中断处理程序扩充最有效的方法

陈钦梧　（汕头大学计算机与信息工程系）

摘要:本文提出用递归调用和栏截技术来扩充中断处理程序的方法。它有效地利用原有系统资源来进行扩充,化繁为简,使解决问题变得省力又不产生副作用。

一、引言

随着技术飞速的发展,需求不断的增加,旧的程序常常需要不断的修改和扩充。IBMPC XT/AT 系列电脑中,许多基本 I/O 管理均固化在 ROM BIOS 中而且一般是通过软中断指令调用。例如键盘管理(INT 16H)、屏幕显示(INT 10H)、打印输出(INT 17H)等。对这些 BIOS 扩充最好的方法是:①保持系统的向下兼容,并能使用原有系统的中断类型型号;②继承原有系统的功能,即系统功能调用原 BIOS 功能,但不增加中断类型号。

对上述①点也是开发人员必须保证的。而对于第②点,现有系统几乎均有点走弯路。

以 CCBIOS 为例,原有 ROM BIOS 只能处理西文 ASCII 码。为了识别和处理汉字,必须对原 BIOS——INT10H,INT16H,INT17H 进行功能扩充。

早期的一些 CCBIOS 是先分析原 BIOS 功能,在需要的地方进行修改。这要求编程者理解原 BIOS 的意图和结构,甚至涉及很多硬件组织等方面的知识,改起来还很繁锁。而且固化的 ROM BIOS 变成一点用处都没有,似乎也是一种浪费,有时只是修改一点或扩充一点功能,则牵一发而动全身,非常不便。

造成上述困难在于 CPU 没有直接指令可以不通过中断向量来调用中断处理程序(IRET 指令结尾者)。因此,有的系统[1]不得不增加一个中断向量来存放原 BIOS 的入口地址。这样做的好处是,只要求编程者了解原 BIOS 的接口部分即可,需要时可以被引用。因而使编程变得省事又节约代码在 RAM 中的长度。其缺点是,由于 PC 系统提供的中断向量绝大多数已分配有固定的功能类型[2][3],因而每增加一个中断向量都很费事,弄不好还会与别的系统冲突。

后来,人们才想出了用指令组合:

pushF

call Far Old__BIOS__Entrg 来模拟软中断指令的办法。它兼有上述方法的优点又避免增加一个中断向量,因而被后来的很多系统开发者引用——如 CCDOS4.0 后用的即是此法。可以说是解决问题的一种很好办法。

以下提出的栏截和递归调用技术更耐人寻味。作者在开发程序过程中也曾遇到上述困难。通过研究,发现可以用栏截和递归调用来扩充中断处理程序,经过几年来的多次实践,证明它是很有效的。下面就来讨论这一方法。

二、拦截技术

对于一些系统的小缺陷进行修改或小扩充,用拦截技术是最有效的。

实例:某机动车自动检测程序都是编绎 BASIC 的目标程序(.EXE 文件)。由于硬件的变更,只能配另外版本的 CCDOS。可是程序运行之后,所有汉字却都不能正常显示了。

分析:由于编绎 BASIC 目标程序一运行就设置显示模式为 80×25 字符文本模式(AL=03),而多数 CCDOS 汉字显示均基于图形模式(如 AL=06),因而在(AL=03)文本模式下,汉字无法正常显示出来。

如果能在每个源程序头加 Screen2(置图形显示模式)语句再编译,则汉字就能正确显示了。遗憾的是,用户拥有的仅仅是目标程序(.EXE 文件)。

用 Debug 对目标程序进行分析修改也许是可行的,但这有相当的难度。最糟的还是,整个检测系统由一百多个 EXE 文件组成,显然这么修改太烦了。怎么办?

一个引人入胜的解决方案是:切掉 CCDOS 文本显示模式,将其强制设为图形汉字显示模式(AS=06)。

那么,如何切掉?去修改 CCDOS 吗?

我们是在 CCDOS 启动后,执行.EXE 程序之前,插入下面的 Xmode 命令。

Xmode.com 文件如下所示:

```
2488:0100 EB10      JMP  0112
2488:0106 08E4      OR   AH,AH;是设置显示模式功能码?
2488:0108 7503      JNZ  010D
2488:010A B006      MOV  AL,06;若是,强制设为图形显示模式
2488:010C 90        NOP
2488:010D 2E        CS:
2488:010E FF2E0201 JMP  FAR[0102];跳至原CCBIOS(INT10)入口
2488:0112 E80800    CALL 011D
2488:0115 B409      MOV  AH,09
2488:0117 BA8001    MOV  DX,0180 ;显示信息
2488:011A CD21      INT  21
2488:011C C3        RET  ;正常(不驻留)退出
2488:011D B81035    MOV  AX,3510
2488:0120 CD21      INT  21
2488:0122 BA0601    MOV  DX,0106
2488:0125 39DA      CMP  DX,BX;Xmode已驻留过?
2488:0127 74F3      JZ   011C
2488:0129 891E0201  MOV  [0102],BX;保存原INT10入口地址
2488:012D 8C060401 MOV  [0104],ES
2488:0131 B81025    MOV  AX,2510;修改INT10向量,指向CS:106
2488:0134 CD21      INT  21
2488:0136 E8DCFF    CALL 0115
2488:0139 5A        POP  DX
2488:013A CD27      INT  27;驻留退出
2488:0180 D8'Xmode V1.01 圆形汉字模式设置',0D,0A,
          '汕头大学陈钦梧 1989.01.05',0D,0A'$'
```

程序说明:在编移 11D-127 处的指令旨在检测 Xmode 是否第二次执行。若不然,才可驻留退出。

偏移 139 处 POP DX 指令用了一点技巧:它实质上取出了偏移 112 处 CALL 指令的下一指令地址 115,以此作为驻留长度,可方便以后用 DEBUG 再对程序进行修改。若改用 MOVDX,115 指令也可,唯程序修改后可能还要再修改 MOV 指令。

扩充的中断处理程序入口在 CS:106 处。它完成的功能再简单不过了:首先检测 AH 是否为 0——设置显示模式?若是,置 AL=06,然后就都跳到原 CCBIOS 的入口地址。

我们称偏移 106 至 10D 几个指令为"拦截"因为"拦截"后跳回原 CCBIOS 处理。

Xmode 带来的好处是,不管什么样的 CCBIOS 都行,也必须分析其内部。

事实上,目前很多用户都可能会碰上因显示模式错而出现类似上述汉字不能正确显示的问题。不妨用 Xmode 试一试,也许问题就能得到解决!注意,VGA 或 EGA 图形汉字显示模式可能不是 AL=06,则须将偏移 10A 处 MOV AL,06 指令适当修改。

三、递归调用技术

递归调用是指一个子程序内含有调用它自身的指令。为了不致这种调用陷入死循环——即递归终止的条件,是递归子程序内必须要有条件分支指令。对递归进一步的讨论可参考程序设计语言或算法方面的书[4]。这里仅介绍一种利用递归技术扩充中断处理程序的方法。其步骤一般是:

1.中断处理程序均是在 AH 寄存器中放功能号,且多数仅若干功能,故一般 AH<80H。现在可对扩充的中断处理程序增加 AH≥80H 的功能号,并定义它们对应于旧的 BIOS 中断处理程序的功能,二者的关系是:

新功能号 AH and 7FH=旧功能号 AH

2.在扩充的中断处理程序开头加一些条件分支指令,使 AH≥80H 者跳转至旧的 BIOS 中断处理程序,以实现递归终止的条件。

3.新的(即扩充的)AH<80H 功能号保持与旧的 BIOS 向下兼容。例如对 CCBIOS,它既能处理西文(旧功能),也能处理汉字(新功能)。这部分是开发人员要做的主要工作。

4.当在扩充的功能号 AH<80H 内需要调用到旧的 BIOS 功能时,将旧功能号加 80H,如旧功能号 AH=01,现在要用 AH=81H,即可递归调用了。

参考资料:

[1]<<汉字 2.13H 源程序详解>>北京希望电脑公司

[2]<<IBM personal Computer Hardware Reference Library>>赵建雄、余良吟译<<PC 硬体电路分析>>联星出版社,1984 年修订本

[3]<<IBM PC/AT 技术参考手册>>中册

[4]曹注和、刘椿年译<<算法+数据结构=程序>>,科学出版社,1984