

```

/*****
/*
/*----- D D . C -----*/
/* Task      : Display Shell objects and their attributes starting
/*            from a point to be specified.
/*-----*/
/* Authors    : Michael Tischer and Bruno Jennrich
/* developed on : 09/25/1995
/* last update  : 09/25/1995
/*****
#include <windows.h>
#include <shlobj.h>
#include <stdio.h>
#include <string.h>

#include "enumfold.h"

//----- Constants -----
#define MINIMUM_ONE_ATTR 0          // At least 1 attribute must be set
#define ALL_ATTRIBUTES 1           // All attributes must be set
#define ALWAYS 2                   // Output all items

//----- Typedefs -----
typedef struct tagDISPLAYPARAMS
{
    DWORD dwDesiredMask;
    int    iType;
} DISPLAYPARAMS;
typedef DISPLAYPARAMS *PDISPLAYPARAMS;

typedef struct tagSTARTFROM      // Structure for receiving starting point
{
    LPSTR lpName;
    DWORD csidl;
} STARTFROM;

typedef struct tagATTRIBUTES     // Structure for receiving the attributes
{
    LPSTR lpName;
    DWORD dwAttrs;
} ATTRIBUTES;

//----- Global Variables -----
BOOL g_bRecurse;                // Browse all subfolders as well?

// All valid starting points -----
STARTFROM g_StartFrom[] =
{
    { "DESKTOP",          CSIDL_DESKTOP          },
    { "PROGRAMS",         CSIDL_PROGRAMS         },
    { "CONTROLS",         CSIDL_CONTROLS         },
    { "PRINTERS",         CSIDL_PRINTERS         },
    { "PERSONAL",         CSIDL_PERSONAL         },
    { "STARTUP",          CSIDL_STARTUP          },
    { "RECENT",           CSIDL_RECENT           },
    { "SENDTO",           CSIDL_SENDTO           },
    { "BITBUCKET",        CSIDL_BITBUCKET        },
    { "STARTMENU",        CSIDL_STARTMENU        },
    { "DESKTOPDIRECTORY", CSIDL_DESKTOPDIRECTORY },
    { "DRIVES",           CSIDL_DRIVES           },
    { "NETWORK",          CSIDL_NETWORK          },
    { "NETHOOD",          CSIDL_NETHOOD          },
    { "FONTS",            CSIDL_FONTS            },
    { "TEMPLATES",        CSIDL_TEMPLATES        },
    { NULL, 0 } };

// All valid attributes -----
ATTRIBUTES g_Attributes[] =
{
    { "CANCOPY",          SFGAO_CANCOPY          },
    { "CANMOVE",          SFGAO_CANMOVE          },
    { "CANLINK",          SFGAO_CANLINK          },
    { "CANRENAME",        SFGAO_CANRENAME        },
    { "CANDELETE",        SFGAO_CANDELETE        },
    { "HASPROPSHEET",     SFGAO_HASPROPSHEET },
    { "DROPTARGET",       SFGAO_DROPTARGET       },
    { "LINK",             SFGAO_LINK             },

```

```

{ "SHARE",          SFGAO_SHARE          },
{ "READONLY",       SFGAO_READONLY       },
{ "GHOSTED",        SFGAO_GHOSTED        },
{ "FILESYSANCESTOR", SFGAO_FILESYSANCESTOR },
{ "FOLDER",         SFGAO_FOLDER         },
{ "FILESYSTEM",     SFGAO_FILESYSTEM     },
{ "HASSUBFOLDER",   SFGAO_HASSUBFOLDER   },
{ "REMOVABLE",      SFGAO_REMOVABLE      },
{ NULL, 0 } };

/*****
/* EnumShellCallback - Callback function for enumerating Shell
/* objects
/*-----
/* Parameters:      lpFolder      - Address of IShellFolder
/*                  interface of current object
/*                  pszPath       - Pathname of object or NULL
/*                  pszDisplayName - DisplayName of object
/*                  ulAttrib      - Attributes of object
/*                  dwUser        - User information
/*                  pidlPath      - PIDL of object (relative to the
/*                  Desktop)
/*                  pidl         - PIDL of object (relative to
/*                  parent object)
/*                  iLevel       - Nesting level
/* Return value : TRUE - enumerate other objects of the current
/*                  nesting level.
/*                  FALSE - continue at parent level
/*-----
/* Info : This callback function is called by EnumFold
*****/
BOOL WINAPI EnumShellCallback( LPSHELLFOLDER lpFolder,
                              PSTR          pszPath,
                              PSTR          pszDisplayName,
                              DWORD         ulAttrib,
                              DWORD         dwUser,
                              LPITEMIDLIST pidlPath,
                              LPITEMIDLIST pidl,
                              int           iLevel )
{
    PDISPLAYPARAMS pDP;

    pDP = ( PDISPLAYPARAMS )dwUser;

    if( ( iLevel == 0 ) || g_bRecurse )
    {
        BOOL bPrint;

        // Does object have required attributes? -----
        switch( pDP->iType )
        {
            case MINIMUM_ONE_ATTR:
                bPrint = ( ulAttrib & pDP->dwDesiredMask ) ? TRUE : FALSE;
                break;
            case ALL_ATTRIBUTES:
                bPrint = ( ( ulAttrib & pDP->dwDesiredMask ) ==
                           pDP->dwDesiredMask ) ? TRUE : FALSE;
                break;
            case ALWAYS:
                bPrint = TRUE;
                break;
        }

        if( bPrint )
            // Output current object?
            { int i;

                for( i = 0; i < iLevel; i++ ) printf("  ");

                // Convert display names to OEM character set -----
                CharToOem( pszDisplayName, pszDisplayName );
                CharToOem( pszPath, pszPath );

                // Output object names -----
                printf( "%s [%s]\n", pszDisplayName, pszPath );

                // Display attributes? -----

```

```

if( pDP->dwDesiredMask )
{
    // Found at least one of the attributes to be displayed -----
    if( pDP->dwDesiredMask & ulAttrib )
    {
        for( i = 0; i < iLevel; i++ ) printf(" ");          // Indent
        printf("attr: ");
        i = 0;

        // Get name of found attribute -----
        while( g_Attributes[i].lpName )
        {
            // Output attribute name? -----
            if( ( g_Attributes[i].dwAttrs &
                pDP->dwDesiredMask & ulAttrib ) ==
                g_Attributes[i].dwAttrs )
                printf("%s ", g_Attributes[i].lpName ); //Output attributes
            else
            {
                if( pDP->dwDesiredMask & ( 1 << i ) )
                { int j;
                  for( j = strlen( g_Attributes[i].lpName ); j>= 0; j-- )
                      printf(" ");
                }
            }
            i++;
        }
        printf("\n");
    }
    else
    {
        for( i = 0; i < iLevel; i++ ) printf(" ");          // Indent
        printf("attr: -\n");
    }
}
}

// At level 0 all the objects are enumerated. The enumeration of
// higher levels is determined by g_bRecurse
return ( iLevel > 0 ) ? g_bRecurse : TRUE;
}

/*****
/* main - Start function
/*-----
/* Parameters:      argc - Number of passed parameters
/*                  argv - Pointer to parameters
/* Return value:    none
*****/
void main( int argc, char *argv[] )
{
    int i;
    DWORD dwStart,
           dwAttrs;
    DISPLAYPARAMS DP;

    CoInitialize( NULL );

    dwStart = (DWORD)-1;
    dwAttrs = 0;
    g_bRecurse = FALSE;

    printf("DesktopDir - (c) MITI & BHJ\n");

    if( argc == 1 )
    {
        printf("Call: dd START [-r] [-dTYPE] [-aATTRIBUT] [-aATTRIBUT]\n");
        printf("-r : Recursion through subfolders\n");
        printf("-d : Display-Type\n");
        printf("      TYPE : - : Display when at least one attribute set\n");
        printf("      TYPE : + : Display when all attributes set\n");
        printf("      TYPE : x : Always display\n");
        printf("START: DESKTOP      ATTRIBUT: CANCOPY\n");
        printf("      PROGRAMS      ATTRIBUT: CANMOVE\n");
        printf("      CONTROLS      ATTRIBUT: CANLINK\n");
    }
}

```

```

printf("        PRINTERS                CANRENAME\n");
printf("        STARTUP                  CANDELETE\n");
printf("        RECENT                     HASPROPSHEET\n");
printf("        SENDTO                     DROPTARGET\n");
printf("        BITBUCKET                  LINK\n");
printf("        STARTMENU                  SHARE\n");
printf("        DESKTOPDIRECTORY           READONLY\n");
printf("        DRIVES                     GHOSTED\n");
printf("        NETWORK                    FILESYSANCESTOR\n");
printf("        NETHOOD                    FOLDER\n");
printf("        FONTS                      FILESYSTEM\n");
printf("        TEMPLATES                  HASSUBFOLDER\n");
printf("                                REMOVABLE\n");
exit( 0 );
}

DP.iType = ALWAYS;

for( i = 1; i < argc; i++ )           // Browse all parameters
{
    // Determine whether display occurs with presence of at least one --
    // or all attributes or always. --
    if( !_strnicmp( argv[i], "-d", 2 ) )
    {
        switch( argv[i][2] )
        {
            case '-':
                DP.iType = MINIMUM_ONE_ATTR;
                break;
            case '+':
                DP.iType = ALL_ATTRIBUTES;
                break;
            case 'X':
            case 'x':
                DP.iType = ALWAYS;
                break;
            default:
            {
                LPSTR lpStr = "Invalid output parameter\n";           // ü to OEM
                CharToOem( lpStr, lpStr );
                printf( lpStr );
                exit(0);
            }
            break;
        }
    }
    else
    // Browse subfolders? -----
    if( !strcmp( argv[i], "-r" ) ) g_bRecurse = TRUE;
    else
    // Which attributes must the objects to be displayed have? -----
    if( !_strnicmp( argv[i], "-a", 2 ) )
    { int j = 0;
      // Compare command line parameters with attribute names -----
      while( ( g_Attributes[ j ].lpName ) && ( j >= 0 ) )
          if( !strcmp( g_Attributes[ j ].lpName, &argv[i][2] ) )
          {
              dwAttrs |= g_Attributes[ j ].dwAttrs;
              j = -1 ;
          }
          else j++;

      if( j > 0 )           // no matching attribute found
      {
          printf("Wrong attribute in parameter %d\n", i );
          exit( 0 );
      }
    }
    else
    // At which starting point should the display begin? -----
    {
        int j = 0;
        // Compare command line parameters with starting points -----
        while( ( g_StartFrom[ j ].lpName ) && ( j >= 0 ) )
            if( !strcmp( g_StartFrom[ j ].lpName, argv[i] ) )
            {

```

```

        dwStart = g_StartFrom[ j ].csidl;
        j = -1 ;
    }
    else j++;

    if( j > 0 ) // matching starting point not found -----
    {
        printf("Wrong starting point in parameter %d\n", i );
        exit( 0 );
    }
}

DP.dwDesiredMask = dwAttrs;

EnumFolders ( dwStart, EnumShellCallback, (DWORD)&DP ); // Begin display

CoUninitialize();
}

```