

Our good old first Win32 programme of lecture 8

```
#include <windows.h>
```

```
int WINAPI WinMain(HINSTANCE hInstance,  
                  HINSTANCE hPrevInstance,  
                  LPSTR      lpCmdLine,  
                  int         nCmdShow)  
{  
    MessageBox(NULL, "This is our first Windows  
Programming Application.", "Virtual  
University", MB_OK);  
  
    return 0;  
}
```

Modal Dialog Box contains a ***Modal Loop***

DialogBox() API function creates a Modal Dialog.

DialogBox() does not return until the dialog is dismissed.

Dialog Resource Definition Statement

Dialog Resource Template

```
IDD_ABOUT_DIALOG DIALOG DISCARDABLE  0, 0, 265, 124
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON    "OK",IDOK,208,7,50,14
    PUSHBUTTON       "Cancel",IDCANCEL,208,24,50,14
    LTEXT             "Some copyright text", IDC_STATIC,
                      67, 27,107,47
    ICON              IDI_ICON_VU,IDC_STATIC,17,14,20,20
END
```

Controls in a Dialog Resource Definition statement

LTEXT	Left-aligned static control
RTEXT	Right-aligned static control
CTEXT	Centre-aligned static control

DialogBox() API

```
INT_PTR DialogBox(  
    HINSTANCE hInstance, // handle to module  
    LPCTSTR lpTemplate,  // dialog box template  
    HWND hWndParent,     // handle to owner window  
    DLGPROC lpDialogFunc // dialog box procedure  
);
```

Dialog Box Procedure

```
BOOL CALLBACK DialogProc(  
    HWND hwndDlg,    // handle to dialog box  
    UINT uMsg,       // message  
    WPARAM wParam,   // first message parameter  
    LPARAM lParam    // second message parameter  
);
```

Dialog Box Procedure

```
BOOL CALLBACK AboutAuthorDialog(HWND hDlg, UINT message,
                                WPARAM wParam, LPARAM lParam)
{
    switch(message)
    {
        case WM_INITDIALOG:
            return TRUE;

        case WM_COMMAND:
            switch(LOWORD(wParam))
            {
                case IDOK:
                case IDCANCEL:
                    EndDialog(hDlg, 0);
                    return TRUE;
            }
            break;
    }
    return FALSE;
}
```

WM_INITDIALOG message

Sent by system just after creating the dialog and just before making it visible

If FALSE is returned, it prevents the system from setting the default keyboard focus.

If TRUE is returned, keyboard focus is set to the control specified by *wParam*.

Displaying “About...” box from a menu

```
case ID_HELP_ABOUT: // menu identifier

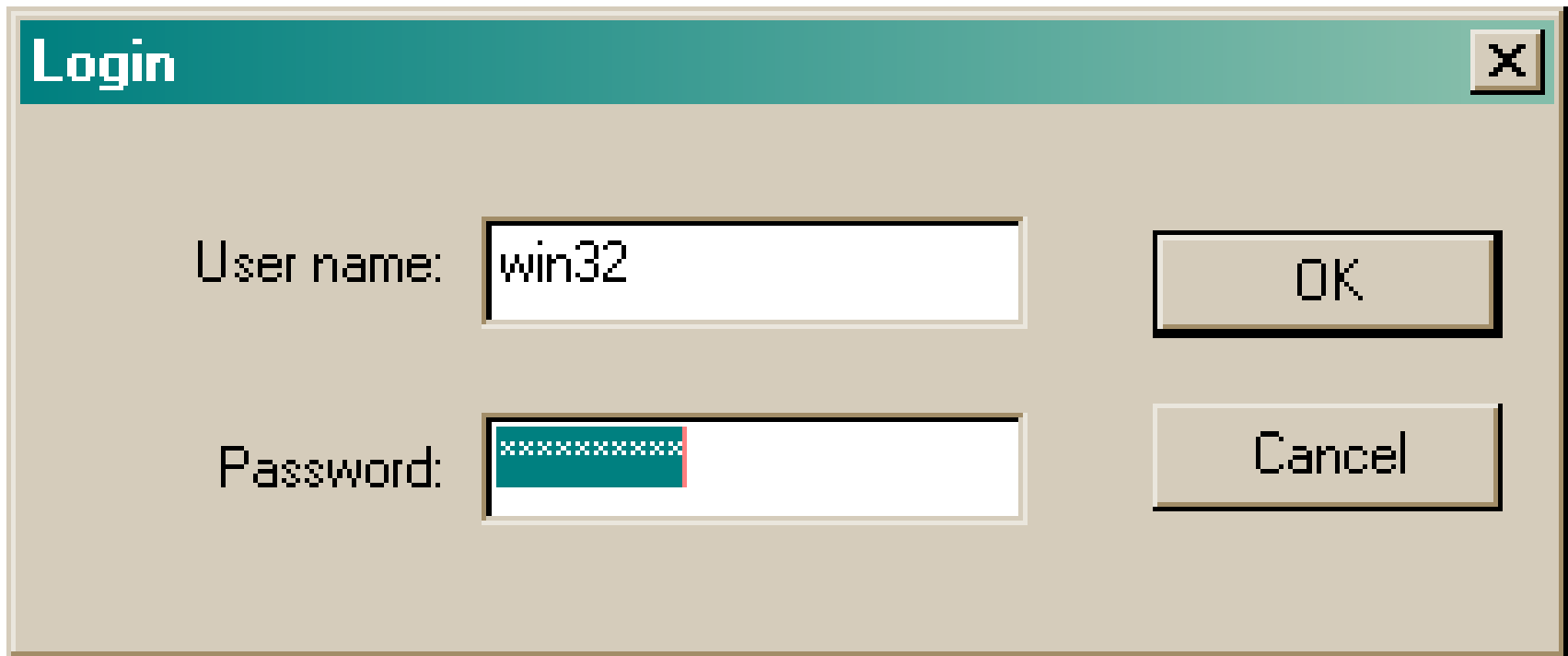
    DialogBox(hInstance,
              MAKEINTRESOURCE(IDD_ABOUT_DIALOG),
              hwnd,
              AboutAuthorDialog);

    break;
```

About dialog



Modal Dialogs



A modal dialog box titled "Login" with a close button (X) in the top right corner. The dialog contains two input fields: "User name:" with the text "win32" and "Password:" with masked characters "xxxxxxxxxx". To the right of the input fields are two buttons: "OK" and "Cancel".

Login

User name: win32

Password: xxxxxxxxxxxx

OK

Cancel

GetDlgItem

Retrieve a handle to a control in the specified dialog box.

```
HWND GetDlgItem(  
    HWND hDlg,      // handle to dialog box  
    int nIDDlgItem  // control identifier  
);
```

SetWindowText

Set the text of a control or the title bar of a window

```
BOOL SetWindowText(  
    HWND hWnd,      // handle to window or control  
    LPCTSTR lpString // title or text  
);
```

GetDlgCtrlID

Retrieve the identifier of the specified control.

```
int GetDlgCtrlID(  
    HWND hwndCtl  // handle to control  
);
```

GetDlgItemText

Retrieve the text associated with a control in a dialog box

```
UINT GetDlgItemText(  
    HWND hDlg,      // handle to dialog box  
    int nIDDlgItem, // control identifier  
    LPTSTR lpString, // pointer to buffer for text  
    int nMaxCount   // maximum size of string  
);
```

SendDlgItemMessage

sends a message to the specified control in a dialog box

```
LRESULT SendDlgItemMessage(  
    HWND hDlg,    // handle to dialog box  
    int nIDDlgItem, // control identifier  
    UINT Msg,     // message to send  
    WPARAM wParam, // first message parameter  
    LPARAM lParam  // second message parameter  
);
```


Edit control messages

EM_LIMITTEXT,

wParam, // text length

lParam // not used; must be zero

Sets the text limit of an edit control

Window/control messages

Setting or getting text associated with a window or control

WM_GETTEXT

wParam, // number of characters to copy

lParam // text buffer

WM_SETTEXT

wParam, // not used; must be zero

lParam // window-text string (LPCTSTR)

GetWindowText() function internally sends a
WM_GETTEXT message to get the text

Window/control messages

Set or retrieve current selection in an edit control

EM_SETSEL or EM_GETSEL

wParam, // starting position

lParam // ending position

Parameters to a Dialog

Passing information to the dialog

```
INT_PTR DialogBoxParam(  
    HINSTANCE hInstance,    // handle to module  
    LPCTSTR lpTemplateName, // dialog box template  
    HWND hWndParent,        // handle to owner window  
    DLGPROC lpDialogFunc,   // dialog box procedure  
    LPARAM dwInitParam      // initialization value  
);
```

lParam parameter of WM_INITDIALOG contains dwInitParam

CreateDialog

```
HWND CreateDialog(  
    HINSTANCE hInstance, // handle to module  
    LPCTSTR lpTemplate,  // dialog box template name  
    HWND hWndParent,     // handle to owner window  
    DLGPROC lpDialogFunc // dialog box procedure  
);
```

Creates a modeless dialog and returns a handle to the new dialog

Window show state

```
BOOL ShowWindow(  
    HWND hWnd,    // handle to window  
    int nCmdShow  // show state  
);
```

Sets the show-state of a specified window

Dispatching modeless dialog messages

Processes a message if it is intended for the specified dialog box.

```
BOOL IsDialogMessage(  
    HWND hDlg, // handle to dialog box  
    LPMSG lpMsg // message to be checked  
);
```

Message Loop

Message loop to dispatch messages to a modeless dialog

```
While (GetMessage (&msg, NULL, 0, 0) > 1)
{
    if (!IsDialogMessage (hDlg, &msg))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
}
```


Windows Common Dialogs

- File Open
- Choose font
- Choose colour
- Print

Windows Common Controls

- Date time picker
- List View
- Property sheets
- Status bar
- Toolbar
- Tree View