如果 DLL 通过包含一个 DESCRIPTION 语句的模块定义文件来连接，则 Description 就是 DESCRIPTION 语句的值，ExportFuncName 代表了一个可输出函数的名字，Ordinal 是那个函数的序数值(一个整数)。

1.1.1　一个 DLL 中的类

IMPDEF 对使用 C++类的 DLL 是特别方便的。其原因如下：其一，如果当用户定义一个类时使用了关键字_export，则该类中所有非内部函数成员和静态数据成员都被输出。由于 IMPDEF 列出了所有的可输出的函数，自动地包括了函数成员和静态数据成员，所以让实用程序为用户建立一个模块定义文件是容易的。

其二。由于这些函数的名字可以是同名的，所以在一个模块定义文件的 EXPORTS 部分列出了它们，并从模块定义文件创建输入库，这将是很麻烦的工作，但是如果用户使用

IMPDEF 去创建模块定义文件，它将为每个可被输出函数分配一个序数值，如果被输出的函数名字具有相同的名字，IMPDEF 将包括一个原来的没有二义性的函数名作为函数项的注释跟在后面。例如：IMPDEF 为使用 C++类的 DLL 而产生的模块定义文件将是如下的形式：

```
LIBRARY      FileName
DESCRIPTION  'Description'
EXPORTS
                    MangledExportFuncName  @Ordinal  ;  ExportFuncName
                    MangledExportFuncName  @Ordinal  ;  ExportFuncName
```

其中，FileName 是 DLL 的根文件名，如果 DLL 连接时包含一个模块定义文件，且该模块定义文件包含一个 DESCRIPTION 语句，则 Description 是 DESCRIPTION 语句的值，MangledExportFuncName 提供了同名的名字，Ordinal 是函数的序数值(一整数)，而 ExportFuncName 给出了函数的原始名字。

1.1.2 一个 DLL 中的函数

IMPDEF 建立了一个可编辑的源文件，在源文件中列出了所有在 DLL 中的可被输出的函数，可以先编辑这个.DEF 文件使它仅包含用户应用程序中所需要加以引用的函数，再在编辑后的.DEF 文件上运行 IMPLIB。运行的结果创建一个引入库，该引入库包含某一 DLL 所包含的输出函数的特定子集的引入信息。

例如，假设用户正在使用研制某一 DLL，该 DLL 提供几个应用程序所要使用的函数；并假设每一个 DLL 中的输出函数都已用_export 定义。如果所有的应用程序要用到所有的函数，那么就可以使用 IMPLIB 简单地创建一个 DLL 的引入库，并把该引入库和 DLL 一起使用，创建的引入库包含了所有 DLL 可输出函数的引入信息。引入库能连接到任何的应用程序中去，这样为了使用在 DLL 中的可输出函数，用户就可不必在模块定义文件的 IMPORTS 部分列出所有要使用的在 DLL 中可输出函数名，为程序设计工作减轻负担。

现在，假设用户所编写的程序只想使用所有 DLL 可输出函数中一部分函数。在理想的状态下，用户最好将程序与一个经过定制的引入库相连接，该引入库只提供用户应用程序所需要函数的引入信息，显然要引入的函数必须是 DLL 所能提供函数的子集。所有在 DLL 中其它的输出函数对用户应用程序来说是透明的或者说不可见的。

要使创建 IMPORT 库满足这些情况，必须将经过编译和链接的 DLL 作为操作对象运行 IMPDEF，IMPDEF 创建一个模块定义文件，该模块定义文件包含一个 EXPORT 部分，列出全部 DLL 可输出函数。用户可以编辑那个模块定义文件，在 EXPORTS 部分删除在用户的引入库中不想要的函数项。一旦用户将所不想要的输出函数删除后，在模块定义文件上运行 IMPLIB。运行的结果将创建一个 IMPORT 库，它仅包含用户所需要的，在模块定义文件的 EXPORTS 部分列出的输出函数的输入信息。

1.2 IMPLIB：引入库管理程序

IMPLIB 实用程序建立一个引入(IMPORT)库，引入库包含的函数将不再在 Windows 应

用程序的 IMPORTS 部分列出，这样在 Windows 应用程序的模块定义文件的 IMPORTS 部分就可以减少列出的项，甚至可以全部省略。

如果一个模块使用 DLLs 中的函数，则用户有两种方式通知链接器：

■ 在模块定义文件中增加一个 IMPORTS 部分，并在其中列出那个模块将使用的 DLLs 中的每个函数。

■ 或者在链接模块时，包含为 DLLs 建立的 IMPORT 库。

因为 TLINK 的缺省设置满足许多应用程序，模块定义文件通常是不必使用的。

如果在创建一个 Windows 应用程序，则至少将使用一个引入库即 IMPORT.LIB。IMPORT.LIB 是标准 Windows DLLs 的引入库。(当用户在 IDE 下建立一个 Window 应用程序，并使用 BCC 连接时，IMPORT.LIB 被自动连接。如果用户单独使用 TLINK 连接时，就需要显式地与 IMPORT.LIB 连接。)

参见关于使用 IMPDEF 和 IMPLIB 建立一个适合特定应用程序的引入库的信息。一个输入库列出了一个或多个 DLL 中的一些或全部的输出函数。IMPLIB 从 DLL 或从为 DLL 建立的模块定义文件中直接生成 IMPORT。

为一个 DLL 生成一个输入库，格式是

IMPLIB Options LibName [DefFiles... | DLLs..]

这里 Options 是一个或多个 IMPLIB 选项的选项列表(见表 1.1)，LibName 是新 IMPORT 库的名字，DefFiles 是为一个或多个 DLLs 建立的一个或多个模块定义文件的列表，DLL 是一个或多个现存的 DLL 列表。用户必须指定一个 DLL 或模块定义文件。

表 1.1 IMPLIB 选项

选项	作用
-i	告诉IMPLIB忽略WEP，因为Windows退出过程需要终止一个DLL。如果你在同一IMPLIB命令行指明多于一个DLL时使用这个选择。 *你可以使用连字号也可以使用破折号在IMPLIB选择符的前面，但选择符必须是小写字符。
警告控制：	
-t	简洁的警告信息
-v	详细的警告信息
-w	无警告信息

1.2.1 重建 IMPORT.LIB

当 Microsoft 公司发行新版本的 Windows 时，你可能需要用新的 IMPORT.LIB 代替旧的。更方便的方式是自己创建 IMPORT.LIB。

这个命令行建立了当前版本的 IMPORT.LIB(在 WindowsSYSTEM 下在同一行输入这条命令)：

```
IMPLIB -i  IMPORT.LIB GDI.EXE KERNEL.EXE USER.EXE KEYBOARD.DRV
           SOUND.DOV WIN87EM.DLL
```

如果 Windows 是扩展的，以便它能使用其它的 DLL，任何一个新的 DLL 将必须出现在同一命令行。

1.3 IMPLIBW：面向 Windows 的 IMPORT 库管理程序

关于输入库及 DLL 的使用详见前面关于 IMPLIB 的讨论。

Windows 实用程序 IMPLIBW 可以建立一个 IMPORT 库来代替为 Windows 应用程序而建立的模块定义文件中的部分和全部输出函数。不象本书讨论的其它工具和实用程序，IMPLIBW 是运行于 Windows 系统下的程序。

1.3.1 选择一个 IMPORT 库

用 IMPLIBW 建立一个 IMPORT 库，从屏幕菜单中用鼠标按键选择进入 IMPLIBW，将会出现一个空的窗口。选择 File | Create...选项，将出现一个对话框显示出当前目录下的所有 DLL。可以用鼠标键选择窗口中的目录名来查看不同目录下的 DLL。

1.3.1.1 从一个 DLL 中选择

当用户选择一个 DLL，点按鼠标键选择 Create(或简单地在一个 DLL 文件名上按鼠标键选择)，IMKPLIBW 将产生一个与被选择的 DLL 相同名的 IMPORT 库，其扩展名为.LIB。

1.3.1.2 从一个模块定义文件中选择

如果用户有为 DLL 建立的模块定义文件，就能使用它来代替 DLL 以产生一个 IMPORT 库。

不用选择 DLL，从编辑控制中直接输入.DEF 文件名，并按键选择 Create 选项。IMPLIBW 将生成与.DEF 文件同名的 IMPORT，其扩展名为.LIB。

1.3.2 建立 IMPORT 库

当 IMPLIBW 检查 DLL 或模块定义文件并创建 INPORT 库没有错误时，窗口显示“No Warning”。这就是所做的所有工作。现在有了一个能包含于设计规划中的 IMPORT 库。

第二章　MAKE：程序管理器

Borland 的命令行 MAKE，源于 UNIX 的同名程序，帮助用户保持当前可执行程序的版本是最新的。许多程序由多个源文件组成，每个文件在与其它文件合并前要经过预处理，汇编，编译，和其它步骤。忘记重新编译已修改过的文件，而修改的东西又是很重要的，则会导致很麻烦的错误。另一方面，重新编译每一文件将是很浪费时间的。

怎样为 Window 应用程序建立一个 makefile 文件见程序员指南的第八章“创建一个 Windows 应用程序”。

MAKE 解决了上述问题，向 MAKE 提供你程序的源文件和目标文件如何经过处理产生最后产品的描述信息。MAKE 将查看这些描述，并在你的文件上标上日期，因为需要建立最新的版本。在这一处理期间，MAKE 可以调用许多不同的编译器，汇编器，链接器，和实用程序，但它永远只做产生最新版本程序所必需的。MAKE 的用途不仅仅是在程序设计时。你能用 MAKE 来控制任何通过用文件名来选择文件并处理它们以产生最后产品的过程。一些通常的使用包括文本处理，自动备份，排序文件到其它目录，和清除目录中的临时文件。

2.1　MAKE 是怎样工作的

MAKE 通过做下面的工作以使你的程序保持最新版本：

- 读入一个名为 makefile 你已建立的特别文件，这个文件告诉 MAKE 哪一个.OBJ 和库文件必须链接以便产生可运行文件，哪个源文件必须与前面的文件一起编译产生.OBJ 文件。
- 检查每个.OBJ 文件的时间和日期相对于它的源文件和头文件的日期与时间，如果他们都比.OBJ 的晚，则 MAKE 认为源文件被修改过，必须重新编译。
- 调用编译器重新编译源文件。
- 一旦所有.OBJ 依赖的文件都检查过了，则检查.OBJ 相对于你的可执行文件的日期和时间。
- 如果.OBJ 文件都比.EXE 文件日期晚，则调用链接程序重新产生.EXE 文件。

原始的 IBM PC 和兼容机不带内部时钟，如果你的系统在这种情况下必须先正确地设置时间与日期。

MAKE 完全依赖 DOS 标记在文件上的时间，这意谓着，为使 MAKE 去做它的工作，你的系统日期和时间必须是正确的，要经常更换系统时钟的电池。电力不足将导致时钟的错误，MAKE 便不能照常工作。

2.2　启动 MAKE

有两种版本的 MAKE：保护方式的版本 MAKE.EXE，和实时方式的版本 MAKER.EXE。他们工作是一样的。唯一的不同是保护方式能处理较大的 makefile 文件。我们说 MAKE 时，是指两种版本。

使用 MAKE，在 DOS 提示符下键入 make。MAKE 就查找名为 MAKEFILE 的特定文件。如果 MAKE 不能找到 MAKEFILE，它就寻找 MAKEFINE.MAK；如果又没发现它或 BUILTINS.MAK(后面描述)，则将出现错误信息并退出。

如果你要使用的名字不是 MAKEFILE 或 MAKEFILE.MAK，则给 MAKE 加文件选项(-f)，如下：

```
MAKE -f MKYFILE.MAK
```

MAKE 一般的语法是

```
make [option...] [target...]
```

这儿 option 是 MAKE 选择项，target 是要做的目标文件名。

下面是 MAKE 的语法规则：

■ 词 make 后跟一空格，然后是一系列 make 选项。

■ 每个 make 选择项必须与相邻者用一空格分开。选择项能以任何顺序排列，选择项的数目不限(只要在命令行还有空间)。所有的选项并不指明一个字串是否能有选项-或+跟在它们后面。这取决于你是希望关闭选项(-)或开启(+)。

■ MAKE 选项链后跟一个空白符，然后是目标选项链。

■ 每个目标必须与相邻者用一空格分开。MAKE 以目标项列出的顺序来查看目标文件的生成，按要求来重新编译它们。

如果命令行不包含任何目标名，MAKE 使用在显式规则构成命令行里提到的第一个目标文件。如果命令行中有一个或多个目标，则他们将按需要被处理。

2.2.1 命令行选项

下面是完整的 MAKE 的命令行选项列表。注意不同情况下大小写的意义，例如，选择-d 不能代替-D。注意在选择项前可以使用'-'也可以使用'/'。

表 2.1: MAKE 选项

选项	作用
-？或-h	输出帮助信息。缺省选项后跟加号'+'。
-a	对.OBJ文件进行依赖性检查。
-B	构造全部目标而忽略文件日期。
-d directory	在使用-S选项时，告诉MAKE在特定的子目录里输出它的文件。directory包括驱动器号。对保护模式的MAKE版本无作用。
-D identifier	给由一个字符组成的字串定义名字标识符。
[-D] iden=string	定义iden为等号后的字串的标识名。如果字串包含空格和制表符，则必须用引号括起。选项-D可写也可不写。