计算机图形与图像丛书

高级实用开发工具精粹

PC TECHNIQUES C/C++ POWER TOOLS®

JEFF DUNTEMANN AND KEITH WEISKAMP 著

张维存 等译
流 水
甘 特 审校

学 苑 出 版 社

PC Techniques C/C++ Power Tools

PC 技术
C/C++ 高级实用开发工具精粹

Jeff Duntemann & Keith Weiskamp 著
张维存 流 水等 译
甘 特 审校

学 苑 出 版 社

(京)新登字 151 号

内 容 提 要

本书分专题详细地介绍了 C/C++工具，主要介绍了如下内容：确定及访问硬件、键盘和键盘输入、磁盘文件操作、文本视频编程、图形编程、关于用户界面的编程、打印机和打印机编程、串口编程、内存管理、排序与检索方法、操作串数据、数据结构、对时间和日期的处理、算法与编程的方法、Windows 编程、编写内存驻留程序、对 DOS 的编程、系统级编程、对 BIOS 的编程、代码优化技术、调试和维护代码、编程方法等内容。

本书叙述清晰，通俗易懂，使用方便，适用于具有一些计算机基础知识的读者使用，同时也是从事 C/C++的用户的极其有用的参考书。

需要本书者，请与北京海淀 8721 信箱书刊部联系，邮政编码：100080，**电话**：2562329。

版 权 声 明

计算机图形与图像丛书

PC 技术

C/C++高级实用开发工具精粹

著　　者：Jeff Duntemann & Keith Weiskamp
译　　者：张维存　流　水
审　　校：甘　特
责任编辑：甄国宪
出版发行：学苑出版社　　**邮政编码**：100036
社　　址：北京市海淀区万寿路西街 11 号
印　　刷：北京双青印刷厂
开　　本：787×1092　1/16
印　　张：36.25　**字数**：841 千字
印　　数：1～5000 册
版　　次：1994 年 6 月北京第 1 版第 1 次
ISBN7－5077－0884－5/TP・26
本册定价：78.00 元(含盘)

引 言

某些念头的产生往往是因为你的知识有空白存在。我想赶快 写这本书，则是因为以前我的刊物中出现了空档。

还在 1988 年时，我主持一种名叫《TURBO-TECHNIX》的期刊，该期刊由 Borland International 公司出版。它深受欢迎并为公司带来了很多好处。后来由于广告收入不足，刊物停办。在 1988 年中，我们着手重新设计 11 月/12 月一期的版面，当时它的版面到处都是空白。

你知道，要在 160 页版面上登载各类文章、广告、表格，并使这些内容整洁紧凑，这决非一件易事，你不能弄得太挤，也不能让空白太多，一切要正合适。要么花费大量时间对正文进行剪辑；对照片、图表进行压缩，要么只好学会保留空白。我们的艺术主编不喜欢空白，而我又不喜欢剪剪拼拼。而且对这类联系紧密的文章也没有多少可删的。这与我们的对象是肥皂剧文摘不一样。如果你执意要剪裁，往往会损失内容。

于是我想起了一个主意，那就是专门设计一些小的、篇幅可变的文章用来填补那些空档。我让作者写这类大小从几行到一页的文章，并使自己手头总有一堆这类文章备用。这样，当我们在编录一期刊物出现空白时，都能找到一些东西来填补。

受到一时的鼓舞，我写了几篇，并管它们叫 HAX，我们的艺术主编很喜欢。于是开始为编 11 月/12 月这一期着手积累这类文章。在 9 月这期刊物编成之际，Borland 公司认为拉的线太长，考虑到广告收益，公司决定收手。

我花了一年时间编成了另外一种刊物，这次是同 Keith Weiskamp 使用。当《PC 技术》的第一期于 1990 年 3 月面世时，HAX 就被收了进去。不过，这回有了很大的不同。这些文章不再是仅仅用以填补空档，而已成为该刊物的重大特色。杂志一上市，人们便峰拥抢购。

我觉得(尽管不尽准确)大的主题是书来阐述，小一点的由杂志中的文章阐述，更小一点的，则应用 HAX。那些较小的主题在别的刊物中都未曾涉及。出版几期后，我们的 HAX 就已演变成在别处见不到的知识系列。别的刊物要么不在乎空档，要么干脆将其裁剪掉。没多长时间，我们有了数百篇 HAX，要么已付印，要么正待付印，而且还有新的源源不断地寄来。至此，我们非常自然地想到要把这些收集到的小知识、小技巧、“密诀”(它们都是 PC 编程中很精深且不广为人知的东西)汇集成书。

这本书一部分内容是从《PC 技术》上转印的，这包括摘自现已无从可得的早几期的很多 HAX 和短文。我们以为后来的读者应该仍然可以得到这些信息，并在需要时能在书架上随手取到。然而，这儿出现的大量的 HAX 却是首次面向读者。

在结束这篇序前，我想为在这本书的写成提供过帮助的人致以敬意。Keith 和我是主要负责的(尤其是 Keith)，我们的名字已印在了封面，然而，投入到本书创作这一艰巨工作的集体努力却是巨大的。Martin Waldron 对所有新材料作了技术编辑；他的才智使得一百五十个左右联系松散的部分紧凑地合在了一起。没有他的帮助及其巧妙的编辑技巧，也就没有这本书。Robin Watkins 管理编辑和生产方面(协调一百多个风格各异的人工作决非

轻松)。Bantam 的 Jean Davis Taft 帮助我们出版了一流的书。Brad Grannis 在 Barbara Nicholson 的协助下作了大部分的排版工作。Lenity Himburg 编排了目录。此外还感谢 Robin 负责与一百多位有关作者联系并告知他们已出版的事。最后,我们感谢 Steve Guty,我们在 Bantam 的了不起的编辑,以及众多的为本书作出贡献的才华出众的作者。

我们希望在将来再出一些类似的书。材料总是不缺的。您想看到些什么?请给我一个信息。如果您对本书内容感到满意,何不试翻翻《PC 技术》?我们每两个月刊登一些"它如何工作"和"怎么办"之类的实用性材料(比如时钟功能),详见本书封皮的介绍。

无论做什么,都不要放弃!

Jeff Duntemann KGTJF
Scottsdale, Arizona
September 1992

目 录

第一章　确定及访问硬件

作为程序员，我们有时忘记软件存在之前尚有硬件，特别是考虑到大学中针对可移植性的态度时更是如此：该类观点以为硬件“不洁”，应被尽量压在多层屏蔽性“虚拟机”下边。

这实在是笑话。

我们不应在所有机器上都建立好几个层次以使它们看起来都一样（这样做的结果往往是使它们具有最小的共性，从而使这种做法显得毫无理由），而是应该选定一个硬件平台，尽可能了解它，编写出最大限度运用它的软件。

要做到这点，首先要了解硬件是如何工作的，编写软件来辨别和确定给定系统都安装了些什么硬件。这一步做到了，接下来是既要避免用危险的或易导致硬件溢出的方法来控制硬件，同时又要使机器工作得尽可能得迅速高效。在第一章中我们的作者将阐述如何辨别和确定已安装的硬件以及如何直接存取它（同时对我们可能遇到的硬件古怪行为给予零星的提示）。在做出判断以前我们需要安全有效地使用这类信息。当然，你一边看一边就能不断获取这类信息。在我带给微机世界的各种增强功能中，我最喜欢的便是前面板的重置开关。犯错误在所难免，重新启动应该变得容易。

1.1　定位磁盘驱动器

■ Deborah L. Cooper

这儿为了帮助你搜寻连接到微机的所有驱动器提供了一个有用的工具。

几年前我用汇编语言写了一个寻找文件的程序。尽管当时很多功能类似的程序已有了（现在情况仍然如此），我仍把它作为递归编程练习来做。

但是就在最近，我想把这个程序做一修改，使它在寻找指定文件时能自动搜寻所有磁盘驱动器。我所提到的这个例程在程序 1.1 中。DRIVES. ASM 文件是一个演示程序，在其自动且迅速地搜索所有附着在系统上的磁盘驱动器时显示“Valid drive：#”的信息。

用以完成此任务的方法很简单。首先将当前测试驱动器存起来，以便测定后恢复。测试本身是针对以 A 开头的可能的驱动器号的循环。通过功能 0EH，选择新的驱动器，设置当前驱动器为序列中下一个驱动器号。判断该驱动器是否存在，只要调用功能 19H，即获取缺省驱动器功能。如果系统被设置成一个合法的缺省驱动器号，寄存器 DL 和 AL 在 DOS 调用返回后将相等，指出该驱动器确实存在。

通过对驱动器号作循环可测试所有值落在 1（A：）到 26）Z：）的有效驱动器。注意，如果你的系统在驱动器号的指定中存在空缺，（比如，你有一个叫 F：的驱动器，但没有驱动器 D：或 E：）该程序在碰到第一个空缺时会停止搜索。

代码描述

概述

所列程序是用来搜寻连接到微机上的所有驱动器。

代码说明

文件	描述	使用者
DRIVE. ASM	完成驱动器搜寻的汇编语言程序	————

编译/汇编

DRIVE. ASM 可用任一与 TASM 兼容之汇编程序汇编并用 EXE2BIN 或 TLINK 的 It 选项将其转变为 COM 文件。

程序 1. 1　DRIVES. ASM

```
;DRIVES. ASM
;(c) 1991 by Deborah L. Cooper
;
codesg   segment
         assume   cs:codesg
         org      100h                ;make this a COM program
start:   jmp      start_2             ;skip over data area

org_drive        db       0           ;current drive code
got_drive        db       0dh,0ah,'Valid drive: ','$' ;output msg

;Get and save the current drive so we can restore it later on
start_2:
         mov      ah,19h              ;get default drive code
         int      21h                 ;call dos
         mov      org_drive,al        ;save it (0=A, 1=B, ... )
;Now go through 1 - 26 possible drive codes and display the
;message when we find a valid disk drive
select: mov       ah,0eh              ;select drive function
         mov      dl,0                ;start with Drive A
         int      21h                 ;call dos
;Use function 19h to see if drive actually exists
select_2:
         mov      ah,19h              ;get default disk drive
         int      21h                 ;call dos
         cmp      dl,al               ;is it a valid drive?
         je       got_one             ;yes, show the message!
next_drive:
         inc      dl                  ;no, prepare to test next one
         cmp      dl,27d              ;maximum 26 possible!
         jae      all_done            ;go if done now
         mov      ah,0eh              ;select drive function
```

```
        int     21h                     ;call dos
        jmp     select_2                ;and go back to test it
got_one:
        push    dx                      ;save our drive number
        mov     dx,offset got_drive     ;point DX to message
        mov     ah,09h                  ;display string function
        int     21h                     ;call dos
;Show the message with drive letter
        pop     dx                      ;recover test drive code
        push    dx                      ;save on stack for a minute
        xchg    al,dl                   ;put drive code in AL for printing
        add     al,41h                  ;convert to ASCII letter
        mov     ah,0eh                  ;display byte function
        int     10h                     ;call bios
        pop     dx                      ;recover our drive number
        jmp     next_drive              ;and go back
;Now restore our original drive and exit
all_done:
        mov     ah,0eh                  ;select drive function
        mov     dl,org_drive            ;original drive code
        int     21h                     ;call dos
        mov     ah,4ch                  ;terminate program function
        int     21h                     ;call dos
codesg  ends
        end     start                   ;end of demo!
```

1.2 简单的 CPU 检测

■ Nicholas Wilt

这是检测微机 CPU 的一个有用工具。

用软件检测具体的 CPU 模式既复杂又容易出错。这是因为很多 CPU 检测例程都很繁琐：我是在 8088 还是在 NEC V20 上运行？它是 80386SX 呢还是 80386DX？这类问题很困难，而且随着 Intel 系列以及与 Intel 兼容的处理器越来越多将变得更加困难。

幸运的是，你并不一定总要了解你在其上运行的 CPU 模式。从软件角度来说，只要将它分成 80286 以前系列、80286 以及 80386 就行了。如果你在其上运行的是 80286 以前的机器，你将很痛苦—没有扩展指令，没有 32 位的寄存器。如果运行的是 80286 或更高级的机器，就可以使用扩展指令集，包括 PUSH、数、INS(输入字符串)和 OUT(输出字符串)。如果运行的是 80386 或更好的，你就可以用 32 位寄存器和更加扩展的指令集(包括 SET 条件)。可以利用提供的各种便利编写出最高度优化的代码。

程序 1.2 给出了一个返回 CPU 类型的简短的汇编语言函数。如果机器的 CPU 是 80286 以前的(包括 8086、8088、V20 和 V30)，函数返回 0。如果 CPU 是 80286，返回 1，如果 CPU 是 80386 级机器，则返回 2。程序 1.3 是一个 C 语言的测试程序，程序 1.4 是产生测试程序的简单批处理文件。该批处理文件对目录结构做了一些假设，在试用它之前先检查一下。

函数没有参数，故将它纳入各种语言环境以及调用起来都很方便。只要将在汇编语言源程序首部的.MODEL 标头加以修改，便可适应别的编译器的内存模式。

代码描述

概述

所提供的 C 和 ASM 程序是用来检测 CPU 类型的—8086、80286 或是 80386。

代码说明

文件	描述	使用者
CPUTEST. C	L. C 语言测试程序，使用了汇编语言源程序 CPUTYPE. ASM 来检测 CPU 类型	CPUTYPE. ASM
CPUTYPE. ASM	包含用以决定处理器类型的低级函数 cputype	————
MAKECPU. BAT	批处理文件，用 Borland C++对以上源代码进行汇编、编译、连接	————

编译/汇编

用批处理文件或是用 Borland C++和 TASM 分别汇编和编译。如果你用批处理文件，可能需作一些修改，以使它适应目录配置。

程序 1.2　CPUTYPE. ASM

```
.MODEL LARGE,C

.386

.CODE

; cputype: function returns 0 if 8088 family,
;         1 if 80286 family,
;         2 if 80386 or better.
; No arguments.
; Returns value in AX.

        PUBLIC      cputype

cputype PROC
xor     dx,dx               ; Assume 8086 for now
push    dx                  ; flags <- 0
popf                        ;
pushf                       ; Get flags back
pop     ax                  ;
and     ax,0f000h           ; If the top 4 bits are set,
cmp     ax,0f000h           ;
je      quitcputype         ; return.   It's an 8086.
inc     dx                  ; It's at least an 80286.
```

```
push        0F000h                  ; flags <- 0F000h
popf                                ;
pushf                               ; Get flags back
pop         ax                      ;
and         ax,0F000h               ; If the top 4 bits aren't set,
jz          quitcputype             ; it's an 80286
inc         dx                      ; Otherwise it's a 386 or better
quitcputype:
xchg        ax,dx                   ; Return value in AX
ret                                 ; Return
cputype ENDP

END
```

程序 1.3 CPUTEST.C

```
/* cputest.c: Tests the cputype() function.  Prints message
 * telling the results.
 */

#include <stdio.h>
int cputype(void);

void main(void)
{
  printf("cputype detects an ");
  switch (cputype()) {
    case 0:
      printf("8086.  I sympathize.\n");
      break;
    case 1:
      printf("80286.  You\'re getting there!  \n");
      break;
    case 2:
      printf("80386.  You\'re cookin\'!");
      break;
  }
}
```

程序 1.4 MAKECPU.BAT

```
tcc  -I\TC\INCLUDE -ml -v -c cputest.c
tasm /mx/zi cputype.asm
tlink /v/c/map \lib\c0l test cputype,cputest.exe,,\tc\lib\emu \tc
\lib\cl \tc\lib\mathl
```

1.3 一个C++的I/O端口类

■ Kevm Dean

这是一个有用的头文件，可在你的C++程序中包括它以便在做I/O端口访问时使用对象。

不像Turbo C的inport()和outport()函数，Turbo Pascal的Port和PortW数组是对硬件端口编址的有创见的方法。下面两行代码在功能上虽然相同。

```
out portb(0X3F8,data);  /* Turbo C */
Port[$3F8]:=Data  {Turbo Pascal}
```

(他们都从端口COM1发送一字节)，后者更接近于下面的概念，即端口是可以像对一般内存一样读写的对象。

这儿，关键词是对象。通过TURBO C++，我们可以将inportb()和outputb()封装到一个类中，并且比Turbo Pascal的Port和PortW数组功能更强(见程序1.5)。

技巧：一个类成员函数带有const修改符表示该函数不修改类的任何成员。任何试图对const对象调用非const函数的做法都可能导致编译错或警告。

因为一些端口是作为位掩码来使用的(例如，一个COM端口的调制解调器控制寄存器控制着中断和某些线信号)，所以像/=，&=和 ^ =这类运算符被定义用来简化位操作。同时，许多设备使用的不是单个端口而是端口数组并将其中一个定义为基(例如，COM1对应0X3F8)。运算符[ ](整数下标)定义在byteport类中，通过用地址_addr+index存取端口简化了存取端口数组的操作。例如，

```
char data;
byteport com1(0X3F8); //COM1的基端口
corn1=data;  //发送一个字节
com1[4] |= 0x08  //置中断
data=com1  //接收一字节
```

以上代码为COM1声明了一端口对象，发送了一字节，通过调制解调器控制寄存器(地址为0X3F8+4=0X3FC)置中断，最后接收一字节。

为了模拟Turbo Pascal的PortW数组，拷贝bytePort类定义，重新命名它为wordPort，将byte类型的例子都替换成Word类型。

代码描述

概述

此头文件可以用来支持C++ I/O端口类。在任何需对I/O端口进行读写操作的C++程序中都可使用此文件。

代码说明

文件	描述	使用者
PORTS. H	一个包括用来存取 I/O 端口类的 C++头文件	————

编译/汇编

用#include 在 C++程序中包括 PORTS. H

程序 1.5　PORTS. H

```
// Get prototypes for inport and outport functions.
#include <dos.h>

typedef unsigned char byte;
typedef unsigned short word;

// Byte port.
class bytePort {
  word _addr;    // Port address

  public:

  // Constructors and destructor
  bytePort(word addr) : _addr(addr) { }
  bytePort(const bytePort &p) : _addr(p._addr) { }
  bytePort() { }
  ~bytePort() { }

  // Set port address
  void addr(word addr) {
  _addr = addr;
  }

  // Get port address
  word addr() const {
  return _addr;
  }

  // Read byte value from port
  operator byte() const {
  return inportb(_addr);
  }

  // Write byte value to port
  bytePort& operator=(byte value) const {
  outportb(_addr, value);
  return *this;
  }

  // Set bits at port
```