Windows 通用菜单生成器

王君杰　(清华大学自动化系)

摘要:在许多管理信息系统和决策支持系统中,用户对信息的需求乃至相应的操作都具有很大的不确定性,这就需要一套灵活的菜单生成工具。作者分析了当今一些商业软件在这方面的成功与不足,提出了一种将菜单的安装和相应代码的生成一步集成的方法。

一、背景介绍

在企业、机关的日常业务中,涉及到管理、决策等活动的信息无论从形式、内容、数据来源都有很大的不确定性。因此,MIS、DSS 的使用过程中,用户会发现原有的菜单、报表等工具不能完全满足工作中的需求,他所能做的只有向开发人员求助,回到语言环境,打开源程序,加入相应的代码,这样的反复在软件维护中是相当频繁的,我们称之为"需求爆炸"。即使正式运行后,这种问题还是会经常困扰着开发人员。这里面还蕴涵了一个软件结构化的问题。一般的应用程序开发时,其界面规划和编程是分离的,比如在菜单中多加了一项,就必须在相应的程序中再加入一段代码,这种非结构化给软件的维护增加了开销。所以,开发人员就会想到:与其用两个月的时间搜肠刮肚考虑用户的一百个需求,还不如站高一步,为用户开发一个易学易懂、能自己操作的构造工具,让用户在工作中按照自己的意愿去组织环境,例如:菜单生成器、报表生成器等等,以不变应万变。

在可视性语言如火如荼的今天,习惯于传统 Windows 编程的程序员们不必为大量繁琐而又无需技巧的工作头痛,"所见即所得"的思想为他们节约了大量的宝贵时间,各种软件的交互性有了很大的提高。

在菜单生成系统上,Foxpro 2.6 for Windows,Visual Basic 3.0,Visual C++ 2.0,Borland C++4.0 等软件都采取了交互性很强的菜单生成方式,用户可以在类似于表格或对话框的环境中象填写表格一样地描述自己所要建立的菜单,而不需要什么专业知识,并且可以随时测试新菜单以检验其正确性.但是,这种方式也存在一个很大的缺点,它们都是基于某个应用程序的,定义出来的菜单资源只是附属于一个确定的应用程序,而没有独立性和可拼装性。而且在 C / C++语言中最常用的资源描述文件中,除了菜单资源外,还包括了一些与应用程序密切相关的对话框、点位图等资源,这就大大削弱了其可移植性。

毫无疑问,Visual Basic(以下简称 VB)、Visual C++(以下简称 VC)等语言环境都是面向程序员而不是用户,虽然提供了丰富的工具,但每个工具都是与应用程序的编程密切相关的,即使用户可以在简单的工具中完成一部分界面规划,例如菜单的内容、对话框的绘制,但若是没有编程经验的话,他所建立的资源是无法使用的,比方说,他在 Resource Workshop 中建立了名目繁多的菜单,他却无法写出:

```
Switch(Message)
 {case WM#J_COMMAND:
   case ID_MENUI:
        ......
   break;
   ......
  break;
  ......
 }
```

再漂亮的资源也无法使用。

二、软件功能及使用

笔者在详细分析了这些软件的菜单生成机制后,下决心在菜单资源的独立性和易学性上作些改进,开发出

了基于 Windows 环境的通用菜单生成器,可以作为一个应用程序模块移植入任意一个 Windows 应用软件,可以完成菜单的编辑、维护管理、安装工作。

整个软件分为三个模块:菜单编辑模块、菜单文件管理模块以及菜单安装模块。

在菜单编辑模块中,提供了两种编辑方式:菜单窗口方式及电子表格方式。菜单窗口方式同时面向用户和开发人员,允许对菜单的标题、便捷键、激活进程、各项属性进行定义,以列表形式显示,不同层次采用不同的伸缩格式,效果一目了然,并可对菜单项进行插入、删除、测试操作,测试结果当即显示,以便用户当场修改。在这里需要强调的是,菜单项所对应的激活进程可以是任何语言编写的,只要编译成可执行模块即可安装。在这一意义上,该菜单生成器大大提高了软件的复用性,比起目前的一些商用软件是一个极大的飞跃。

电子表格方式主要面向开发人员,显示各菜单的完整数据结构,包括标题、编号、父节点、激活进程、层次及各项属性,便于开发人员在调试过程中参考,并可直接在表格上修改编辑。

编辑完毕后,可将菜单资源以文件形式存盘,今后即可按不同需求安装至任意窗口,菜单资源的独立性即体现于此。

菜单文件管理模块主要用于对菜单文件的各项操作,例如列举文件名、拷贝、删除、文件内容快速浏览。

菜单安装模块可以在任意窗口上安装任意已有的菜单,选定菜单文件后,安装程序即可检索该文件中所有菜单项的数据结构,安装至指定的窗口。

菜单的安装过程分为两步,首先,安装菜单的界面部分;然后是菜单中所指定的激活进程的代码生成。

一般来说,用户创建菜单后,必须在相应子程序段中编制相应的代码,才能正确响应菜单命令(Visual C++,Visual Basic,Foxpro 等都是如此),而我们的目标就是将这种两步到位的机制改进为一步到位,即菜单的创建与相应的代码生成由系统统一完成。

三、编程思想及实现

在笔者开发的菜单生成器中,可以将菜单的界面部分与它所激活的进程绑在一起。它所激活的进程可以在生成菜单时一并指定,用户在指定窗口加入一菜单项,其激活进程被自动引入。删除某一菜单项时,其激活进程被自动引出。

实现这一目标并不是一件容易的事。在 Foxpro 环境下,我们可以在运行过程中象写普通文本文件一样直接生成 Menu1.prg,然后在主控程序中直接写入 Do Menu1.prg 即可。

但 VB 中要实现这种功能就不是那么容易了.当你安装入 Menu1 后,VB 自动生成与改菜单项有关的事件驱动子程序的头尾:

```
Sub Menu1_click()
  end Sub
```

一旦选取了 Menu1 后,系统将自动进入该子程序执行命令,而不会执行你所创建的进程,也就是说,VB 只认自己的家门,而不认识你为它打开的房间,所以要它取出你的东西,唯一的办法就是把你的东西放进他的房间。

在这种情况下,我们采用了"模板窗口"的思想,事先定义了一窗口,安装了许多层次的菜单(但使其不可见),这样,VB 就为你打开它的房间,我们在子程序中加入了以下语句:

```
Sub Menu1_click()
      shell("m1.exe",1)
  End Sub
```

这里,m1.exe 是个虚拟的可执行模块,没有任何功能。用户在运行过程中安装其菜单时,只需将与菜单项对应的激活进程拷贝为 m1.exe,则相当于 VB 的房间里放入了我们自己的东西。至此安装完毕,退出系统后,自动删除所有虚拟模块。这种思想很简单,但必须有严密的算法,才能使菜单和进程完全一致。

四、应用及评价

该菜单生成器在几个中等规模的管理信息系统中运行情况良好,用户对其简单易学的操作较为满意,认为在一定程度上解决了一般 MIS 系统中需求反复的问题,使编程人员在系统运行后与菜单维护有关的工作量减少了大约 40%左右。

附注:菜单安装模块中主要算法的部分源代码删略,有兴趣者请与作者联系。