[MFC总结之CListCtrl用法及技巧](https://www.cnblogs.com/renyuan/p/6425908.html)

**复制于：http://www.cnblogs.com/lidabo/archive/2012/08/23/2652796.html**

**https://www.cnblogs.com/renyuan/p/6425908.html**

**1、基本操作**

     分别从下面四点来介绍CListCtrl的基本操作：

**①设置列表视图显示方式**

**Ⅰ.**CListCtrl有四种样式：LVS\_ICON、LVS\_SMALLICON、LVS\_LIST、LSV\_REPORT，可通过控件属性来设置。本文所述均为LSV\_REPORT属性。

**Ⅱ.**扩展样式：

      常用的扩展样式有三种：LVS\_EX\_FULLROWSELECT、LVS\_EX\_GRIDLINES、LVS\_EX\_CHECKBOXES，分别对应作用 选中某行时使正行高亮、设置网格线、item前生成Ckeckbox控件。

      使用**SetExtendedStyle**(style)函数设置扩展样式，使用**GetExtendedStyle**()函数获取样式，如：

                       m\_listInfo.SetExtendedStyle(LVS\_EX\_FULLROWSELECT | LVS\_EX\_GRIDLINES);  
      **Ⅲ.**使用CListView时，需要在PreCreateWindow()函数中添加  cs.style | =  LVS\_REPORT;

来将其设置为LVS\_REPORT风格，否则插入无效。还用另一种方法来设置风格，即在OnInitialUpate()中获取CListCtrl控制权，然后修改风格，如下所示：

                      CListCtrl &theCtrl =**GetListCtrl**();

                      theCtrl.**ModifyStyle**(0, LVS\_REPORT);

**②插入操作**

先插入列：

int**InsertColumn**( int nCol, LPCTSTR lpszColumnHeading, int nFormat, int nWidth, int nSubItem)

插入列时，可指明列号、列名称、列名称显示样式，列宽等信息。对于列号为0的那一列，始终是靠左显示，后面会有修改使其剧中显示的方法，其他列通过设置nFormat属性可以居中显示。

插入行：

int **InsertItem**( int nItem, LPCTSTRlpszItem )

直接插入一行，nItem指明行号，lpszItem指明该行第0列的信息。

设置信息：

BOOL **SetItemText**(int nItem,  int nSubItem, LPCTSTR lpszText )

设置第nItem行nSubItem列的信息（nItem:0,1,2,3……； nSubItem:1,2,3……）

**③删除操作**

       有三个操作函数：

                       BOOL **DeleteAllItems**()  -------删除所有的行

                       BOOL **DeleteItem**(nItem) --------删除某一行

                       BOOL**DeleteColumn**(nCol) -----删除某一列

**④获取/设置属性函数**

     有很多函数了，就不一一介绍了。常用的有

                      int **GetItemCount**() -------- 获取已插入信息的行数

                      BOOL **SetItemState**(int iLink, UINTstate, UINTstateMask ) ---------设置行状态，如高亮显示等

等等

**2、获取选中行的行号**

       获取选中行的行号，然后对该行进行相关处理，这点在编程中用的非常多。

       当鼠标单击item时，控件向父窗口发送**NM\_CLICK消息**，其响应函数为OnNMClickXXXX(NMHDR \*pNMHDR, LRESULT \*pResult)，在该函数下来编写代码获取鼠标点击的行号。

       有两种方法来获取行号：第一种是使用**GetFirstSelectedItemPosition**和**GetNextSelectedItem**配合来获取；第二种是先获取鼠标位置信息，然后调用**HitTest**函数来找出行号。示例分别如下：

        第一种方法，该示例截自MSDN，可作修改后使用。

[复制代码](javascript:void(0);)

POSITION pos = pList->GetFirstSelectedItemPosition();

if (pos == NULL)

TRACE0("No items were selected!\n");

else

{

while (pos)

{

int nItem = pList->GetNextSelectedItem(pos);

TRACE1("Item %d was selected!\n", nItem);

// you could do your own processing on nItem here

}

}

[复制代码](javascript:void(0);)

第二种方法，该示例来自我的项目，可作修改后使用。

[复制代码](javascript:void(0);)

//获取单击所在的行号

//找出鼠标位置

DWORD dwPos = GetMessagePos();

CPoint point( LOWORD(dwPos), HIWORD(dwPos) );

m\_listCtrl.ScreenToClient(&point);

//定义结构体

LVHITTESTINFO lvinfo;

lvinfo.pt = point;

//获取行号信息

int nItem = m\_listCtrl.HitTest(&lvinfo);

if(nItem != -1)

m\_itemSel = lvinfo.iItem; //当前行号

[复制代码](javascript:void(0);)

对于LVHITTESTINFO 结构体，其有四个成员，在上述HitTest调用中，其第一个成员作为输入，另外三个作为输出。具体变量含义可查看MSDN。

[复制代码](javascript:void(0);)

typedef struct \_LVHITTESTINFO {

POINT pt;

UINT flags;

int iItem;

int iSubItem;

} LVHITTESTINFO, \*LPLVHITTESTINFO;

[复制代码](javascript:void(0);)

**3、复选框操作**

       有时需要在item前面添加一个CheckBox，供用户选择，然后对所有选中项进行处理。

       这里涉及到两个问题：第一个，如何添加CheckBox风格；第二个，如何判断某一行的CheckBox状态是否发生改变。

       对于第一个问题，在**基本操作**里已经有所阐述了，即通过SetExtendedStyle函数添加LVS\_EX\_CHECKBOXES扩展风格。

      这里重点探讨第二个问题，首先，操作复选框状态的有两个函数：

                      BOOL **GetCheck**(int nItem)-------获取复选框状态

                      BOOL **SetCheck**( int nItem, BOOL fCheck = TRUE )-------设置复选框状态

其次，我们要搞清楚以下四点：

**①**当列表的项item改变时，控件会向父窗口发送**LVN\_ITEMCHANGED**消息，因此可以在LVN\_ITEMCHANGED消息的响应函数中对复选框的状态进行处理（查询或设置）。

**②**鼠标点击CheckBox时，消息的顺序是 NM\_CLICK —> LVN\_ITEMCHANGED，即CheckBox的状态是在 NM\_CLICK消息函数结束后才会发生变化，在NM\_CLICK中使用GetCheck无效。

**③**鼠标点击Item（非CheckBox区域）时，消息的顺序是 LVN\_ITEMCHANGED —> NM\_CLICK。

**④** 调用InsertItem 函数时，也会产生LVN\_ITEMCHANGED消息。鉴于此，通常会自定义一个BOOL型变量m\_bHit 来判断是点击操作还是插入操作，该变量初始赋FALSE，当有鼠标点击item时赋TRUE, 检测完是否有CheckBox被点击后重新复位为FALSE。

        示例如下所示：

[复制代码](javascript:void(0);)

void CXXXX::OnNMClickXXXX(NMHDR \*pNMHDR, LRESULT \*pResult)

{

//获取单击所在的行号

//找出鼠标位置

DWORD dwPos = GetMessagePos();

CPoint point( LOWORD(dwPos), HIWORD(dwPos) );

m\_listCtrl.ScreenToClient(&point);

//定义结构体

LVHITTESTINFO lvinfo;

lvinfo.pt = point;

//获取行号信息

int nItem = m\_listCtrl.HitTest(&lvinfo);

if(nItem != -1)

m\_itemSel = lvinfo.iItem; //当前行号

//判断是否点击在CheckBox上

if(lvinfo.flags == LVHT\_ONITEMSTATEICON)

m\_bHit = TRUE;

\*pResult = 0;

}

void CXXXX::OnLvnItemchangedXXXX(NMHDR \*pNMHDR, LRESULT \*pResult)

{

LPNMLISTVIEW pNMLV = reinterpret\_cast<LPNMLISTVIEW>(pNMHDR);

//判断m\_bHit，即是否点击了CheckBox

if(m\_bHit)

{

m\_bHit = FALSE; //复位

if(m\_listCtrl.GetCheck(m\_itemSel))

{ //CheckBox被选中

//do your own processing

}

else

{ //CheckBox取消选择

//do your own processing

}

}

\*pResult = 0;

}

[复制代码](javascript:void(0);)

**4、动态设置选中行的字体颜色**

         有时可能需要设置某行的文字为特殊颜色，以表示某种特殊含义，比如正在下载的信息用绿色，暂停下载的用灰色。

         首先，给出一个CodeProject的链接，这篇文章讲的非常好，主要是利用Custom Draw。<http://www.codeproject.com/Articles/79/Neat-Stuff-to-Do-in-List-Controls-Using-Custom-Dra>

         然后，来谈谈我的方法，这里主要谈对选中行的字体颜色进行动态修改，当然也是我通过上面文章和自己实践结合得出的。

        我们需要搞清楚以下几点（可以结合下面修改某一行的字体颜色的方法来看）：

① 当控件绘制时，会发送**NM\_CUSTOMDRAW**消息，该消息的消息响应函数为

[复制代码](javascript:void(0);)

void CXXXX::OnNMCustomdrawXXXX(NMHDR \*pNMHDR, LRESULT \*pResult)

{

LPNMLVCUSTOMDRAW pLVCD = reinterpret\_cast<LPNMLVCUSTOMDRAW>(pNMHDR);

// TODO: Add your control notification handler code here

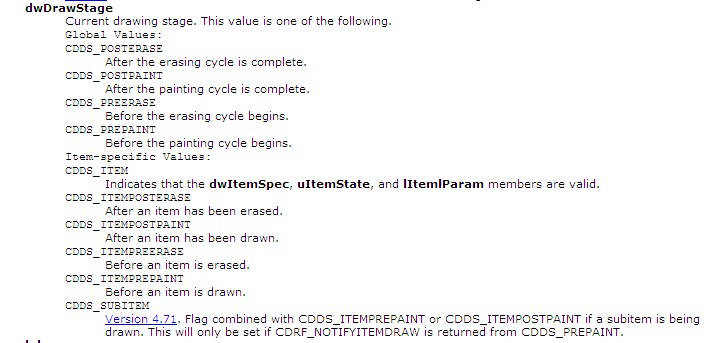
\*pResult = CDRF\_DODEFAULT;

//………………

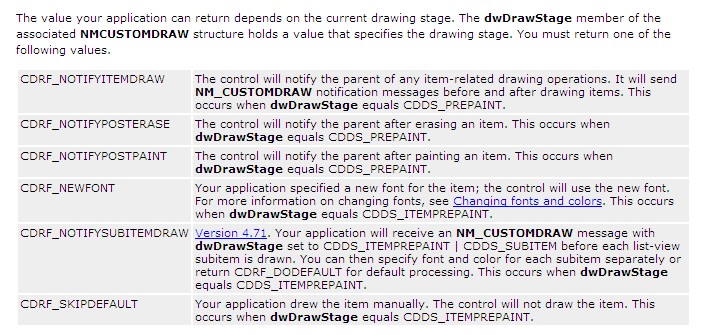
}

[复制代码](javascript:void(0);)

②其中，**pNMHDR为输入参数**，其指向**NMLVCUSTOMDRAW**结构体，该结构包含了很多信息，包括字体颜色、背景等等，特别是第一个成员，为**NMCUSTOMDRAW**结构体变量，其包含了**Current drawing stage**（不知道怎么编译比较好），其可能的值如下图（截自MSDN）所示



 ③ **pResult为输出参数**，该参数决定了接下来向windows发送什么消息（与绘制有关的），通过发送该消息我们可以进入下一步需要的处理阶段。具体输出哪个值取决于Current drawing stage，其可能的值如下图（截自MSDN）所示



④ 有一点必须注意(英文的，我觉得看起来比翻译过来更精确)：

     One thing to keep in mind is you must always **check the draw stage before doing anything else**, because your handler will receive many messages, and the draw stage determines what action your code takes.

        下面我们来看看如何修改某一行的字体颜色：

①  首先，我们应该明白要修改字体颜色，应该在**pre-paint 阶段**来完成

② 因此，在消息响应函数中，我们首先判断是否处于pre-paint stage（即pLVCD->nmcd.dwDrawStage == CDDS\_PREPAINT），然后通过修改输出值pResult 的值来通知windows我们需要处理每个item的消息（即设置 \*pResult = CDRF\_NOTIFYITEMDRAW）。

③ 再次进入消息响应函数时，我们判断是否处于Item的pre-paint stage（即pLVCD->nmcd.dwDrawStage == CDDS\_ITEMPREPAINT），如果是则进行相关处理，即修改字体颜色等等。

④ 处理完了后重新设置 \*pResult = CDRF\_DODEFAULT，表示我们不再需要其他特殊的消息了，默认执行即可。

         示例如下：

[复制代码](javascript:void(0);)

void CXXXX::OnNMCustomdrawXXXX(NMHDR \*pNMHDR, LRESULT \*pResult)

{

LPNMLVCUSTOMDRAW pLVCD = reinterpret\_cast<LPNMLVCUSTOMDRAW>(pNMHDR);

\*pResult = CDRF\_DODEFAULT;

// First thing - check the draw stage. If it's the control's pre-paint stage,

// then tell Windows we want messages for every item.

if ( CDDS\_PREPAINT == pLVCD->nmcd.dwDrawStage )

{

\*pResult = CDRF\_NOTIFYITEMDRAW;

}

else if ( CDDS\_ITEMPREPAINT == pLVCD->nmcd.dwDrawStage )

{

// This is the notification message for an item.

//处理，将item改变背景颜色

if( /\*符合条件\*/ )

pLVCD->clrText = RGB(255,0,255);

\*pResult = CDRF\_DODEFAULT;

}

}

[复制代码](javascript:void(0);)

上面谈的方法主要用于设置静态字体颜色，当然，如果你的列表的信息在不断变化（即用SetItemText不断修改），那么也就实现了动态改变了，否则需要在合适的地方调用重绘函数：

                         BOOL RedrawItems( int nFirst, int nLast )

表示在nFirst和nLast之间的行需要进行重绘。

**5、设置选中行的背景颜色**

         设置选中行的背景颜色，可以将选中行以特殊颜色显示，容易明白当前处理的是哪一行。尽管有高亮，但是高亮是基于焦点的，如果你选中了某一行，然后焦点转移了，这是就无法判断你选的是哪一行了。

        设置选中行的背景颜色的方法和第四节中讲的修改字体颜色的方法是相似的，都是利用Custom Draw。这里涉及到设置当前选中行为特殊颜色，同时要恢复前一次选中行的颜色，否则就乱了。因此需要记录前一次选中行、当前选中行的行号，相信通过前面的总结，这点并不难实现。然后在当前选中行和前一次选中行之间进行重绘即可。

       示例如下：

[复制代码](javascript:void(0);)

void CXXXX::OnNMClickXXXX(NMHDR \*pNMHDR, LRESULT \*pResult)

{

//…………

//重绘item，更改背景颜色

int nFirst = min(m\_itemSel,m\_itemForeSel);

int nLast = max(m\_itemSel,m\_itemForeSel);

m\_listCtrl.RedrawItems(nFirst, nLast); //在前一次选中的item和当前选中的Item之间进行重绘

\*pResult = 0;

}

void CXXXX::OnNMCustomdrawXXXX(NMHDR \*pNMHDR, LRESULT \*pResult)

{

LPNMLVCUSTOMDRAW pLVCD = reinterpret\_cast<LPNMLVCUSTOMDRAW>(pNMHDR);

\*pResult = CDRF\_DODEFAULT;

// First thing - check the draw stage. If it's the control's prepaint

// stage, then tell Windows we want messages for every item.

if ( CDDS\_PREPAINT == pLVCD->nmcd.dwDrawStage )

{

\*pResult = CDRF\_NOTIFYITEMDRAW;

}

else if ( CDDS\_ITEMPREPAINT == pLVCD->nmcd.dwDrawStage )

{

// This is the notification message for an item.

//处理，将item改变背景颜色

if(m\_itemSel == pLVCD->nmcd.dwItemSpec)

{ //当前选中的item

pLVCD->clrTextBk = RGB(255,0,0);

}

else if(m\_itemForeSel == pLVCD->nmcd.dwItemSpec)

{ //前一次选中的item，恢复为白色

pLVCD->clrTextBk = RGB(255,255,255);

}

\*pResult = CDRF\_DODEFAULT;

}

}

[复制代码](javascript:void(0);)

**6、禁止拖动表头**

       重载**OnNotify**消息响应函数，屏蔽两个消息通知码：**HDN\_BEGINTRACKW** 和**HDN\_DIVIDERDBLCLICKW**。示例如下：

[复制代码](javascript:void(0);)

BOOL CXXXX::OnNotify(WPARAM wParam, LPARAM lParam, LRESULT\* pResult)

{

// TODO: Add your specialized code here and/or call the base class

//屏蔽两个消息通知码，使得禁止拖动List表头

NMHEADER\* pNMHeader = (NMHEADER\*)lParam;

if(((pNMHeader->hdr.code == HDN\_BEGINTRACKW) |

(pNMHeader->hdr.code == HDN\_DIVIDERDBLCLICKW)))

{

\*pResult = TRUE;

return TRUE;

}

return CDialog::OnNotify(wParam, lParam, pResult);

}

[复制代码](javascript:void(0);)

**7、让第一列居中显示**

        在插入列时，我们可以通过参数nFormat来设置文本居中显示，但是这种设置对于第一列是没有作用的。这时我们可以考虑将我们的内容从第二列开始插入（设置为居中显示）。先插入第一列，然后删除第一列，这样原先的第二列就充当了第一列。

**8、设置行高和字体**

        设置CListCtrl的行高没有函数接口，可以通过自绘来实现，但是比较麻烦。有一个比较简单的方法是通过使用一个空白的图像将行撑起来，使其高度发生变化。示例如下：

CImageList m\_image;

m\_image.Create(1,24,ILC\_COLOR32,1,0);

m\_listInfo.SetImageList(&m\_image, LVSIL\_SMALL);

 对于字体的设置，我们可以使用**SetFont**函数来实现。以修改CListView的字体为例，在OnInitialUpdate函数中插入列之前调用SetFontSelf函数（该函数自定义，如下示例所示）。首先创建一个字体，然后调用SetFont进行设置。需要注意的是，在退出时需要delete 掉创建的字体，避免内存泄露。

[复制代码](javascript:void(0);)

//设置字体和大小

void CMyListView::SetFontSelf(int nHeight, LPCTSTR lpszFacename)

{

//先删除原有字体

if(m\_font != NULL)

delete m\_font;

m\_font = new CFont;

//创建字体

m\_font->CreateFont(

nHeight, // nHeight

0, // nWidth

0, // nEscapement

0, // nOrientation

FW\_NORMAL, // nWeight

FALSE, // bItalic

FALSE, // bUnderline

0, // cStrikeOut

ANSI\_CHARSET, // nCharSet

OUT\_DEFAULT\_PRECIS, // nOutPrecision

CLIP\_DEFAULT\_PRECIS, // nClipPrecision

DEFAULT\_QUALITY, // nQuality

DEFAULT\_PITCH | FF\_SWISS, // nPitchAndFamily

lpszFacename); // lpszFacename

//设置字体

CListCtrl &theCtrl = GetListCtrl(); //获取控制权，引用变量

theCtrl.SetFont(m\_font, TRUE);

}

[复制代码](javascript:void(0);)

**9、虚拟列表技术**

        给一个链接，介绍的比较详细：<http://hi.baidu.com/qi_xian/blog/item/929b04ce27d02c0592457ef8.html>

       当数据量大时，使用InsertItem插入数据的过程是很漫长的。这时我们有两个方法来解决该问题：一是使用CListCtrl的虚拟列表技术，二是采用分页显示的方法。对于虚拟列表技术，上述链接中的文章讲的很详细，我用过它的比较简单的方法，后来改用了分页方法。

       使用虚拟列表技术，有三点需要搞清楚：

**①** 使用虚拟技术时，需要将CListCtrl控件的Owner Data属性设置为ture。

**②** 给虚拟列表添加元素时，不需要使用InserItem函数，通过调用SetItemCount设置数据总个数，然后由系统产生不同的消息，在相应的消息响应函数中完成插入工作。

**③** 虚拟列表向父窗口发送的消息有三种： ⑴ 当它需要数据时，发送**LVN\_GETDISPINFO**消息； ⑵ 当用户试图查找某个元素时，发送**LVN\_ODFINDITEM**消息； ⑶当需要缓冲数据时，发送 **LVN\_ODCACHEHINT**消息。

        当我们使用LVN\_GETDISPINFO 的消息处理函数来插入元素时，必须首先检查列表请求的是什么数据（如LVIF\_TEXT、LVIF\_IMAGE等），然后插入不同的子项。示例如下：

[复制代码](javascript:void(0);)

void CDataAnalysis::OnLvnGetdispinfoAnalysisList(NMHDR \*pNMHDR, LRESULT \*pResult)

{

NMLVDISPINFO \*pDispInfo = reinterpret\_cast<NMLVDISPINFO\*>(pNMHDR);

// TODO: Add your control notification handler code here

LV\_ITEM\* pItem= &(pDispInfo)->item;

int iItemIndex= pItem->iItem;

size\_t converted = 0;

wchar\_t wStr[30]; //Unicode字符串

if (pItem->mask & LVIF\_TEXT) //字符串缓冲区有效

{

switch(pItem->iSubItem)

{

case 0: //填充数据项的名字,xxxxx表示要填充的字符

mbstowcs\_s(&converted, wStr, 30, xxxxxx, \_TRUNCATE);

lstrcpy(pItem->pszText,wStr);

break;

case 1: //填充子项1

mbstowcs\_s(&converted, wStr, 30, xxxxxx, \_TRUNCATE);

lstrcpy(pItem->pszText,wStr);

break;

case 2: //填充子项2

mbstowcs\_s(&converted, wStr, 30, xxxxxx, \_TRUNCATE);

lstrcpy(pItem->pszText,wStr);

break;

case 3: //填充子项3

lstrcpy(pItem->pszText,xxxxxx);

break;

}

}

\*pResult = 0;

}

[复制代码](javascript:void(0);)

**10、点击表头时进行归类排序**

         系统通过发送**LVM\_SORTITEMS**消息来处理归类问题，在该消息的处理函数中需要调用一个**回调函数**，这个回调函数需要我们来设计，以完成不同的归类方法。回调函数原型如下：

**int CALLBACK CompareFunc(LPARAM lParam1, LPARAM lParam2, LPARAM lParamSort)**

          针对上述回调函数，有以下几点需要搞清楚：

**①** 对于参数lparam1和lparam2，分别为CListCtrl的两行数据，是用于比较的对象。通过CListCtrl的成员函数**SetItemData**来设置，该函数原型：

**int SetItemData(int nIndex,  DWORD\_PTR dwItemData )**

其第一个参数为行号，第二个参数指明了该行对应的参数。参数dwItemData 通常设为一行参数的数组，如: pData[2][2] = {{1, 3},{2, 3}}; 每次使用pData[i]作为dwItemData。

**②** 对于参数lParamSort，用于指明列项，即第几列。该参数和回调函数一同通过CListCtrl的成员函数**SortItems**来设置，其函数原型为：

**BOOL SortItems( PFNLVCOMPARE pfnCompare,DWORD\_PTR dwData )**

参数 pfnCompare 为回调函数入口地址， 参数dwData 为列项。

**③ SetItemData**在初始插入数据时进行调用来设置，**SortItems**则在点击列表头时响应的消息处理函数中进行设置。

示例如下：

[复制代码](javascript:void(0);)

//初始化列表视图控件

BOOL CDataAnalysis::InitListCtl()

{

//其他处理，包括设置风格，插入列等等

//插入行

for(int i=0; i<LineNum; i++)

{

//要将char\*转换为wchar\_t\*

mbstowcs\_s(&converted, wStr, 30, m\_analysis[i].Date, \_TRUNCATE);

m\_listAnalysis.InsertItem(i, wStr); //日期

mbstowcs\_s(&converted, wStr, 30, m\_analysis[i].Time, \_TRUNCATE);

m\_listAnalysis.SetItemText(i, 1, wStr); //时间

mbstowcs\_s(&converted, wStr, 30, m\_analysis[i].ID, \_TRUNCATE);

m\_listAnalysis.SetItemText(i, 2, wStr); //ID

m\_listAnalysis.SetItemText(i, 3, m\_analysis[i].lpszEvent); //事件

//设置回调函数的参数

m\_listAnalysis.SetItemData(i, (LPARAM)(m\_analysis+i));

}

return TRUE;

}

void CDataAnalysis::OnHdnItemclickAnalysisList(NMHDR \*pNMHDR, LRESULT \*pResult)

{

LPNMHEADER phdr = reinterpret\_cast<LPNMHEADER>(pNMHDR);

// TODO: Add your control notification handler code here

//设置回调函数的参数和入口地址

m\_listAnalysis.SortItems(SortFunc, phdr->iItem);

\*pResult = 0;

}

//排序的回调函数

int CALLBACK SortFunc(LPARAM lParam1, LPARAM lParam2, LPARAM lParamSort)

{

int result; //返回值

//两行的参数，用于比较

ANALYSISFORMAT\* pAnalysis1 = (ANALYSISFORMAT\*)lParam1;

ANALYSISFORMAT\* pAnalysis2 = (ANALYSISFORMAT\*)lParam2;

//排序

switch(lParamSort)

{

case 0: //日期

result = strcmp(pAnalysis1->Date, pAnalysis2->Date);

break;

case 1: //时间

result = strcmp(pAnalysis1->Time, pAnalysis2->Time);

break;

case 2: //ID

result = strcmp(pAnalysis1->ID, pAnalysis2->ID);

break;

case 3: //事件

result = wcscmp(pAnalysis1->lpszEvent, pAnalysis2->lpszEvent);

break;

default:

break;

}

return result;

}

[复制代码](javascript:void(0);)

**11、向上与向下移动**

        有时需要向上或向下移动表项内容，这里给出向上移动的方法，向下移动的方法类似。

① 利用第2节所述的内容获取行号nItem，判断行号是否为行首，如果不是行首则进入②；

② 获取第nItem行的所有子项内容；

③ 删除第nItem行，并在nItem-1的位置重新插入原先的第nItem行的内容；

④ 使nItem-1的位置高亮显示

示例如下：

[复制代码](javascript:void(0);)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*上移子项\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CStudyTestDlg::OnPageup()

{

if (nItem == 0)

{

MessageBox("该子项已经位于第一行!");

return;

}

// 提取内容

CString temp[4];

int i;

for(i=0;i<4;i++)

temp[i] = m\_ListCtrl.GetItemText(nItem, i);

// 删除

m\_ListCtrl.DeleteItem(nItem);

// 在nItem-1位置处插入

for (i=0; i<4; i++)

m\_ListCtrl.SetItemText(nItem-1,i,temp[i]);

//高亮显示

UINT flag = LVIS\_SELECTED|LVIS\_FOCUSED;

m\_ListCtrl.SetItemState(--nItem, flag, flag);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*下移子项\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CStudyTestDlg::OnPagedown()

{

if (nItem == m\_ListCtrl.GetItemCount()-1)

{

MessageBox("该子项已经位于最后一行!");

return;

}

// 提取内容

CString temp[4];

int i;

for (i=0; i<4; i++)

temp[i] = m\_ListCtrl.GetItemText(nItem, i);

// 删除

m\_ListCtrl.DeleteItem(nItem);

// 在nItem+1位置处插入

for (i=0; i<4; i++)

m\_ListCtrl.SetItemText(nItem+1, i,temp[i]);

//高亮显示

UINT flag = LVIS\_SELECTED|LVIS\_FOCUSED;

m\_ListCtrl.SetItemState(++nItem, flag, flag);

}

[复制代码](javascript:void(0);)

**12、避免闪烁问题**

           这个问题在我的前面一篇博文有提到。

<http://blog.csdn.net/zwgdft/article/details/7394318>

**13、动态调整大小**

        有时由于不确定软件运行时的电脑屏幕大小，需要根据屏幕大小动态设置CListCtrl控件的大小。动态大小的设置时，需要注意不要将高度和宽度设置的超过区域限制，否则就没有滚动条了，导致部分内容无法查看。以我遇到的一个例子来说，其情况见第12节提到的那篇博文所述：将View划分为三个窗格，在左上角View上有个CPropertySheet，其上有几个CPropertyPage，每个属性页上有个CListCtrl，供用户查看信息。那么这时需要设置的CListCtrl的大小即为：

                                              宽度 = 左上角View宽度

                                              高度 = 左上角View高度 - 属性页的Tab项高度

调用**MoveWindow**函数进行设置即可。

分类: [Visual C++](https://www.cnblogs.com/renyuan/category/393539.html)