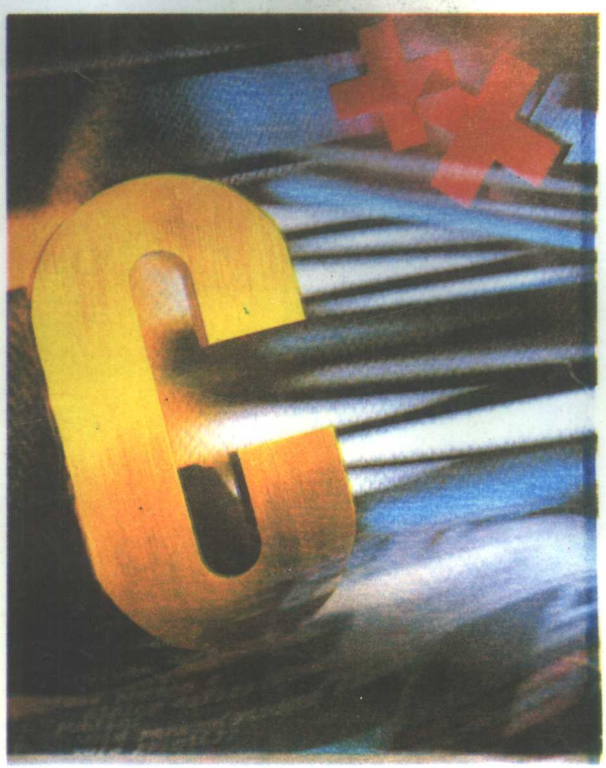
北京希望
电脑公司

10

BORLAND® C++3.0
Turbo Profiler 2.0
用 户 手 册

莫树人 编译
希 望 审校



-62

海洋出版社

北京希望电脑公司 Borland C++ 3.0 系列丛书之十

Turbo Profiler 2.0
用户手册

莫树人　编译
希　望　审校

10

海洋出版社
1992 年 · 北京

内容简介

Turbo Profiler 支持剖视和优化 Borland 公司开发的 Turbo Pascal、3.0 平台上 C++语言和汇编语言。

本书全面系统地介绍了 Turbo Profiler 的功能和用法。首先用不同的例子说明了 Turbo Profiler 的基本功能和操作；然后介绍了 Turbo Profiler 的集成环境的组成和菜单参考；接着讲述了用 Turbo Profiler 进行剖视的方法以及如何利用剖视的结果，说明 Turbo Profiler 的工作原理；最后，在附录中列出了 Turbo Profiler 的命令行选项，介绍了修改配置的方法，讲述了如何进行远程剖视的方法，如何使用 Windows 下的 Turbo Profiler，如果在 80386 机器上进行虚拟剖视，还列出了 Turbo Profiler 的提示和出错信息。

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

Turbo Profiler 2.0
用户手册

莫树人　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：9.625　字数：234 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：8.0 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

此为试读，需要完整PDF请访问：www.ertongbook.com

绪　论

Turbo Profiler 是用户软件开发周期中不可缺少的一个重要环节。它可以使你的程序运行得更快并且更富有效率，从而达到既定目标。

Turbo Profiler 是一个性能分析器，它是一种软件工具，通过分析下列情况来衡量用户程序的性能：

- 用户程序的每一部分各花费多少时间
- 某一行程序执行了多少次
- 已经执行过多少行程序
- 某一个例程被调用了多少次，是由哪些例程调用的
- 用户程序经常访问哪些文件，以及为此花费了多少时间

Turbo Profiler 同时还监视下列计算机资源：

- 处理器时间
- 磁盘读写操作
- 键盘输入
- 打印机输出
- 各种中断

Turbo Profiler 通过监视程序运行的主要情况并且提供关于程序各部分的详细的性能统计报告，能使对户对自己的程序进行“微调”(Fine-tune)。它可以剖析到用户程序的深层并展示其最复杂的操作——从执行次数到语句计数，从中断调用到文件访问活动——从而帮助用户优化程序代码和加快程序的运行速度。

Turbo Profiler 在功能和使用的简便性方面胜过市面上销售的其它剖析器，这主要表现在它提供了下列功能：

- 交互式的剖析迅速找出程序中效率不高的代码。
- 用户在剖析期间可以读取或编辑任何文本文件。
- 可以剖析在 DOS 或 Windows 下运行的任意大小的程序。
- 可以处理使用 Turbo Pascal,Boland 的任意一种 C＋＋编译器和 Turbo Assembler 生成的程序。
- 提供了一个具有多窗口覆盖，鼠标支持和上下文相关式帮助的易用接口。
- 报告程序的各个例程和单行程序的执行时间和执行次数。
- 跟踪展示哪些代码块被执行过，而哪些没有被执行。
- 对所有例程作完整的调用路径跟踪。使用完整的调用栈跟踪来分析调用的频率。
- 在 Files 窗口中通过文件句柄及文件打开、关闭、读或写的时间来监视 DOS 文件的活动。使用一个事件表来记录读或写的字节数。
- 支持有选择的中断监视。监视所有视频、键盘、磁盘、鼠标和用户中断。提供一个完整的事件表或频率监视。通常的符号名形式的 DOS 调用表可以使用户

迅速识别各类中断。

- 支持对 Turbo Pascal 和 Turbo C++ 覆盖程序段的完整跟踪。
- 386 虚拟剖析占用的 RAM 容量最小，其余可供用户程序使用。
- 支持远程系列和网络剖析。

0.0.1 优化和剖析的差别

优化器通过用执行得较快的指令替换那些耗时的指令可以使程序运行得快些，但是优化器并不能对效率低的代码进行修补。

剖析器能帮助用户发现代码效率最低的部分并指出哪些算法可以修改或重写。研究表明，要使程序性能有较大的提高，修改算法和数据结构比优化小段编译过的代码更为有效。不用剖析器(Profiler)就想找出程序瓶颈如同不用调试器(Debugger)就想找出程序错误一样；Turbo Profiler 能够既省时又省力地帮助用户提高程序性能。

0.0.2 软件和硬件要求

Turbo Profiler 对硬件和软件的要求同 Boland 公司的其它语言产品一样。想得到一张完整的需求表，可以参考你正在使用的 Boland 的任意一种语言的用户指南。

0.1 版本 2.0 的新功能

Turbo Profiler 2.0 具备几个新功能：有效分析、Windows 程序剖析、批处理方式剖析和远程网络剖析。

有效分析主要是找出在程序运行期间没有被执行过的代码部分。有了该功能，用户可以查出程序中从来没有被调用过的部分(死代码)，并且可以确认在适当的时候某段代码是否被正确地调用，同时该功能还可以保证程序检测覆盖了用户代码的所有部分。

在 Turbo Profiler 的 1.1 版本中，当程序在通过一根弱信号调制解调器电缆连接的远程计算机上运行时，用户可以对它进行剖析。2.0 版又使远程剖析迈进了一大步：两个系统之间的联系可以是一个局域网络(LAN)。

Turbo Profiler 现在在各个剖析方式下都完全支持 Windows 程序，包括局部剖析、远程系列剖析和远程网络剖析。

Turbo Profiler 2.0 版允许用户编写 DOS 批处理文件进行自动剖析。在批处理方式下不能使用集成环境；取而代之的是，Turbo Profiler 可以自动运行在批处理文件中指定的文件并且将信息存放在.TFS 文件中。有几个 Turbo Profiler 命令可以在批处理文件中使用，这就允许用户用一个 DOS 命令进行多次剖析。

0.2 安装 Turbo Profiler

要在系统上安装 Turbo Profiler，可以运行用户配发盘上的使用方便的安装程序 INSTALL.EXE，该程序可以自动将文件从配发盘拷贝到硬盘。具体作法是：将带有 INSTALL 程序的磁盘插入 A 驱动器，输入“A：INSTALL”然后按回车键，接着按照屏幕上出现的提示去做就可以了。

0.2.1 README 文件

用户在使用 Turbo Profiler 做其它工作以前，请先看一看 Install 盘上的 README 文件。这个文件可能包含了本手册中没有的最新信息，同时还列出了配发盘上的联机记录文件及关于每个文件的简要说明。

要访问 README 文件，可以在 A 驱动器插入 Install 盘，敲入“A:”和回车转到 A 驱动器，然后输入“README”并按回车键。进入 README 文件后，可以使用↑键和↓键上下查看该文件。按 ESC 键即可退出。

0.3 本手册中包含的主要内容

本章简要描述了剖析器 Turbo Profiler 及其功能，并帮助用户为在系统上运行 Turbo Profiler 做好准备。

第一章　剖析的实例

用一个简单的剖析实例使用户对剖析的全过程有一个大概的了解。整个过程从“看看到底发生了什么”的剖析方式开始，然后向用户解释收集到的剖析数据，使用户在剖析过程中深入了解该程序，在此基础之上再进行修改和优化，最后再进行几次剖析以判断每次修改程序的效果。

第二章　Turbo Profiler 环境

详细解释了 Turbo Profiler 环境中的每个菜单项和对话框选项。

第三章　剖析方法

给出了进行成功的剖析所需的一般性指导和忠告。

第四章　剖析器内情

解释了 Turbo Profiler 在用户程序运行时如何收集关于执行时间及次数的数据。

附录 A　Turbo Profiler 的命令行选项

解释了如何使用 TFINST 改变 TPROF 和 TPROFW 的缺省配置。

附录 B　修改 Turbo Profiler 的配置

列出了 Turbo Profiler 的所有命令行选项并解释了每个选项的功能。

附录 C　远程剖析

描述了如何使用两个系统进行剖析；即用户在两个系统上分别运行 Turbo Profiler 和被剖析程序。

附录 D　适用于 Windows 的 Turbo Profiler

描述了在 Windows 环境下如何运行 Turbo Profiler 及如何使用它的特殊功能。

附录 E　80386 处理器上的虚拟剖析

解释了如何在 80386 扩展内存上运行剖析器，而将实际的 640KB 内存全部留给用户程序使用。

附录 F　提示及出错信息

列出了可能出现的提示及出错信息，并指出如何回答这些信息。

0.3.1 关于几个术语的解释

为方便起见，本手册中经常会出现下列术语。它们是：模块、例程和参变量。

此为试读，需要完整PDF请访问：www.ertongbook.com

module(模块)

模块在本手册中不仅指 C 和汇编中的程序模块，还包括了 Pascal 中的单元(unit)。

routine(例程)

类似地，例程在本手册中是指 C 函数和 Pascal 的子程序(或例程，包括函数、过程和对象方法)。在 C 语言中，函数可以返回一个值(象 Pascal 中的函数)，也可以不返回值(象 Pascal 中的过程)。当 C 函数是后一种情况时，它被称之为空函数。我们将经常用例程来表示 C 函数和 Pascal 函数及过程。

argument(参变量)

本手册中参变量同参数(parameter)的意义是等同的。这也适用于启动一个被剖析同 Boland 的联系程序时所使用的命令行参变量和传递给例程的参变量。

第一章　剖析的实例

剖析(profiling)是人们目前了解最少的领域之一，但对开发高质量的软件却具有非常重要的意义。调查表明只有一小部分专业程序员使用剖析器提高代表代码效率。另外的研究表明，既使是最好的程序员在查找程序瓶颈所在时也经常做出错误的判断。

使用剖析器——这种普遍被忽视的工具有什么好处？其一，剖析可以提高程序的整体性能；其二，剖析能提高用户的能力，使他们能编写出高效率的程序代码；最后，剖析就象调试一样，能够缩短程序开发周期。

在本章中，我们将给出一个剖析的实例，并且告诉用户如何剖析可以更加节省时间。使用 Turbo Profiler 可以：

- 了解程序的每一部分各花费多少时间
- 给出带注释的源程序清单和剖析统计报告
- 保存统计数据，在此基础上进行多次剖析
- 在并列显示的窗口中分析统计资料和源代码

本章中给出的实例是找出并打印 1 到 1000 之间的所有素数。如果某个数是整数且只能被 1 和它本身整除，那么它就是素数；素数必须是奇数，因为任何偶数都能被 2 整除(2 是个例外)。判断素数的方法是看它能否被其它小于它的素数或大于 2，3 的整数整除。

剖析该样本程序的目的就是要加快搜索和打印素数的速度。在这个剖析过程中，用户可以学到如何使用 Turbo Profiler 检查程序的结构及其效率。

用户将要看到的第一个程序是 PRIME0。如果对它进行了剖析并且看出哪段代码需要修改，接下来就可以装入并剖析程序 PRIME1。除了 PRIME1 外，本章中出现的其它每一个程序(PRIME2，PRIME3，PRIME4，PRIME5，PRIME6)均是前一个程序改进后的版本。

PRIMEn.*是 C 语言程序。对于使用 Pascal 语言的编程者，我们在磁盘上提供了这些程序的 Turbo Pascal 版本(PRIMEnPA.*)，以便使这部分用户也能读完本章。除了对 C 程序剖析结果的分析外，我们也给出了运行 Pascal 程序后所得结果的注释。

在进行剖析之前，用户要确认所有样本程序(PRIMEn.C 和 PRIMEn.EXE 或 PRIMEnPA.PAS 和 PRIMEnPA.EXE) 是在当前目录之下。

我们提供的程序包括源程序和可执行代码两部分；Turbo Profiler 在进行分析时两者都要用到。同时每个样本都带有编译时的全部符号信息，因为 Turbo Profiler 也需要用到它们。

用户要确自己的程序包含了完整的符号信息。可以使用适当的编译选项做到这一点，如下所示：

- Boland 的 C++编译器：如果是在 IDE 中进行编译，选择“Options|Full Menus”，然后打开 Debugger 对话框(选择“Options|Debugger”)，将 Source Debugging 设置为“Standalone”。如果是在命令行中发出编译命令，可以使用命令行选项“-v”。
- Turbo Pascal(5.0 版或更高)：如果是在 IDE 中进行编译，将“Options|Debug

Information”和“Debug|Stand-Alone Debugging”选项设成 ON。如果是在命令行中发出编译命令，可以使用命令行选项“/v”。

- Turbo Assembler (1.0 版或更高)：使用“/Zi”命令行选项，然后通过“/v”选项使用 TLINK 连接程序。

尽管 TPROF 使用数学协处理器工作，但是如果用户系统具备 80x87 芯片，也只能暂时不用，这样才能得到与本手册中类似的结果(否则屏幕上出现的信息将与本书中的大不相同)。用户应该在运行剖析程序之前在 DOS 提示符下使用命令“SET 87=N”来设置环境变量。当然，用户得到的统计信息也可能与本章中的图示有所不同；这仅仅是因为各个系统之间的差别而已(如 CPU 类型、时钟速度等等)。

1.1 剖析一个程序(PRIME0)

用户剖析和优化一个程序有以下 4 个步骤：

1. 在剖析之前建立程序。
2. 在程序运行时收集数据。
3. 分析收集的数据。
4. 修改程序并重新进行编译。

在修改完程序之后，重复第 1 步到第 3 步以检查改动是否提高了程序的性能。

程序 PRIMF0 采用了 Euclid 的查找素数的方法，即直接检查进行除法后所得的余数。一旦发现了某个素数，就将其保存在数组 primes 中，而以后的数就用存放在 primes 中的每个素数去除以判断其是否为素数。

通过输入下列命令并按回车可以将 PRIME0 装入 Turbo Profiler：

```
TPROF  PRIME0
```

如果用户要对 PRIME0.C 的 Turbo Pascal 版本进行剖析，先要确保 PRIME0PA.PAS 和 PRIME0PA.EXE 是在当前目录之下，然后输入：

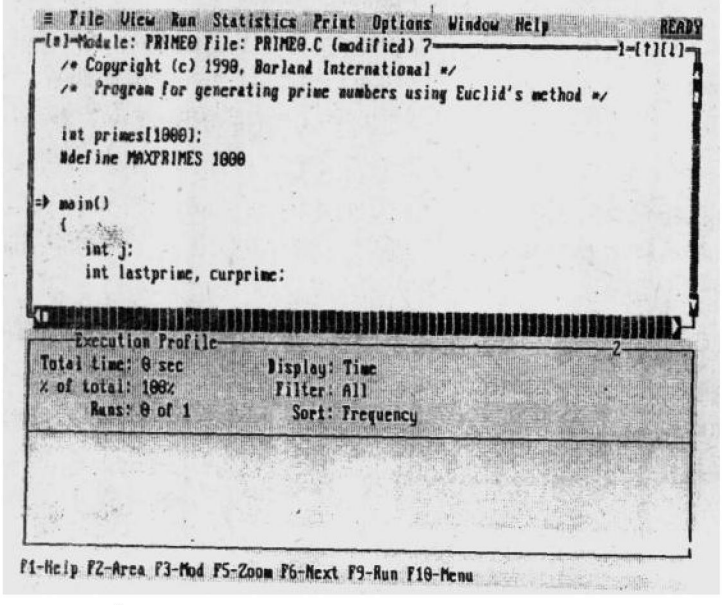


图 1.1 装入 PRIME0 后的 Turbo Profiler

TPROF PRIME0PA

Turbo Profiler 将会打开两个窗口：Module 窗口(显示了 PRIME0 的源程序代码)和 Execution Profile 窗口(在运行 PRIME0 后将会显示统计数据)。

运行 Profile 窗口(在运行 PRIM0 之后，静态显示 Profile)。如上图 1.1 所示。

Module 窗口和 Execution Profile 窗口与前面提到的剖析过程的第 1 步到第 3 步有关。用户可以使用 Module 窗口决定程序的哪一部分需要进行剖析。如果用户运行了某一个程序，Execution Profile 窗口将会显示出信息供用户分析程序使用。

1.1.1 设置剖析选项

用户在开始剖析程序之前，必须先指定要剖析的区域。“区域”是指在程序中需要收集统计数据的位置。区域可以是单行程序、程序结构例如一个循环或者整个例程。

要分析一个简短的例程(象 prime 和 main)，用户必须知道每行程序的执行次数及时间，这时可以将程序中的每一行都标志为区域。

1. 按下 ALT-F10 打开 Module 窗口的局部菜单。
2. 在局部菜单中选择 Add Areas。该菜单列出了区域边界供用户从中选择。
3. 选择 Every Line in Module。这样做将程序模块中的每一行都标志为区域，然后将光标返回到 Module 窗口。

注意这时程序模块中的所有可执行行都被打上了标记符(=▶)。

1.1.2 收集数据

现在已经为剖析过程的第 2 步做好了准备按下 F9 在 Turbo Profiler 下运行 PRIME0。该程序在用户屏幕(User screen)上打印出 1 到 1 000 之间的所有素数。当程序运行完毕后，Excution Profile 窗口中的信息就是该程序的统计数据。

采用下列方法可以放大 Execution Profile 窗口：按下 F5 或在 Window 菜单中选择 Zoom。这时的 Execution Profile 窗口如下图所示：

窗口上面的方格中显示了程序总的运行时间，下面的方格中显示了其它数据信息。每行数据信息具有四个域：

- 区域名
- 该区域所花费的时间(秒)
- 该区域花费时间占全部运行时间的百分比
- 条形比例图，显示了上面的百分比

例如，下面这一行

#PRIME0#31 6.2655 sec 93% | ==========

告诉用户模块 PRIME0 的第三十一行代码的执行大约花费了 6.3 秒——占所有标记区域总运行时间的 93%。相应地，第 31 行的条形图最长是因为该行程序是整个标记区域中最耗时的。

在程序 PRIME0PA 中，该行程序所对应的位置是在第 42 行。

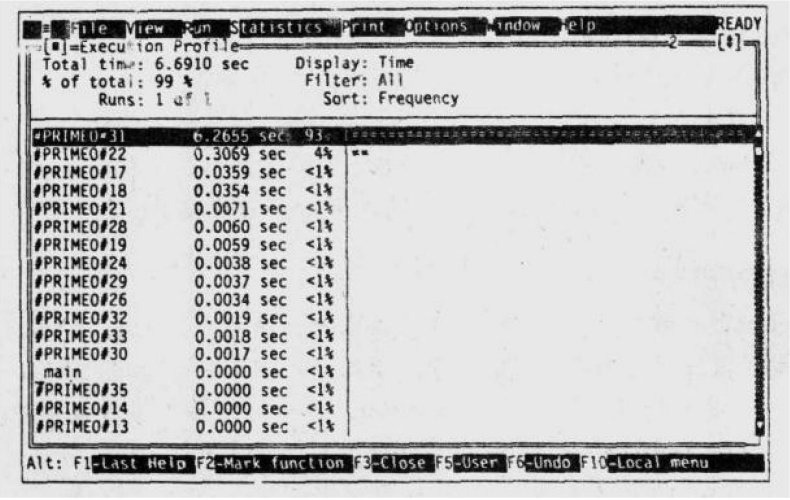
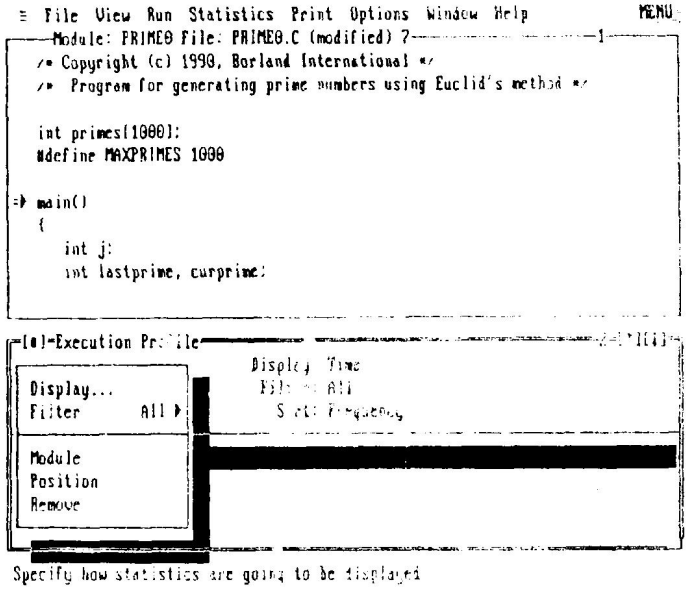


图 1.2　PRIME0 的统计数据

1.1.3　显示统计数据

用户同样可以显示关于执行计数(也称执行频率)的统计数据。



1. 按下 ALT-F10 打开 Execution Profile 窗口的局部菜单(如上图)。
2. 在局部菜单选择 Display。
3. Display Options 对话框将出现在 Execution Profile 窗口中显示数据的六种方式。

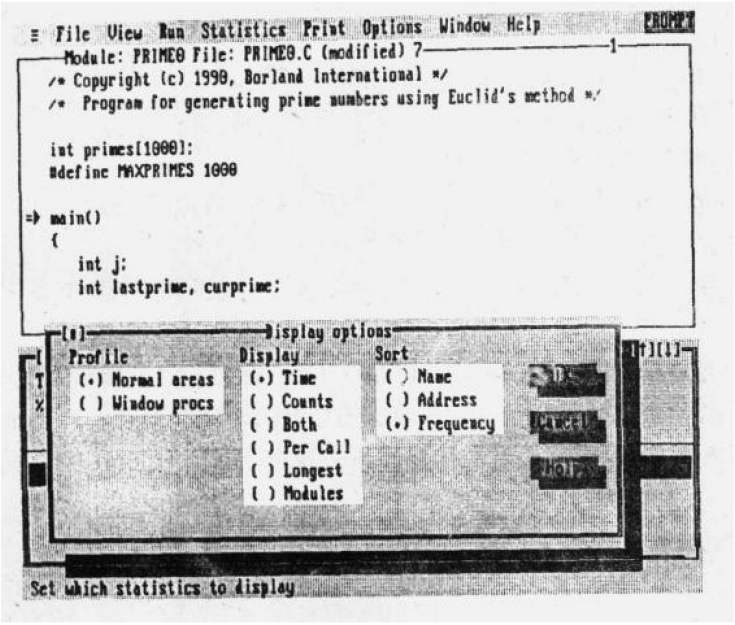


图 1.3 Display Options 对话框

- Time(缺省的设置)显示在每个标记区域中花费的总时间。
- Counts 显示每个区域执行过的次数。
- Both 在屏幕上同时显示时间和次数数据。
- Per Call 显示每次调用花费的平均时间。
- Longest：显示每个区域所花费的最长时间。
- Modules(和被动分析一起使用)显示每个程序模块所花费的时间。

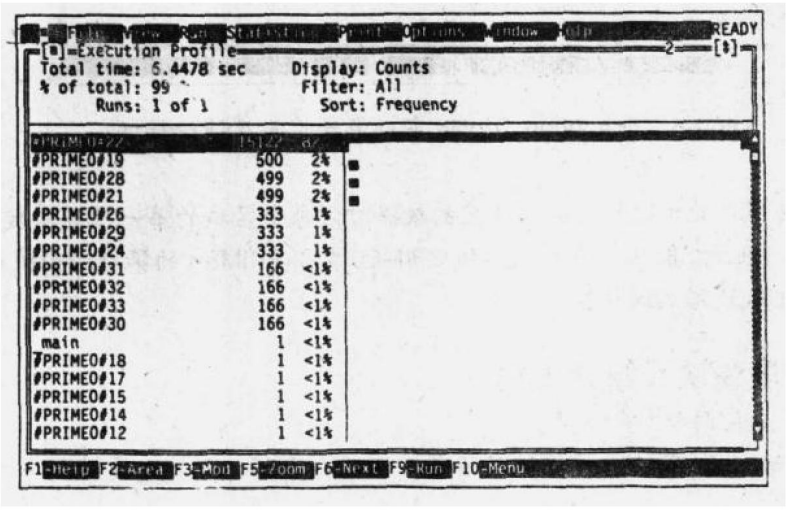


图 1.4 Excution Profile 窗口中的计数显示

此为试读，需要完整PDF请访问：www.ertongbook.com