Environment: VC6, VC7, Windows 9x/NT/2000/XP

In the given article, practically all operations on moving a group of files from one application to another are described. Considered are both the reception of files by the given application and transfer of files from the given application in other programs. In the article, the operations drag and drop, cut, copy, paste are described. The description is constructed using the demonstration project.

While developing the program [Mp3 Music Explorer](http://www.brigsoft.com/mp3explorer), I have confronted the fact that in the documentation from Microsoft, and also on the known sites devoted to development of the software, there is no example in which all necessary operations on moving files would be presented. Therefore, I have created the demonstration project FilesDragDrop. It is based on the MFC SDI. For the files display, the class CListView is used.

[*Click here for a larger image.*](https://www.codeguru.com/cpp/cpp/cpp_mfc/tutorials/article.php/c4145/FilesDragDrop.gif)

Receiving the drop of the list of files in the application is possible by two methods: using the message WM\_DROPFILES and using the mechanism Ole. Both methods, as well transferring a file through the Clipboard, use one structure of file description DROPFILES. To drag a file from the application, only the mechanism Ole is used. Therefore, I implemented all operations of moving files by using the Ole technology.

**1. Initialization**

The realization of moving files is possible without such concepts as Ole Server and Ole Client. They are excessive for this task. It is possible to use the usual MFC project. But, for the necessary Ole classes to work, we need to initialize the Ole library. For this purpose, the following lines are inserted into the function CFilesDragDropApp::

1. InitInstance ():
3. // Initialize OLE 2.0 libraries
4. if (!AfxOleInit())
5. {
6. AfxMessageBox("AfxOleInit Error!");
7. return FALSE;
8. }

**2. Receiving the Files in the Application**

To receiving files using the Drag&Drop technology, I used OnDragOver and OnDrop notifications.

In the OnDragOver function, I determine whether the list of files or something else is carried above a window, and I return the assumed result of the operation. If it is not the files, the returned value is DROPEFFECT\_NONE. It allows the system to establish the correct mouse cursor.

In the OnDrop function, the extraction of files and the clearing of the buffer are made. For these functions to work, the window should be registered as the handler of the Drag&Drop operation. For the registration, the object such as COleDropTarget is used as follows:

1. void CFilesDragDropView::OnInitialUpdate()
2. {
3. .
4. VERIFY( m\_DropTarget.Register(this) );
5. .
6. }

Clipboard and Drag&Drop use the same structure for data transfer. In processing the messages OnPaste and OnDrop, there is an extraction of the list of files from the object COleDataObject:

1. void CFilesDragDropView::OnEditPaste()
2. {
3. COleDataObject DataObject;
4. if( DataObject.AttachClipboard() )
5. DataObjectToList(&DataObject);
6. }
7. }
9. BOOL CFilesDragDropView::OnDrop(COleDataObject\* pDataObject,
10. DROPEFFECT dropEffect,
11. CPoint point)
12. {
14. BOOL bRet = DataObjectToList(pDataObject);
15. .
17. }

The data type used for transferring the files is CF\_HDROP. The data are passed through the global memory, its handle. It is possible to take from COleDataObject by using the GetData function. The amount of transferred files and paths to them can be received with the help of a handle by using the DragQueryFile ( CFilesDragDropView:: FileNamesToList) function. The received files are displayed in a ListView window.

**3. Transferring Files from the Application**

To transfer files from the application using Copy/Paste and Drag&Drop technologies, it is necessary to create a DROPFILES structure. This structure corresponds to a data type CF\_HDROP of the exchange buffer. The paths to files in the structure are separated from each other by the '\0' symbol; the end of the list of files is marked by two symbols: '\0 '. There is a standard function of path extraction from the DragQueryFile buffer, but there is no standard function for creating a buffer. For this purpose, I have designed a CDropFiles class. The class is used in the following way: 1) first, the AddFile function enters all the paths of transferred files, 2) then the CreateBuffer function creates a necessary data structure. For access to the structure, the GetBuffer and GetBuffSize functions are used.

When using the Copy or Cut command, the data generated in CDropFile:: m\_pBuff are entered in the exchange buffer between the applications by the SetClipboardData function.

In Drag&Drop technology, for the transfer of files from the application the function OnBeginDrag is used. It is called by the LVN\_BEGINDRAG message of the CListCtrl class. If we use a window that does not generate the similar message, it is possible to use the WM\_LBUTTONDOWN message. The data for the transfer are entered in COleDataSource object by the CacheGlobalData function; the data transfer is carried out by the DoDragDrop function. For the object COleDataSource to work, it is necessary to create a COleDropSource object. This object is not used in any way in the transfer. All necessary actions are carried out in the constructor.

Memory of the exchange buffer is emptied by the receiving party.

The text of the function follows:

1. void CFilesDragDropView::OnBeginDrag(NMHDR\* pNMHDR,
2. LRESULT\* pResult)
3. {
4. NM\_LISTVIEW\* pNMListView = (NM\_LISTVIEW\*)pNMHDR;
6. CDropFiles DropFiles; // My class for creating a DROPFILES
7. // struct
9. if(!PrepareFileBuff(DropFiles)){
10. ASSERT(0);
11. }
13. COleDropSource DropSource;
14. COleDataSource DropData;
16. HGLOBAL hMem = ::GlobalAlloc(GMEM\_ZEROINIT|GMEM\_MOVEABLE|
17. GMEM\_DDESHARE,
18. DropFiles.GetBuffSize());
19. memcpy( (char\*)::GlobalLock(hMem), DropFiles.GetBuffer(),
20. DropFiles.GetBuffSize() );
22. ::GlobalUnlock(hMem);
24. DropData.CacheGlobalData( CF\_HDROP, hMem );
25. DROPEFFECT de = DropData.DoDragDrop(DROPEFFECT\_COPY|
26. DROPEFFECT\_MOVE,NULL);
28. if(de == DROPEFFECT\_COPY){
29. // Copy files if CTRL button is held;
30. }
31. else{
32. // Move files, as default;
33. DeleteSelectedFiles();
34. }
36. \*pResult = 0;
37. }

**4. Testing the Demonstration Project**

If you transfer files from any catalogue to the FilesDragDrop application, and then from the application to another catalogue, the files really will be transferred. That's why, when testing, I recommend using the special temporary catalogues to avoid the transfer of necessary files.

**Links**

[http://www.brigsoft.com/mp3explorer](http://www.brigsoft.com/bsalarm)—the technology described in the given article, was applied in the Mp3 Music Explorer product.

[http://www.brigsoft.com/edu](http://www.brigsoft.com/colorch)—the author's other articles and sources.

© Alex Rest, 2003

**Downloads**

[Download demo project - 23 Kb](https://www.codeguru.com/code/legacy/cpp_mfc/FilesDragDrop_demo.zip)