

```

/*****
/*                                P E R S I S T . C                                */
/*-----*/
/* Task      : Demonstrates persistent storage of program settings                */
/*            in the Registry using dialog box fields as an                      */
/*            example.                                                            */
/*-----*/
/* Authors    : Michael Tischer and Bruno Jennrich                             */
/* developed on : 08/30/1995                                                       */
/* last update  : 09/12/1995                                                       */
/*****
#include <windows.h>
#include <string.h>
#include "resource.h"

//----- Type Declarations -----

// Structure for saving dialog boxes in binary format -----
typedef struct tagBINARYDIALOG
{
    char Edit[ 256 ];
    int  CheckBox;
    int  Option;
    int  DropDownListSelection;
    int  AutoRun;
} BINARYDIALOG;

//----- Constants -----

// Key under which dialog box information is to be saved
#define MY_KEY      HKEY_CURRENT_USER
#define MY_SUBKEY    "Software\\PC_Intern\\Persist\\Version1.0"

/*****
/* SetAppPath : Sets the keys for starting programs quickly                    */
/*            by clicking the Start button and pointing at Run.                */
/*            Once the key has been entered, the user no longer                */
/*            has to specify the complete path to start the program,            */
/*            but only the name of the EXE file.                                */
/*-----*/
/* Parameters:  lpszProgName - Name of program                                */
/*            lpszCmdLine  - Command line for program call                    */
/*            lpszWorkDir  - Working directory                                */
/*            bSet        - TRUE : Set Registry entry                        */
/*            FALSE       - FALSE : Remove entry                            */
/* Return value : TRUE    : Operation executed                                */
/*            FALSE      : Error creating a Registry entry                    */
/*****
BOOL SetAppPath( LPSTR lpszProgName,
                 LPSTR lpszCmdLine,
                 LPSTR lpszWorkDir,
                 BOOL  bSet )
{
    #define APPPATH_KEY      HKEY_LOCAL_MACHINE
    #define APPPATH_SUBKEY    "Software\\Microsoft\\Windows\\CurrentVersion\\App Paths\\"

    char szRegistryKeyName[ MAX_PATH ];

    strcpy( szRegistryKeyName, APPPATH_SUBKEY );          // Create SubKey
    strcat( szRegistryKeyName, lpszProgName );

    // Delete path as a precaution -----
    RegDeleteKey( APPPATH_KEY, szRegistryKeyName );
    if( bSet )
    {
        HKEY hAppPath;
        DWORD dwDisposition;
        BOOL bRetVal = FALSE;

        // Create Quickstart key -----
        if( RegCreateKeyEx( APPPATH_KEY,
                           szRegistryKeyName,
                           0,
                           "",
                           REG_OPTION_NON_VOLATILE,
                           KEY_ALL_ACCESS,

```

```

        NULL,
        &hAppPath,
        &dwDisposition ) == ERROR_SUCCESS )
{
    // Delete default value (only necessary if RegDeleteKey() ---
    // failed)
    RegDeleteValue( hAppPath, NULL );
    RegDeleteValue( hAppPath, "Path" );

    // Default value of key = Path and name of the EXE file -----
    if( RegSetValueEx( hAppPath,
        NULL,
        0,
        REG_SZ,
        lpszCmdLine,
        lstrlen( lpszCmdLine ) ) == ERROR_SUCCESS )

        bRetVal = TRUE;
    // Path value of key = Working directory -----
    if( RegSetValueEx( hAppPath,
        "Path",
        0,
        REG_SZ,
        lpszWorkDir,
        lstrlen( lpszWorkDir ) ) == ERROR_SUCCESS )

        bRetVal = TRUE;
    RegCloseKey( hAppPath );
}
return bRetVal;
}
return TRUE;
}

/*****
/* SetAutoRun : Sets the keys to automatically start
/* an application after Windows 95 boots
/* up.
/* -----
/* Parameters: lpszProgName - Name of program
/* lpszCmdLine - Command line of program to
/* be executed.
/* bSet - TRUE : Set Registry entry
/* FALSE : Remove entry
/* Return value : TRUE : Operation completed
/* FALSE : Error creating a Registry entry
*****/
BOOL SetAutoRun( LPSTR lpszProgName,
    LPSTR lpszCmdLine,
    BOOL bSet )
{
#define AUTORUN_KEY HKEY_LOCAL_MACHINE
#define AUTORUN_SUBKEY "Software\\Microsoft\\Windows\\CurrentVersion\\Run"

HKEY hAutoRun;
BOOL bRetVal = FALSE;

// Get autorun key -----
if( RegOpenKeyEx( AUTORUN_KEY,
    AUTORUN_SUBKEY,
    0,
    KEY_ALL_ACCESS,
    &hAutoRun ) == ERROR_SUCCESS )
{
    // As a precaution, delete value for Persist if it already exists
    RegDeleteValue( hAutoRun, lpszProgName);

    if( bSet )
    {
        // Equip Persist dialog box entry with path of EXE file -----
        if( RegSetValueEx( hAutoRun,
            lpszProgName,
            0,
            REG_SZ,
            lpszCmdLine,
            lstrlen( lpszCmdLine ) ) == ERROR_SUCCESS )

            bRetVal = TRUE;
    }
}
}

```

```

    }
    else bRetVal = TRUE;
    RegCloseKey( hAutoRun );
}
return bRetVal;
}

/*****
/* SetUninstall : Sets the keys for calling the Uninstaller
/* of Control Panel.
/*
/*-----
/* Parameter : lpszProgName - Name of program
/* lpszDisplayName - Program name displayed in
/* the Control Panel
/* lpszUninstString - Program call for
/* uninstalling
/* bSet - TRUE : Set Registry entry
/* FALSE : Remove entry
/* Return value : TRUE : Operation completed
/* FALSE : Error creating a Registry entry
*****/
BOOL SetUninstall( LPSTR lpszProgName,
                  LPSTR lpszDisplayName,
                  LPSTR lpszUninstString,
                  BOOL bSet )
{
#define UNINSTALL_KEY HKEY_LOCAL_MACHINE
#define UNINSTALL_SUBKEY "Software\\Microsoft\\Windows\\CurrentVersion\\Uninstall\\"

char szRegistryKeyName[ MAX_PATH ];
// First, delete any existing Uninstall key -----
strcpy( szRegistryKeyName, UNINSTALL_SUBKEY ); // Create SubKey
strcat( szRegistryKeyName, lpszProgName );
RegDeleteKey( UNINSTALL_KEY, szRegistryKeyName );

if( bSet )
{
    HKEY hUninst;
    DWORD dwDisposition;
    BOOL bRetVal = FALSE;

    // Create new Uninstall key -----
    if( RegCreateKeyEx( UNINSTALL_KEY,
                      szRegistryKeyName,
                      0,
                      "",
                      REG_OPTION_NON_VOLATILE,
                      KEY_ALL_ACCESS,
                      NULL,
                      &hUninst,
                      &dwDisposition ) == ERROR_SUCCESS )
    {
        // "DisplayName" contains the name of the program as it appears
        // in the Control Panel.
        if( RegSetValueEx( hUninst,
                          "DisplayName",
                          0,
                          REG_SZ,
                          lpszDisplayName,
                          strlen( lpszDisplayName ) ) == ERROR_SUCCESS )
        {
            bRetVal = TRUE;

            // The command line is entered in "UninstallString" -----
            if( RegSetValueEx( hUninst,
                              "UninstallString",
                              0,
                              REG_SZ,
                              lpszUninstString,
                              strlen( lpszUninstString ) ) == ERROR_SUCCESS )
            {
                bRetVal = TRUE;

                RegCloseKey( hUninst );
            }
        }
        return bRetVal;
    }
    return TRUE;
}

```

[illegible]

```

        sizeof( dwValue ) ) != ERROR_SUCCESS );

if( fSaveBinary ) // Save dialog boxe fields in structure?
{
    BINARYDIALOG bd;

    // Create structure -----
    GetWindowText( GetDlgItem( hDlg, IDC_EDIT ),
        bd.Edit, sizeof( bd.Edit ) );

    bd.CheckBox = IsDlgButtonChecked( hDlg, IDC_CHECK );

    if( IsDlgButtonChecked( hDlg, IDC_OPTION1 ) )
        bd.Option = 0;
    else
        if( IsDlgButtonChecked( hDlg, IDC_OPTION2 ) )
            bd.Option = 1;
        else
            if( IsDlgButtonChecked( hDlg, IDC_OPTION3 ) )
                bd.Option = 2;

    bd.DropListSelection = SendMessage(
        GetDlgItem( hDlg, IDC_DROPLIST ),
        CB_GETCURSEL, 0, 0 );

    bd.AutoRun = IsDlgButtonChecked( hDlg, IDC_AUTORUN );

    // Encode structure -----
    XORData( ( PBYTE )&bd, sizeof( bd ) );

    // Save structure under the value of "BinaryRepresentation" -----
    fSaveError |=
        ( RegSetValueEx( hMyPrivateKey,
            "BinaryRepresentation",
            0,
            REG_BINARY,
            ( PBYTE )&bd,
            ( DWORD )sizeof(bd)) != ERROR_SUCCESS);
}
else
{
    // Save dialog box fields in separate key values -----

    // Save text box under "Text" -----
    cchWindowText = GetWindowText( GetDlgItem( hDlg, IDC_EDIT ),
        chBuffer, sizeof( chBuffer ));
    fSaveError |= ( RegSetValueEx( hMyPrivateKey,
        "Text",
        0,
        REG_SZ,
        chBuffer,
        ( DWORD )cchWindowText )
        != ERROR_SUCCESS );

    // Save CheckBox under "CheckBox" -----
    dwValue = IsDlgButtonChecked( hDlg, IDC_CHECK );
    fSaveError |= ( RegSetValueEx( hMyPrivateKey,
        "CheckBox",
        0,
        REG_DWORD,
        ( CONST BYTE *)&dwValue,
        sizeof( dwValue ) )
        != ERROR_SUCCESS );

    // Save RadioButton under "Option" -----
    dwValue = 0xFFFFFFFF;
    if( IsDlgButtonChecked( hDlg, IDC_OPTION1 ) ) dwValue = 0;
    else
        if( IsDlgButtonChecked( hDlg, IDC_OPTION2 ) ) dwValue = 1;
        else
            if( IsDlgButtonChecked( hDlg, IDC_OPTION3 ) ) dwValue = 2;

    fSaveError |= ( RegSetValueEx( hMyPrivateKey,
        "Option",
        0,
        REG_DWORD,

```

```

        ( CONST BYTE *)&dwValue,
        sizeof( dwValue ) )
        != ERROR_SUCCESS );

// Save current list item under "DropList" -----
dwValue = SendMessage( GetDlgItem( hDlg, IDC_DROPLIST ),
        CB_GETCURSEL,
        0, 0 );
fSaveError |= ( RegSetValueEx( hMyPrivateKey,
        "DropList",
        0,
        REG_DWORD,
        ( CONST BYTE *)&dwValue,
        sizeof( dwValue ) )
        != ERROR_SUCCESS );

// Save AutoRun value
dwValue = IsDlgButtonChecked( hDlg, IDC_AUTORUN );
fSaveError |= ( RegSetValueEx( hMyPrivateKey,
        "AutoRun",
        0,
        REG_DWORD,
        ( CONST BYTE *)&dwValue,
        sizeof( dwValue ) )
        != ERROR_SUCCESS );
}

RegCloseKey( hMyPrivateKey );           // Release key
return !fSaveError;
}
return FALSE;
}

/*****
/* LoadControlValues : Load contents of dialog box controls from
/* Registry
/*-----*/
/* Parameter : hDlg - Handle of dialog box
/* Return value : TRUE - Save was successful
/* FALSE - Save error
*****/
BOOL LoadControlValues( HWND hDlg )
{
    char chBuffer[ 256 ];                // Text buffer for Edit box
    DWORD cchWindowText;                // Size of loaded text
    DWORD dwValue;                      // Loaded DWORD value
    DWORD dwSize;                      // Size of loaded data
    DWORD dwType;                      // Type of loaded value
    BOOL fLoadError;                   // Load error occurred?
    HKEY hMyPrivateKey;                // Handle of key
    UINT fBinaryLoad;                  // Load data from binary structure?

    // Open key to be loaded (don't create) -----
    fLoadError = FALSE;                // Still no error!

    if( RegOpenKeyEx( MY_KEY, MY_SUBKEY, 0, KEY_ALL_ACCESS, &hMyPrivateKey ) == ERROR_SUCCESS )
    {
        // Load "BinarySave" DWORD -----
        dwSize = sizeof( fBinaryLoad );
        fLoadError |= ( RegQueryValueEx( hMyPrivateKey,
            "BinarySave",
            NULL,
            &dwType,
            ( PBYTE )&fBinaryLoad,
            &dwSize ) != ERROR_SUCCESS );

        // Load data from binary structure ? -----
        if( (!fLoadError) && fBinaryLoad )
        {
            BINARYDIALOG bd;

            // Enable "BinarySave" CheckBox -----
            CheckDlgButton( hDlg, IDC_BINARYSAVE, TRUE );

            // Load BINARYDIALOG structure -----
            dwSize = sizeof( bd );

```

```

if( RegQueryValueEx( hMyPrivateKey,
                    "BinaryRepresentation",
                    NULL,
                    &dwType,
                    ( PBYTE )&bd,
                    &dwSize ) == ERROR_SUCCESS )
{ // Able to load structure -----

    // Cancel encoding -----
    XORData( ( PBYTE )&bd, sizeof( bd ) );

    // Initialize dialog box fields with loaded values -----
    SetWindowText( GetDlgItem( hDlg, IDC_EDIT ), bd.Edit );
    CheckDlgButton( hDlg, IDC_CHECK, bd.CheckBox );
    switch( bd.Option )
    {
        case 0: CheckDlgButton( hDlg, IDC_OPTION1, TRUE ); break;
        case 1: CheckDlgButton( hDlg, IDC_OPTION2, TRUE ); break;
        case 2: CheckDlgButton( hDlg, IDC_OPTION3, TRUE ); break;
    }
    SendMessage( GetDlgItem( hDlg, IDC_DROPLIST ),
                  CB_SETCURSEL,
                  bd.DropListSelection, 0 );
    CheckDlgButton( hDlg, IDC_AUTORUN, bd.AutoRun );
}
else fLoadError = TRUE;
}
else // Load dialog box fields separately -----
{
    // Read Edit box -----
    cchWindowText = sizeof( chBuffer );
    if( RegQueryValueEx( hMyPrivateKey,
                        "Text",
                        NULL,
                        &dwType,
                        chBuffer,
                        &cchWindowText ) == ERROR_SUCCESS )
        SetWindowText( GetDlgItem( hDlg, IDC_EDIT ), chBuffer );
    else fLoadError = TRUE;

    // Read CheckBox -----
    dwSize = sizeof( dwValue );
    if( RegQueryValueEx( hMyPrivateKey,
                        "CheckBox",
                        NULL,
                        &dwType,
                        ( PBYTE )&dwValue,
                        &dwSize ) == ERROR_SUCCESS )
        CheckDlgButton( hDlg, IDC_CHECK, (int)dwValue );
    else fLoadError = TRUE;

    // Read RadioButtons -----
    dwSize = sizeof( dwValue );
    if( RegQueryValueEx( hMyPrivateKey,
                        "Option",
                        NULL,
                        &dwType,
                        ( PBYTE )&dwValue,
                        &dwSize ) == ERROR_SUCCESS )
    {
        switch( dwValue )
        {
            case 0: CheckDlgButton( hDlg, IDC_OPTION1, TRUE ); break;
            case 1: CheckDlgButton( hDlg, IDC_OPTION2, TRUE ); break;
            case 2: CheckDlgButton( hDlg, IDC_OPTION3, TRUE ); break;
        }
    }
    else fLoadError = TRUE;

    // Load list index for DropList -----
    dwSize = sizeof( dwValue );
    if( RegQueryValueEx( hMyPrivateKey,
                        "DropList",
                        NULL,
                        &dwType,
                        ( PBYTE )&dwValue,
                        &dwSize ) == ERROR_SUCCESS )
        SendMessage( GetDlgItem( hDlg, IDC_DROPLIST ),

```

```

                CB_SETCURSEL,
                ( WPARAM )dwValue, 0 );
else fLoadError = TRUE;

// Read AutoRun -----
dwSize = sizeof( dwValue );
if( RegQueryValueEx( hMyPrivateKey,
                    "AutoRun",
                    NULL,
                    &dwType,
                    ( PBYTE )&dwValue,
                    &dwSize ) == ERROR_SUCCESS )
    CheckDlgButton( hDlg, IDC_AUTORUN, (int)dwValue );
else fLoadError = TRUE;
}

RegCloseKey( hMyPrivateKey ); // Release key
return !fLoadError;
}
return FALSE;
}

/*****
/* DlgProc : Simple dialog box procedure */
/*-----*/
/* Parameters: Default dialog box parameters */
/* Return value : Default return values */
*****/
BOOL WINAPI DlgProc( HWND hDlg, UINT wMsg, WPARAM wp, LPARAM lp )
{
    switch( wMsg )
    {
        case WM_INITDIALOG: // Fill DropList with dummy items
            SendMessage( GetDlgItem( hDlg, IDC_DROPLIST ),
                        CB_ADDSTRING, 0, ( LONG )( LPVOID )"Item 1");
            SendMessage( GetDlgItem( hDlg, IDC_DROPLIST ),
                        CB_ADDSTRING, 0, ( LONG )( LPVOID )"Item 2");
            SendMessage( GetDlgItem( hDlg, IDC_DROPLIST ),
                        CB_ADDSTRING, 0, ( LONG )( LPVOID )"Item 3");
            SendMessage( GetDlgItem( hDlg, IDC_DROPLIST ),
                        CB_ADDSTRING, 0, ( LONG )( LPVOID )"Item 4");
            SendMessage( GetDlgItem( hDlg, IDC_DROPLIST ),
                        CB_ADDSTRING, 0, ( LONG )( LPVOID )"Item 5");

            if( !LoadControlValues( hDlg ) )
                MessageBox( hDlg,
                            "Cannot find one or more items!",
                            "Warning!",
                            MB_OK );

            break;
        case WM_COMMAND:
            switch( wp )
            {
                case IDCANCEL:
                    // Exit dialog box without saving -----
                    EndDialog( hDlg, FALSE );
                    break;
                case IDOK:
                    {
                        char szPath[ MAX_PATH ];

                        // Get path of program -----
                        GetModuleFileName( (HINSTANCE)GetWindowLong( hDlg, GWL_HINSTANCE ),
                                           szPath, sizeof( szPath ) );

                        // Exit dialog box under previous save -----
                        if( !SaveControlValues( hDlg ) )
                            MessageBox( hDlg,
                                        "Cannot save one or more items!",
                                        "Warning!",
                                        MB_OK );

                        SetAutoRun( "Persist", szPath,
                                   IsDlgButtonChecked( hDlg, IDC_AUTORUN ) );
                        EndDialog( hDlg, TRUE );
                    }
            }
    }
}

```



```

        break;
    }
}
return FALSE;
}

/*****
/* WinMain : Main function */
/*-----*/
/* Parameters:    Default parameters */
/* Return value : Default return value */
/*****
int WINAPI WinMain( HINSTANCE hInst,
                   HINSTANCE hPrev,
                   LPSTR      lpCmdLine,
                   int         nCmdShow )
{
    // Test if Uninstall was called -----
    if( strcmp( lpCmdLine, "/u" ) == 0 )
    {
        MessageBox( NULL, "Uninstall called!", "", 0 );

        // Delete Registry entries for AutoStart, FastCall and Uninstall ---
        SetAppPath( "PERSIST.EXE", NULL, NULL, FALSE );
        SetAutoRun( "Persist", NULL, FALSE );
        SetUninstall( "Persist", NULL, NULL, FALSE );

        // Delete Registry entries for persistent dialog box -----
        RegDeleteKey( MY_KEY, MY_SUBKEY );
    }
    else // Program started normally
    {
        char szPath[ MAX_PATH + 5 ]; // MAX_PATH plus Uninstall parameters
        char szCurDir[ MAX_PATH ]; // Current directory

        // Get path of program -----
        GetModuleFileName( hInst, szPath, sizeof( szPath ) );

        // With each restart, store App Paths and Uninstall information ----
        // in Registry. (Needed in case application is moved to a new
        // directory, for example)

        GetCurrentDirectory( sizeof( szCurDir ), szCurDir );
        SetAppPath( "PERSIST.EXE", szPath, szCurDir, TRUE );

        strcat( szPath, " /u"); // Add switch for uninstallation
        SetUninstall( "Persist", "Persist", szPath, TRUE );

        // Open modal dialog box -----
        DialogBox( hInst, MAKEINTRESOURCE( IDD_DIALOG ), NULL, DlgProc );
    }
    return 0;
}

```