(一）树控制的主要功能

树控制和视（Tree Control&View)主要用来显示具有一定层次结构的数据项，如资源管理器中的磁盘目录等，以供用户在其中进行各种选择。

树控制中的每个数据项包括数据项名称的文本字符串和用于表示该数据项的图像，每个数据项下面均可包含各种子项，整个结构就象目录树一样。对于包含各种子项的数据项，可通过鼠标双击来展开或合拢，这可以通过控制树的不同风格来实现树控制的不同显示形态。这些风格主要包括：

TVS\_HASLINES表示用连线来连接父项和它下面的各个子项，这可以使树的显示层次结构更加清晰，但在无父项的各子项之间并没有连线；

TVS\_LINESATROOT表示在无父项的各子项即根下面的各子项之间存在连线；

TVS\_HASBUTTONS表示在带有子项的父项前面增加一个带“＋”或“－”的按钮，这使得用户也可以通过单击这个小按钮来实现子项的展开和合拢，当存在子项时，按钮的初始状态为“＋”，当子项被展开时，按小按钮由“＋”变为“－”号，当子项合拢时，小按钮由“－”变为“＋”号，这一风格同样对于根项无效，如果需要可通过组合TVS\_LINESATROOT风格来实现；

TVS\_EDITLABELS表示允许让用户单击具有输入焦点的数据项来修改其名称。

对于树控制，MFC中也以两种形式来封装，即树控件（CTREECTRL）和树视（CTREEVIEW），来满足用户的不同需求，对于一般要求的用户如在对话框中应用，使用树控件比较方便，而对于具有较高要求的用户，在使用树视时还具有视窗口的各种方便特性，可以更好地满足文档／视结构的要求。当在窗口中使用树视时，树视会占满两个窗口的客户区域并自动随窗口的框架结构的调整而调整，并能够很好地处理诸如菜单、加速键和工具条中的各种命令消息。在使用树视时只要利用其成员函数CTreeCtrl取得其一个引用，就可以象树控制一样方便地应用：CtreeCtrl

&treeCtrl = GetTreeCtrl()

（二）树控制的对象结构

1、树控制的建立方法 CtreeCtrl＆treeCtrl 建立树控制对象结构

Create 建立树控制并绑定对象，CTreeCtrl::Create的调用格式如下：

BOOL Create( DWORD dwStyle, const RECT& rect, CWnd\* pParentWnd, UINT nID );

其中参数dwStyle用来确定树控制的类型；

rect用来确定树控制的大小和位置；

pParentWnd用来确定树控制的父窗口，通用是一个对话框并且不能为NULL；

nID用来确定树控制的标识。树控制的风格可以是下列值的组合： TVS\_HASLINES 表示树控制在各子项之间存在连线； TVS\_LINESATROOT 表示树控制在根项之间存在连线；TVS\_HASBUTTONS 表示树控制视在父项左侧存在展开合拢控制按钮； TVS\_EDITLABELS 表示可以控制鼠标单击修改树项的名称； TVS\_SHOWSELALWAYS 表示选中项即使在窗口失去输入焦点时仍然保持选中状态； TVS\_DISABLEDRAGDROP表示禁止树控制发送TVN\_BEGINDRAG消息

2、树控制的属性类

树控制属性类包括取得树控制中项数GetCount、

取得树控制中项相对于父项的偏移值GetIndent、

取得树控制图像列表控制句柄GetImageList、

设置树控制图像列表控制句柄SetImageList、

取得匹配下一个树项GetNextItem、

判断给定树项是否包含子项ItemHasChildren、

取得树项子项GetChildItem

取得下一个同属树项GetNextSiblingItem、

取得前一个同属树项GetPrevSiblingItem、

取得父树项GetParentItem、

取得第一个可视树项GetFirstVisibleItem、

取得下一个可视树项GetNextVisibleItem、

取得前一个可视的树项GetPrevVisibleItem、

取得被选中的树项GetSelectedItem、

取得根树项GetRootItem、

取得树项的属性GetItem、

设置树项的属性SetItem、

取得树项的状态GetItemState、

设置树项的状态SetItemState、

取得与树项关联图像GetItemImage、

设置与树项关联图像SetItemImage、

取得树项文本GetItemText、

设置树项文本SetItemText和取得树项编辑控制句柄GetEditControl等。

3、树控制的操作方法

树控制的操作方法包括插入一个树项InsertItem、删除一个树项DeleteItem、删除所有树项DeleteAllItems、展开或合拢树项的子项Expand、选中特定树项SelectItem、选择一个树项作为第一个可视树项SelectSetFirstVisible、编辑一个可视的树项EditLabel和排序给定父树项的子树项SortChildren等。

三、树控制的数据结构

在使用树控制时需要了解两个个非常重要的数据结构TV\_ITEM和TV\_INSERTSTRUCT，前一个数据结构是用来表示树控制的树项信息，后一个数据结构是用来定义将树项增加到数据控制中所需要的数据内容。另外，还需要NM\_TREEVIEW、TV\_DISPINFO和TV\_HITTESTINFO三个数据结构，这几个数据结构的定义方法如下：

① 基本数据项结构

typedef struct \_TV\_ITEM

{

UINT mask; //结构成员有效性屏蔽位

HTREEITEM hItem; //数据项控制句柄

UINT state; //数据项状态

UINT stateMask; //状态有效性屏蔽位

LPSTR pszText; //数据项名称字符串

int cchTextMax; //数据项名称的最大长度

int iImage; //数据项图标索引号

int iSelectedImage;//选中数据项图标索引号

int cChildren; //子项标识

LPARAM lParam; //程序定义的位数据

} TV\_ITEM, FAR \*LPTV\_ITEM;

mask

指出其它的结构成员哪些包含有效数据的标记数组。当这个结构被TVM\_GETITEM消息使用时，mask成员指出项的属性被取回。这个成员可以是下列值的一个或多个。

TVIF\_CHILDREN

cChildren成员是有效的。

TVIF\_DI\_SETITEM

树形视控件将保留支持信息并且不重新请求它。当处理TVN\_GETDISPINF通知时，这个标记是有效的。

TVIF\_HANDLE

hItem成员有效。

TVIF\_IMAGE

iImage成员有效。

TVIF\_PARAM

lParam成员有效。

TVIF\_SELECTEDIMAGE

iSelectedImage成员有效。

TVIF\_STATE

state和stateMask成员有效。

TVIF\_TEXT

pszText和cchTextMax成员有效。

hItem

这个函数引用的项句柄。

state

位标记和图像列表索引的设置，指出项的状态。当设置了一个项的状态，stateMask成员指出这个成员的位是有效的。当取加一个项的状态时，这个成员返回stateMask成员指出的位的当前状态。

这个成员的0至7位包含了项的状态标记。关于可能的项状态标记，参见Tree

View Control Item States.

覆盖图像覆盖在项的图标图像之上。这个成员的8至11位指定了以1为基准的覆盖图像索引。如果这些位是0，这个项没有覆盖图像。要隔离这些位，使用TVIS\_OVERLAYMASK掩码。要在这个成员中设置覆盖图像索引，使用INDEXTOOVERLAYMASK宏。图像列表的覆盖图像是被ImageList\_SetOverlayImage函数设置的。

状态图像是仅次于项图标的用于指出项状态。通过发送TVM\_SETIMAGELIST消息来指定一个状态图像列表。要设置一个项的状态图像，在TVITEM结构的stateMask成员中包含TVIS\_STATEIMAGEMASK值。结构的state成员的12至15位指定状态图像列表中被绘制图像的索引。

要设置状态图像索引，使用INDEXTOSTATEIMAGEMASK。这个宏把一个索引适当的设置到12至15位上。要指出项没有状态图像，设置索引为0。这意味着在状态图像列表中的图像0不能被作为一个状态图像使用。要隔离state成员的位12至15，使用TVIS\_STATEIMAGEMASK掩码。

stateMask

state成员的位是有效的。如果你取回了一个项的状态，设置stateMask成员的位来指出state成员中的这个位被返回。如果你正在设置一个项的状态，设置stateMask成员的位来指出state成员的这个位是你想设置的。要设置或取回一个项的覆盖图像的索引，设置TVIS\_OVERLAYMASK位。要设置和取回一个项的状态图像索引，设置TVIS\_STATEIMAGEMASK位。

pszText

如果这个结构指定了项属性，那么这个成员是指向一个以空字符结束的字符串，包含有项的文本。如果这个成员是值LPSTR\_TEXTCALLBACK，那么父窗口为保存名字负责。既然这样，当树形视控件需要显示、保存或编辑项文本时，向父窗口发送TVN\_GETDISPINFO通过消息，当项文本改变时，发送TVN\_SETDISPINFO通知消息。

如果结构是取回项的属性，这个成员是取回项文本缓冲的地址。

cchTextMax

pszText成员指定缓冲的大小，以字符为单位。如果这个结构被使用来设置项属性，这个成员被忽略。

iImage

当项是在非选择状态中时，是树形控件的图像列表的索引。

如果这个成员是值I\_IMAGECALLBACK，父窗口为保存索引负责。既然这样，当树形视控件需要显示这个图像时，向父窗口发送TVN\_GETDISPINFO通知消息来获得索引。

iSelectedImage

当项被选择时，是树形控件图像列表的索引。

如果这个成员是值I\_IMAGECALLBACK，父窗口为保存索引负责。既然这样，当树形视控件需要显示这个图像时，向父窗口发送TVN\_GETDISPINFO通知消息来获得索引。

cChildren

标记指出哪一个项有关联的子项。这个成员可以是下列值之一。

zero

这个项没有子项。

one

这个项有一个或更多的子项。

I\_CHILDRENCALLBACK

The parent window keeps track of whether

the item has child items. In this case, when the tree view control needs to display the item, the control sends the parent a TVN\_GETDISPINFO notification message to determine whether the item has child items.

If the tree view control has the TVS\_HASBUTTONS style, it uses this member to determine whether to display the button indicating

the presence of child items. You can use this member to force the control to display the button even though the item does not have any child items inserted. This allows you to display the button while minimizing the control's memory usage by inserting child

items only when the item is visible or expanded.

lParam

与这项相关的32位值。

② 插入树项结构

typedef struct \_TV\_INSER TSTRUCT

{

HTREEITEM hParent; //父项控制句柄

HTREEITEM hInsertAfter; //插入树项的位置

TV\_ITEM item; //数据项的结构

} TV\_INSERTSTRUCT, FAR \*LPTV\_INSERTSTRUCT;

其中插入的位置如果是TVI\_FIRST 或TVI\_LAST ，则分别插入到树控制的最前面或最后面，如果是TVI\_SORT ，则插入的树项自动插入到合适的位置。

③ 树控制通知消息结构

typedef struct \_NM\_TREEVIEW

{

NMHDR hdr; //通知消息句柄

UINT action; //通知消息标志

TV\_ITEM itemOld; //原来的数据结构

TV\_ITEM itemNew; //新的数据结构

POINT ptDrag; //拖动指针

} NM\_TREEVIEW;

其中

typedef struct tagNMHDR {

HWND hwndFrom;//发送消息的控件句柄

UINT\_PTR idFrom;//控件ID

UINT code;//消息ID

} NMHDR;

④ 取得或设置数据结构

typedef struct \_TV\_DISPINFO

{

NMHDR hdr; //通知消息控制句柄

TV\_ITEM item; //数据项结构

} TV\_DISPINFO;

⑤指针测试数据结构 typedef

struct \_TVHITTESTINFO { tvhtst POINT pt; //客户区域屏幕坐标指针 UINT flags; //存放测试结果的变量 HTREEITEM hItem; //测试的数据项结构}

TV\_HITTESTINFO, FAR \*LPTV\_HITTESTINFO; 其中flags测试结果可以是如下值：TVHT\_ABOVE 在客户区域上面 TVHT\_BELOW 在客户区域下面 TVHT\_NOWHERE 在客户区域中并在最后一项下面 TVHT\_ONITEM 在与树项关联的位图或标签内 TVHT\_ONITEMBUTTON 在与树项关联的按钮上 TVHT\_ONITEMICON 在与树项关联的位图上 TVHT\_ONITEMINDENT 在与树项关联的联线上 TVHT\_ONITEMLABEL 在与树项关联的标签上 TVHT\_ONITEMRIGHT 在树项的右侧区域中TVHT\_ONITEMSTATEICON 在用户定义的状态图标上 TVHT\_TOLEFT 在客户区域的左侧TVHT\_TORIGHT 在客户区域的右侧

（四）树控制的应用技巧示例

这里仍以基于对话框演示实例来具体介绍树控制及其和图像列表相结构的应用技巧：

通过“FILE->NEW->PROJECTS->MFC AppWizard(EXE)”建立名为VCTREE的工程，在建立过程中选择基于对话框（Dialog

based）的应用；将对话框中的默认控件删除，并将所有对话框属性中的Language域设置为Chinese(P.R.C.),以使应用程序支持中文；

建立两个图标IDI\_PM和IDI\_CJ，用来表示图标的选中和非选中状态，对于每个图标都应建立32X32和16X16两种大小，以保证程序的需要；在对话框窗口中添加树控制对象（TREE

CONTROL）,并设置五个按钮“增加｜删除｜查看｜排序｜关闭”，其对应标识分别如下：

控制名称 标题名称 标识符号

树控制 IDC\_TREECTRL

按钮 增 加 IDC\_ADD

删 除 IDC\_DEL

查 看 IDC\_VIEW

排 序 IDC\_SORT

关闭 IDOK

5、选中树控制控件，选择“VIEW->ClassWizard->Memory

Variables。??骺刂艻DC\_TREECTRL 引入成员变量，其变量类型为：

变量名 种类 变量类型

m\_TreeCtrl Control CTreeCtrl

同时利用“MESSAGES MAP”为各命令按钮增加控制功能函数。

6、然后在代码文件VCTREEDlg.CPP中分别加入如下控制代码：

（1）在文件开始处增加图像列表定义 CImageList

Cil1,Cil2;//大小图标像列表

（2）在初始化文件开始处增加代码

BOOL CVCTREEDlg::OnInitDialog()

{

CDialog::OnInitDialog();

。。。。。

// TODO: 在此添加额外的初始化代码

CVCTREEApp \*pApp=(CVCTREEApp \*)AfxGetApp();//创建图象列表

Cil1.Create(16,16,ILC\_COLOR,2,2);

Cil1.Add(pApp->LoadIcon(IDI\_ICON1));

Cil1.Add(pApp->LoadIcon(IDI\_ICON3));

m\_TreeCtrl.SetImageList(&Cil1,TVSIL\_NORMAL); //设置图象列表

DWORD dwStyles=GetWindowLong(m\_TreeCtrl.m\_hWnd,GWL\_STYLE);//获取树控制原风格

dwStyles|=TVS\_EDITLABELS|TVS\_HASBUTTONS|TVS\_HASLINES|TVS\_LINESATROOT;

SetWindowLong(m\_TreeCtrl.m\_hWnd,GWL\_STYLE,dwStyles);//设置风格

char \* CJ[4]={"玉溪卷烟厂","云南卷烟厂","沈阳卷烟厂","成都卷烟厂"};//根数据名称

char \* PM[4][5]={ {"红梅一","红梅二","红梅三","红梅四","红梅五"},//产品数据项

{"白梅一","白梅二","白梅三","白梅四","白梅五"},

{"绿梅一","绿梅二","绿梅三","绿梅四","绿梅五"},

{"青梅一","青梅二","青梅三","青梅四","青梅五"}};

int i,j;

HTREEITEM hRoot,hCur;//树控制项目句柄

TV\_INSERTSTRUCT TCItem;//插入数据项数据结构

TCItem.hParent=TVI\_ROOT;//增加根项

TCItem.hInsertAfter=TVI\_LAST;//在最后项之后

TCItem.item.mask=TVIF\_TEXT|TVIF\_PARAM|TVIF\_IMAGE|TVIF\_SELECTEDIMAGE;//设屏蔽

TCItem.item.pszText="数据选择";

TCItem.item.lParam=0;//序号

TCItem.item.iImage=0;//正常图标

TCItem.item.iSelectedImage=1;//选中时图标

hRoot=m\_TreeCtrl.InsertItem(&TCItem);//返回根项句柄

for(i=0;i<4;i++){//增加各厂家

TCItem.hParent=hRoot;

TCItem.item.pszText=CJ[i];

TCItem.item.lParam=(i+1)\*10;//子项序号

hCur=m\_TreeCtrl.InsertItem(&TCItem);

for(j=0;j<5;j++){//增加各产品

TCItem.hParent=hCur;

TCItem.item.pszText=PM[i][j];

TCItem.item.lParam=(i+1)\*10+(j+1);//子项序号

m\_TreeCtrl.InsertItem(&TCItem);

}

m\_TreeCtrl.Expand(hCur,TVE\_EXPAND);//展开树

}

m\_TreeCtrl.Expand(hRoot,TVE\_EXPAND);//展开上一级树

return TRUE; // 除非将焦点设置到控件，否则返回TRUE

}

（3）增加树项功能的实现

在增加树项功能时，除了需要定义和设置插入树项的数据结构之外，还需要注意的是新增树项的名称初始时均为“新增数据”，增加后允许用户给数据项设置自定义名称。

在编程时应特别注意m\_TreeCtrl.EditLabel(hInsert);

后面不能跟任何其它程序命令，否则这条编辑指令无效。

void CVCTREEDlg::OnBnClickedAdd()

{

HTREEITEM hSel=m\_TreeCtrl.GetSelectedItem();//取得选择项句柄

if(hSel==NULL)

return;//无任何选项则返回

static int nAddNo=100;//编号大于为新增数据

TV\_INSERTSTRUCT TCItem;//定义插入项数据结构

TCItem.hParent=hSel; //设置父项句柄

TCItem.hInsertAfter=TVI\_LAST;//在最后增加

TCItem.item.mask=TVIF\_TEXT|TVIF\_PARAM|TVIF\_IMAGE|TVIF\_SELECTEDIMAGE;//设屏蔽

TCItem.item.pszText="新增数据";

TCItem.item.lParam=nAddNo++;//索引号增加

TCItem.item.iImage=0;//正常图标

TCItem.item.iSelectedImage=1;//选中时图标

HTREEITEM hInsert=m\_TreeCtrl.InsertItem(&TCItem);//增加

m\_TreeCtrl.Expand(hSel,TVE\_EXPAND);

m\_TreeCtrl.EditLabel(hInsert);//修改增加的数据

}

（4）删除树项功能的实现

在实现删除功能时，应对存在子项的树项进行提示，以警告用户是否连同其子项一起删除。

void CVCTREEDlg::OnDel()

{ //删除子项功能函数

HTREEITEM hSel=m\_TreeCtrl.GetSelectedItem();//取得选项句柄;

if(hSel==NULL)

return;//无任何选项则返回

if(m\_TreeCtrl.ItemHasChildren(hSel))//判断是否有子项

if(MessageBox("厂家下存在品名,一同删除?","警告",MB\_YESNO)==IDNO)

return;

m\_TreeCtrl.DeleteItem(hSel);

}

（5）完成节点编辑代码

void CVCTREEDlg::OnTvnEndlabeleditTreectrl(NMHDR \*pNMHDR, LRESULT \*pResult)

{

LPNMTVDISPINFO pTVDispInfo = reinterpret\_cast<LPNMTVDISPINFO>(pNMHDR);

// TODO: 在此添加控件通知处理程序代码

m\_TreeCtrl.SetItem(&pTVDispInfo->item);

\*pResult = 0;

}