Shell

Windows programming

In this talk

- What is shell?
- Shell extension

- F2 / Del
- Splitter
- Shell – FindFirstFile / FindNextFile
- Status
- Ini (Save /load screen size)

What is shell?

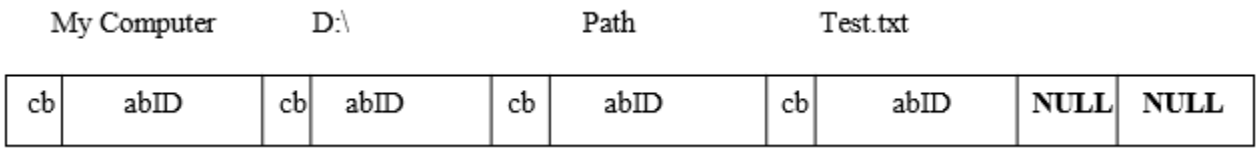
- Every object in a folder has an ID

```
typedef struct _SHITEMID {
        USHORT cb;
        BYTE   abID[1];
} SHITEMID, * LPSHITEMID;
```

- Most of the time, it is part of a list

```
typedef struct _ITEMIDLIST{
        SHITEMID  mkid;
} ITEMIDLIST;
```

- PIDL: Pointer to ID list PIDL, ends with 2 '\0'



- for (int i = 0; i < 10; i++) {
- }

- Int I = 0;

- P = ptu dautien;
- While (p->pNext != 0) {
- // XL
  - P = p->pNext;
- }

Two types of PIDL

- Absolute
  - Desktop.My Computer.C:.Windows
- Relative
  - 1 tier: \Relax
  - Multi tier:
    \Relax\Music
    \Relax\Videos

Traversal

```
void Traverse(LPITEMIDLIST pidl){
	while (pidl->mkid.cb != 0){
		//Do stuff here

		// Go to next item
		pidl = (LPITEMIDLIST) ( ((LPBYTE) pidl) + pidl->mkid.cb )
	}
}
```

CSIDL

- Constant special item ID list
- Environment variables
  - %ProgramFiles% => **CSIDL_COMMON_PROGRAMS**
  - %WinDir% => **CSIDL_WINDOWS**
  - %System% => **CSIDL_SYSTEM**

- **CSIDL_DESKTOP**: Root of shell namespace
  - **CSIDL_DESKTOPDIRECTORY**: depend on user !
- **CSIDL_DRIVES**: My computer / This PC

Working with folder

- Main interface to work with: **IShellFolder**

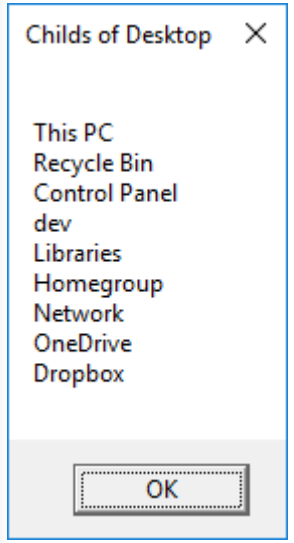
- From PIDL to folder path:
  - **SHGetPathFromIDList**
  - **GetDisplayNameOf()**
- From path to PIDL
  - **ParseDisplayName**

Folder traversal

```
LPSHELLFOLDER psfDesktop = NULL;
SHGetDesktopFolder(&psfDesktop); //Lay Desktop

LPENUMIDLIST penumIDL = NULL;  // Lay enumerator
psfDesktop->EnumObjects(NULL, SHCONTF_FOLDERS, &penumIDL);

LPITEMIDLIST pidl = NULL; //PIDL dùng để duyệt
HRESULT hr = NULL;

do {
    hr = penumIDL->Next(1, &pidl, NULL);
    if (hr == S_OK){
        //Xử lí pidl ở đây

        pMalloc->Free(pidl);
    }
} while(hr == S_OK);

pMalloc->Release(); //Giải phóng giao diện IMalloc
psfDesktop->Release(); //Giải phóng giao diện IShellFolder
```

Desktop traversal example

```
WCHAR info[1024];
info[0] = '\0';

do {
    hr = penumIDL->Next(1, &pidl, NULL);

    if (hr == S_OK){
        WCHAR buffer[1024];
        STRRET strret;
        psfDesktop->GetDisplayNameOf(pidl, SHGDN_NORMAL, &strret);
        StrRetToBuf(&strret, pidl, buffer, 1024);
        StrCat(info, buffer);
        StrCat(info, L"\n");

        pMalloc->Free(pidl);
    }
} while(hr == S_OK);

MessageBox(0, info, L"Childs of Desktop", 0);
```



- 260
- 10.240

cuu duong than cong . com

SHFileOperation

- Move, Copy, Delete, Rename File & Folder
- SHFILEOPSTRUCT
- **SHChangeNotify**: let shell know what changes


```
SHFILEOPSTRUCT sfo;
sfo.hwnd = NULL;
sfo.wFunc = FO_COPY;
sfo.pFrom = _T("C:\\test.txt\0");
sfo.pTo = _T("C:\\Windows\0");
sfo.fFlags = FOF_SILENT | FOF_NOCONFIRMATION;

SHFileOperation(&sfo);

SHChangeNotify(SHCNE_UPDATEDIR, SHCNF_PATH, (LPCVOID) _T("c:\\Windows"), 0);
```


- **ShellExecuteEx**: Run an program

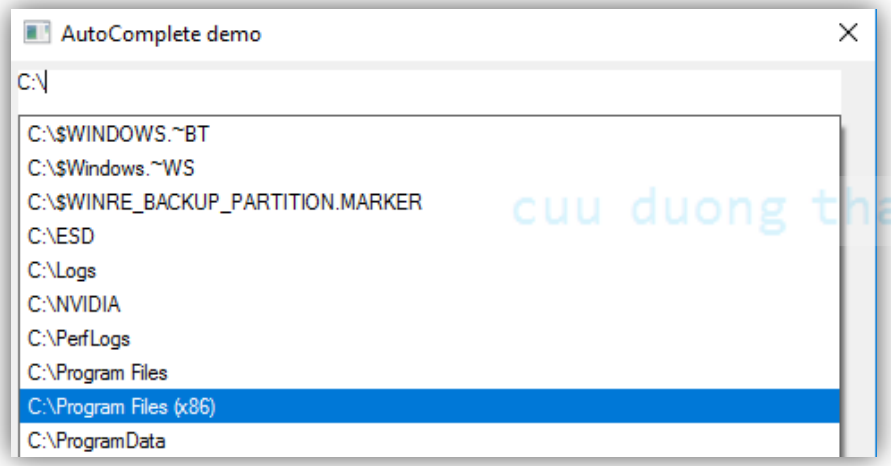
BrowseForFolder

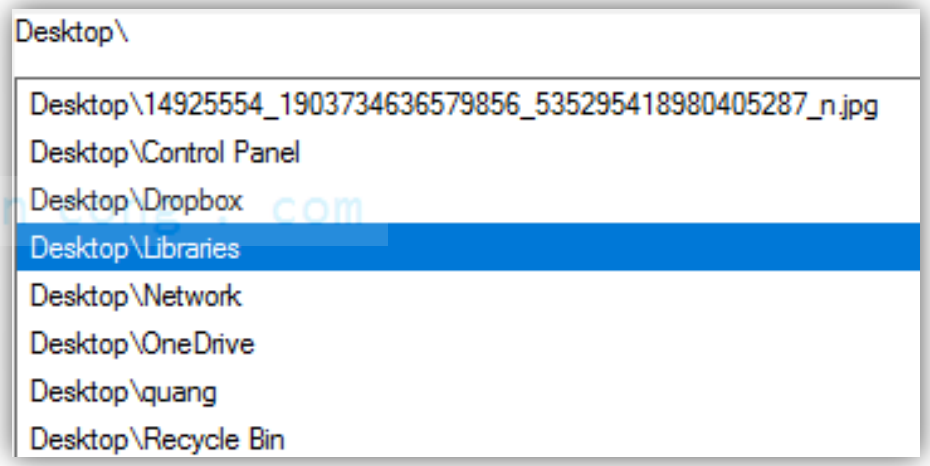
```
LPITEMIDLIST PidlBrowse(HWND hwnd, int nCSIDL, LPWSTR pszDisplayName) {
    LPMALLOC pMalloc = NULL;
    SHGetMalloc(&pMalloc); //Lay Allocator cua IMalloc
    LPITEMIDLIST pidlRoot = NULL;
    SHGetFolderLocation(hwnd, nCSIDL, NULL, NULL, &pidlRoot);

    BROWSEINFO bi = {0}; // ZeroMemory
    bi.hwndOwner = hwnd;
    bi.pidlRoot = pidlRoot;//Goc cua hop thoai o dau
    bi.pszDisplayName = pszDisplayName;
    bi.lpszTitle = L"Tiêu đề";
    bi.ulFlags = BIF_USENEWUI
        // | BIF_NONEWFOLDERBUTTON
        // | BIF_BROWSEINCLUDEFILES
        ;
    bi.lpfn = NULL;     bi.lParam = 0;

    LPITEMIDLIST pidlSelected = NULL;
    pidlSelected = SHBrowseForFolder(&bi); //Hien thi hop thoai

    if (pidlRoot != NULL) pMalloc->Free(&pidlRoot);
    pMalloc->Release();//Giai phong giao dien IMalloc
    return pidlSelected;
}
```

- GetSystemImageList

Autocomplete

```
SHAutoComplete(hEdit,
    SHACF_FILESYSTEM // All shell objects
    // SHACF_FILESYS_ONLY // Chỉ tập tin
    // SHACF_FILESYS_DIRS // Tập tin và thư mục
);
```



Component Object Model

- Platform-independent, distributed, object-oriented system for creating binary software components that can interact
- Foundation for
  - OLE (compound documents)
  - ActiveX Object (Internet-enabled components)

https://msdn.microsoft.com/en-us/library/windows/desktop/ms680573(v=vs.85).aspx

A standard, not OOP language

- Specify only
  - Object model
  - Programming requirements


```
class CCommandHandler : public IUICommandHandler
{
public:
    // IUnknown methods.
    STDMETHODIMP_(ULONG) AddRef();
    STDMETHODIMP_(ULONG) Release();
    STDMETHODIMP QueryInterface(REFIID iid, void** ppv);
```

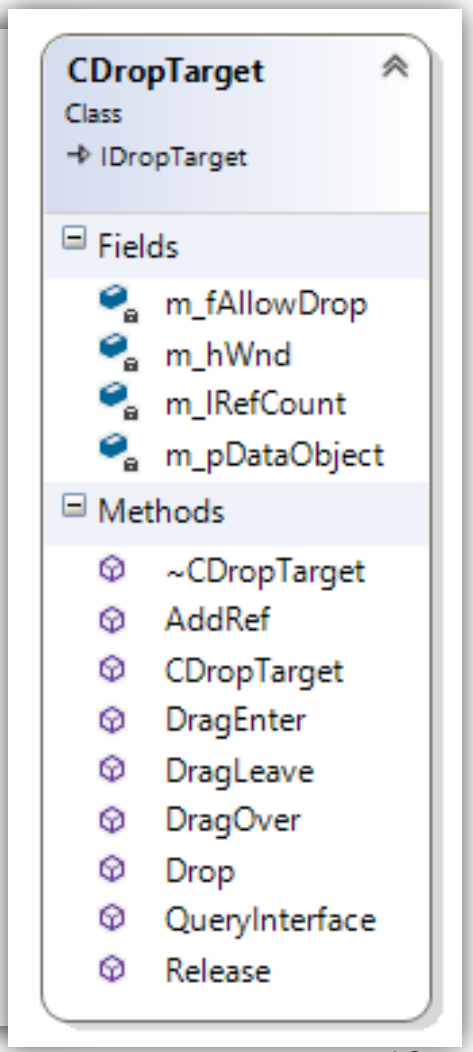
IUnknown

Lookback – Ribbon CommandHandler

```
class CCommandHandler : public IUICommandHandler // Command handlers mus
{
public:
    // Static method to create an instance of the object.
    static HRESULT CreateInstance(IUICommandHandler **ppCommandHandler);

    // IUnknown methods.
    STDMETHODIMP_(ULONG) AddRef();
    STDMETHODIMP_(ULONG) Release();
    STDMETHODIMP QueryInterface(REFIID iid, void** ppv);

    // IUICommandHandler methods
    STDMETHOD(UpdateProperty)(UINT nCmdID,
        REFPROPERTYKEY key,
        const PROPVARIANT* ppropvarCurrentValue,
        PROPVARIANT* ppropvarNewValue);

    STDMETHOD(Execute)(UINT nCmdID,
        UI_EXECUTIONVERB verb,
        const PROPERTYKEY* key,
        const PROPVARIANT* ppropvarValue,
        IUISimplePropertySet* pCommandExecutionProperties);

private:
    CCommandHandler() : m_cRef(1) {}
    LONG m_cRef;                        // Reference count.
};
```

Drag & drop

```
HRESULT __stdcall CDropTarget::Drop(IDataObject * pDataObject,
    DWORD grfKeyState, POINTL pt, DWORD * pdwEffect) {
    FORMATETC fmtetc = { CF_HDROP, 0, DVASPECT_CONTENT, -1, TYMED_HGLOBAL
    STGMEDIUM stgmed;
    WCHAR szFileName[10240] = {0};

    if(pDataObject->QueryGetData(&fmtetc) == S_OK)
        if(pDataObject->GetData(&fmtetc, &stgmed) == S_OK)  {
            HDROP hdrop = (HDROP) GlobalLock(stgmed.hGlobal);
            UINT uNumFiles = DragQueryFile ( hdrop, -1, NULL, 0 );

            for (UINT i = 0; i < uNumFiles; ++i) {
                DragQueryFile(hdrop, i, szFileName, 10240);
                MessageBox(0, szFileName, L"Path", MB_OK);
            }

            GlobalUnlock(stgmed.hGlobal);
            ReleaseStgMedium(&stgmed);
        }

    return S_OK;
}
```



SGTMEDIUM: IAdviseSink, IDataObject, IOleCache

Iunknown implementation

```
HRESULT __stdcall CDropTarget::QueryInterface(
    REFIID iid, void ** ppvObject) {
    if(iid == IID_IDropTarget || iid == IID_IUnknown) {
        AddRef();
        *ppvObject = this;
        return S_OK;
    } else {
        *ppvObject = 0;
        return E_NOINTERFACE;
    }
}
```

```
ULONG __stdcall CDropTarget::AddRef(void) {
    return InterlockedIncrement(&m_lRefCount);
}

ULONG __stdcall CDropTarget::Release(void) {
    LONG count = InterlockedDecrement(&m_lRefCount);

    if(count == 0) {
        delete this;
        return 0;
    } else {
        return count;
    }
}
```

Link creation

```
void CreateLink() {
    IShellLink* pLink;

    // Get a pointer to the IShellLink interface.
    // It is assumed that CoInitialize has already been called.
    HRESULT hr = CoCreateInstance(CLSID_ShellLink, NULL,
        CLSCTX_INPROC_SERVER, IID_IShellLink, (LPVOID*) &pLink);

    pLink->SetPath(L"C:\\Windows"); // shortcut target
    pLink->SetDescription(L"A sample of link creation");

    // Query IShellLink for the IPersistFile interface, used for saving the
    // shortcut in persistent storage.
    IPersistFile* pStorage;
    pLink->QueryInterface(IID_IPersistFile, (LPVOID*) &pStorage);
    pStorage->Save(L"E:\\Test.lnk", TRUE);

    MessageBox(0, L"Link created successfully!", L"Success", MB_OK);

    pStorage->Release();
    pLink->Release();
}
```

Link resolution

```
void ResolveLink() {
    IShellLink* pLink;
    HRESULT hr = CoCreateInstance(CLSID_ShellLink, NULL,
        CLSCTX_INPROC_SERVER, IID_IShellLink, (LPVOID*) &pLink);

    IPersistFile* pStorage;
    pLink->QueryInterface(IID_IPersistFile, (LPVOID*)&pStorage);
    pStorage->Load(L"E:\\Test.lnk", STGM_READ);

    WCHAR buffer[1024];
    pLink->GetPath(buffer, 1024, NULL, SLGP_SHORTPATH);

    MessageBox(0, buffer, L"Link target path", MB_OK);

    pStorage->Release();
    pLink->Release();
}
```

Link to shell object

- To set the identifier list, call the IShellLink::SetIDList method, specifying the address of an identifier list, like printer

Shell extension

Shell actions

Handler	Description
Shortcut menu handler	Called before a file's shortcut menu is displayed. It enables you to add items to the shortcut menu on a file-by-file basis.
Data handler	Called when a drag-and-drop operation is performed on dragShell objects. It enables you to provide additional clipboard formats to the drop target.
Drop handler	Called when a data object is dragged over or dropped on a file. It enables you to make a file into a drop target.
Icon handler	Called before a file's icon is displayed. It enables you to replace the file's default icon with a custom icon on a file-by-file basis.
Property sheet handler	Called before an object's **Properties** property sheet is displayed. It enables you to add or replace pages.
Thumbnail Image handler	Provides an image to represent the item.
Infotip handler	Provides pop-up text when the user hovers the mouse pointer over the object.
Metadata handler	Provides read and write access to metadata (properties) stored in a file. This can be used to extend the Details view, infotips, the property page, and grouping features.

Shell operations

Handler	Description
Column handler	Called by Windows Explorer before it displays the Details view of a folder. It enables you to add custom columns to the Details view.
Copy hook handler	Called when a folder or printer object is about to be moved, copied, deleted, or renamed. It enables you to approve or veto the operation.
Drag-and-drop handler	Called when a file is dragged with the right mouse button. It enables you to modify the shortcut menu that is displayed.
Icon Overlay handler	Called before a file's icon is displayed. It enables you to specify an overlay for the file's icon.
Search handler	Called to launch a search engine. It enables you to implement a custom search engine accessible from the **Start** menu or Windows Explorer.