MFC Topics: Message Boxes

Introduction to Message Boxes

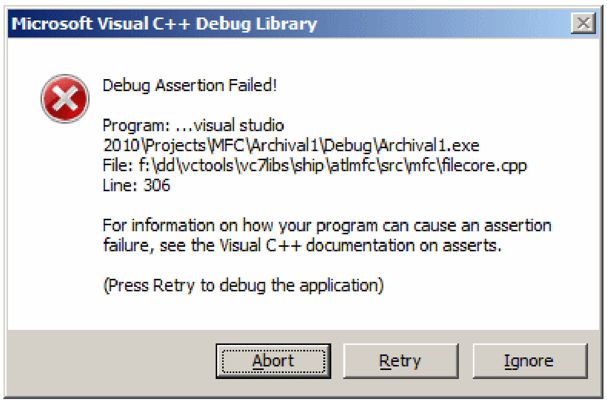
Definition

A message box is a bordered rectangular window that displays a short message to the user. The message can be made of one sentence, various lines, or a few paragraphs. The user cannot change the text but can only read. A message box is created either to inform the user about something or to request his or her decision about an issue. When a dialog box displays from an application, the user must close it before continuing using the application. This means that, as long as the message box is displaying, the user cannot access any other part of the application until the message box is first dismissed.

The simplest message box is used to display a message to the user. This type of message box is equipped with only one button, labeled OK. Here is an example:

Figure 34: A Simple Message Box

A more elaborate or detailed message box can display an icon and/or can have more than one button:


Microsoft Visual C++ Debug Library

Debug Assertion Failed!

Program: ...visual studio 2010\Projects\MFC\Archival1\Debug\Archival1.exe
File: f:\dd\vctools\vc7libs\ship\atlmfc\src\mfc\filecore.cpp
Line: 306

For information on how your program can cause an assertion failure, see the Visual C++ documentation on asserts.

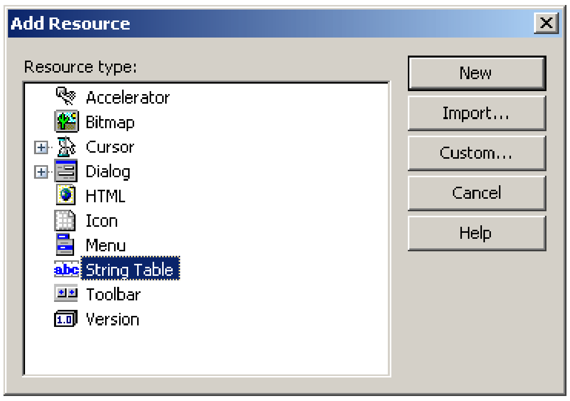
(Press Retry to debug the application)

Practical Learning: Starting an MFC Application

1. Start Microsoft Visual Studio
2. On the main menu, click File -> New Project...
3. In the Project Types, click Visual C++
4. In the Templates list, click Win32 Project
5. Set the Name to **Exercise2**
6. Click OK
7. On the first page of the wizard, click Next
8. On the second page, accept Windows Application and click Empty Project
9. Click Finish
10. To make sure that this application uses MFC, on the main menu, click Project -> Properties...
11. In the left list, click Configuration Properties
12. In the right list, click User of MFC, click the arrow of its combo box and select Use MFC In A Shared DLL
13. Click OK
14. In the Solution Explorer, right-click Source Files, position the mouse on Add, and click New Item...
15. In the Templates lists click C++ File (.cpp)
16. Set the Name to **Exercise**
17. Click Add
18. In the empty document, type the following:

```
#include <afxwin.h>

class CExerciseApp : public CWinApp
{
    BOOL InitInstance()
    {
 return TRUE;
    }
};

CExerciseApp theApp;
```

19. Save the file
20. If the Resource View is not visible, on the main menu, click View -> Other Windows -> Resource View.
    In the Reource View, right-click Exercise1 -> Add -> Resource...
21. In the Add Resource dialog box, click String Table



22. Click New
23. Under the Type column header, click IDS_STRING101 and edit it to display **IDS_ANNOUNCE** and press Enter
24. To save, on the Standard toolbar, click the Save button
25. Close that window by clicking the Close button ☒
26. In the Resource View, expand the Exercise1 node et notice the Exercise1.rc node

Creating a Message Box

To create a message box, the Win32 library provides a global function called **MessageBox**. Its syntax is:

```
int MessageBox(HWND hWnd, LPCTSTR lpText, LPCTSTR lpCaption, UINT uType);
```

If you want to use this version from Win32, because it is defined outside of MFC, you should start it with the scope access operator â€œ::â€ as follows:

```
::MessageBox(. . .);
```

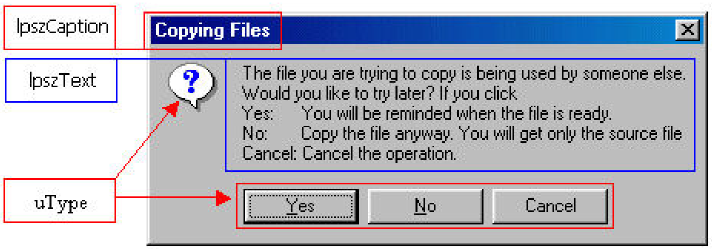
As we will see shortly, the Win32's **MessageBox()** function requires a handle (not necessarily difficult to provide but still). To make the creation of a message a little easier, the MFC library provides its own version of the **MessageBox()** function. Its syntax is:

```
int MessageBox(LPCTSTR lpszText,
        LPCTSTR lpszCaption = NULL,
        UINT nType = MB_OK);
```

To still make it easier, the MFC framework provides the **AfxMessageBox** function used to create a message box. Although this function is global, its scope is limited to MFC applications. This means that you can use the Win32's global **MessageBox()** function with any compiler used on the Microsoft Windows operating systems but you cannot use the MFC's global **AfxMessageBox()** function with any compiler other Visual C++. The **AfxMessageBox()** function is overloaded with two versions whose syntaxes are:

```
int AfxMessageBox(LPCTSTR lpszText,
    UINT nType = MB_OK,
    UINT nIDHelp = 0);
int AFXAPI AfxMessageBox(UINT nIDPrompt,
    UINT nType = MB_OK,
    UINT nIDHelp = (UINT) -1);
```

Based on the above functions, a message box can be illustrated as follows:



The file you are trying to copy is being used by someone else.
Would you like to try later? If you click
Yes: You will be reminded when the file is ready.
No: Copy the file anyway. You will get only the source file
Cancel: Cancel the operation.

The Parameters of a Message Box

The Owner of a Message Box

As seen above, you have the choice among three functions to create a message. There is no valid reason that makes one of them better than the other. They do exactly the same thing. The choice will be based on your experience and other factors.

If you decide to use the Win32's **MessageBox()** function, you must specify the handle to the application that created the message box. As we will learn eventually when we study controls, you can get a handle to a (**CWnd**-derived) control with a call to **m_hWnd**. For example, if a button on a dialog box initiates the message box, you can start this function as follows:

```
::MessageBox(m_hWnd, â€¦);
```

We also saw that you can get a pointer to the main window by calling the MFC's global **AfxGetMainWnd()** function. This function only points you to the application. To get a handle to the application, you can call the same **m_hWnd** member variable. In this case, the message box can be started with:

```
::MessageBox(AfxGetMainWnd()->m_hWnd, â€¦);
```

If you are creating a message but do not want a particular window to own it, pass this **hWnd** argument as **NULL**.

Using a String Resource

If you decide to call the **AfxMessageBox()** function, as seen previously, you have two options. You can pass a string as argument. Here is an example:

```
AfxMessageBox(L"Welcome to Microsoft Foundation Class Library.");
```

An alternative is to pass a string identifier from a resource file.

Practical Learning: Creating a Message Box

1. In the Resource View, expand the String Table node and double-click the String Table file
2. Click under Caption (on the right side of 101) et type **Make sure you fill out your time sheet before leaving**
3. Click the Exercise.cpp tab and change the file as follows:

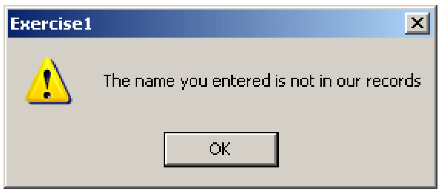
```
#include <afxwin.h>
#include "resource.h"

class CExerciseApp : public CWinApp
{
    BOOL InitInstance()
    {
 AfxMessageBox(IDS_ANNOUNCE);
 return TRUE;
    }
};

CExerciseApp theApp;
```

4. To execute the application, press Ctrl + F5
5. When asked to build, click Yes

The Message of the Box

For all the message box functions, the *Text* argument a null-terminated string. It specifies the text that would be displayed to the user. This argument is required. Here is an example that uses the **MessageBox()** function:

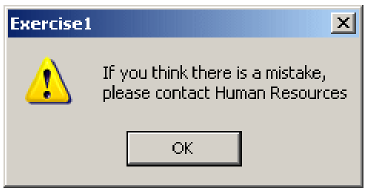
```
AfxMessageBox(L"The name you entered is not in our records");
```


Exercise1

The name you entered is not in our records

If you want to display the message on various lines, you can separate sections with the new line character '\n'. Here is an example:

```
AfxMessageBox(L"If you think there is a mistake,\nplease contact Human Resources");
```


Exercise1

If you think there is a mistake,
please contact Human Resources

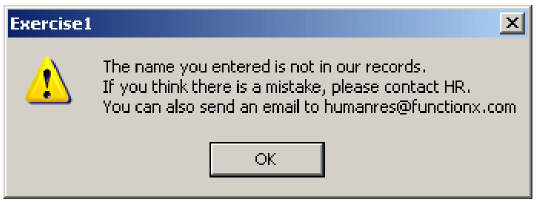
If the message you are creating is too long to fit on one line, you can separate lines by ending each with a double-quote and starting the next line with a new double-quote. As long as you have not closed the function, the string would be considered as one entity. You can also use string editing and formatting techniques to create a more elaborate message. This means that you can use functions of the C string library to create your message.

Practical Learning: Displaying a Message

1. To create a message box, call the **AfxMessageBox()** function as follows:

```
#include <afxwin.h>
#include "resource.h"

class CExerciseApp : public CWinApp
{
    BOOL InitInstance()
    {
 AfxMessageBox(L"The name you entered is not in our records.\n"
        L"If you think there is a mistake, please contact HR.\n"
        L"You can also send an email to humanres@functionx.com");

 return TRUE;
    }
};

CExerciseApp theApp;
```

2. Press Ctrl + F5 to test the application and click Yes


Exercise1

The name you entered is not in our records.
If you think there is a mistake, please contact HR.
You can also send an email to humanres@functionx.com

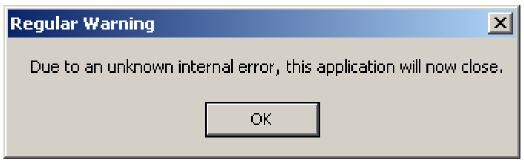
3. Click OK to close the message box and return to your programming environment

The Message's Title

The caption of the message box is the text that displays on its title bar. If you create a message box using the **AfxMessageBox()** function, it allows you to specify only one argument, namely the text to display to the user, which is the value of the *Text* argument. In this case, the title bar of the message box would display the name of the application that "owns" the message box.

If you want to display your own caption on the title bar, call the **MessageBox()** function. Specify the argument for the caption. The *Caption* argument is a null-terminated string that would display on the title bar of the message box. Here is an example:

```
MessageBox(NULL,
      L"Due to an unknown internal error, this application will now close.",
      L"Regular Warning", 0);
```

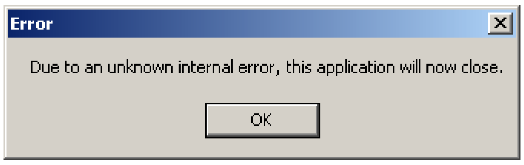

Regular Warning

Due to an unknown internal error, this application will now close.

Although the *Caption* argument is optional in the MFC,s **MessageBox()** and the **AfxMessageBox()** functions, it is required for the Win32's **MessageBox()** function. Because it is in fact a pointer, you can pass it as NULL. Here is an example:

```
MessageBox(NULL,
      L"Due to an unknown internal error, this application will now close.",
      NULL, 0);
```

In this case, the title bar of the message box would display Error:


Error

Due to an unknown internal error, this application will now close.

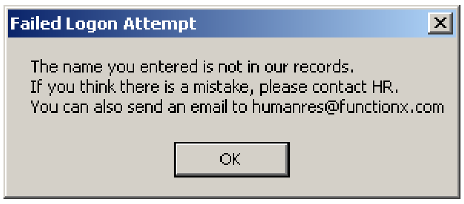
This caption may not be friendly on most applications and could appear frightening to the user. Therefore, unless you are in a hurry, you should strive to provide a friendly and more appropriate title.

Practical Learning: Displaying a Message's Caption

1. To display a caption for the message box, change the **MessageBox()** call as follows:

```
#include <afxwin.h>
#include "resource.h"

class CExerciseApp : public CWinApp
{
    BOOL InitInstance()
    {
 MessageBox(NULL,
  L"The name you entered is not in our records.\n"
  L"If you think there is a mistake, please contact HR.\n"
  L"You can also send an email to humanres@functionx.com",
  L"Failed Logon Attempt", 0);

 return TRUE;
    }
};

CExerciseApp theApp;
```

2. Test the application


Failed Logon Attempt

The name you entered is not in our records.
If you think there is a mistake, please contact HR.
You can also send an email to humanres@functionx.com

3. Return to your programming environment

Message Box Buttons

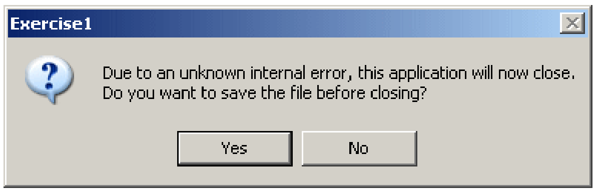
The *uType* argument is used to provide some additional options to the message box. First, it is used to display one or a few buttons. The buttons depend on the value specified for the argument. If this argument is not specified, the message box displays (only) OK. Otherwise, you can display a message box with a combination of selected buttons.

To display one or more buttons, the *uType* argument uses a constant value that controls what button(s) to display. The values and their buttons can be specified as follows:

Constant	Buttons
MB_OK	OK
MB_OKCANCEL	OK Cancel
MB_ABORTRETRYIGNORE	Abort Retry Ignore
MB_YESNOCANCEL	Yes No Cancel
MB_YESNO	Yes No
MB_RETRYCANCEL	Retry Cancel
MB_HELP	

Here is an example:

```
AfxMessageBox(L"Due to an unknown internal error, this application will now close.\n"
           L"Do you want to save the file before closing?",
                   MB_YESNO);
```


Exercise1

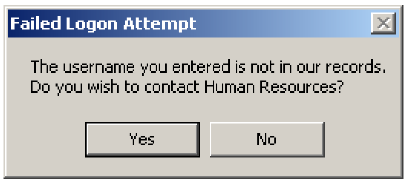
Due to an unknown internal error, this application will now close.
Do you want to save the file before closing?

Practical Learning: Showing Message Box Buttons

1. To display an combination of buttons on the message box, change the **InitInstance()** member function as follows:

```
#include <afxwin.h>
#include "resource.h"

class CExerciseApp : public CWinApp
{
    BOOL InitInstance()
    {
 ::MessageBox(NULL,
  L"The username you entered is not in our records.\n"
  L"Do you wish to contact Human Resources?",
  L"Failed Logon Attempt",
  MB_YESNO);

 return TRUE;
    }
};

CExerciseApp theApp;
```

2. Test the application


Failed Logon Attempt

The username you entered is not in our records.
Do you wish to contact Human Resources?

3. Close it and return to your programming environment

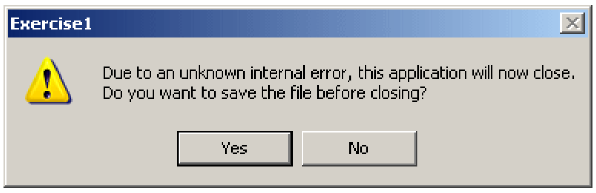
Message Box Icons

Besides the buttons, the message box can also display an icon that accompanies the message. Each icon is displayed by specifying a constant integer. The values and their buttons are as follows:

Value	Icon		Suited when
MB_ICONEXCLAMATION MB_ICONWARNING	⚠	⚠	Warning the user of an action performed on the application
MB_ICONINFORMATION MB_ICONASTERISK	ℹ	ℹ	Informing the user of a non-critical situation
MB_ICONQUESTION	?	?	Asking a question that expects a Yes or No, or a Yes, No, or Cancel answer
MB_ICONSTOP MB_ICONERROR MB_ICONHAND	✖	✖	A critical situation or error has occurred. This icon is appropriate when informing the user of a termination or deniability of an action

The icons are used in conjunction with the buttons constant. To combine these two flags, use the bitwise OR operator â€œ|â€. Here is an example:

```
AfxMessageBox(L"Due to an unknown internal error, this application will now close.\n"
           L"Do you want to save the file before closing?",
           MB_YESNO | MB_ICONWARNING);
```


Exercise1

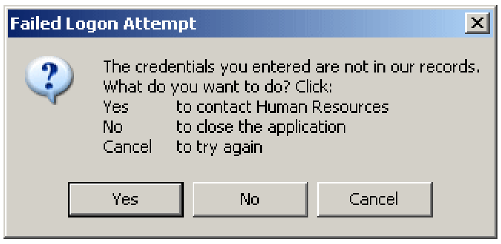
Due to an unknown internal error, this application will now close.
Do you want to save the file before closing?

Practical Learning: Showing an Icon on a Message Box

1. To display the message box with an icon, change **InitInstance()** as follows:

```
#include <afxwin.h>
#include "resource.h"

class CExerciseApp : public CWinApp
{
    BOOL InitInstance()
    {
 MessageBox(NULL,
  L"The credentials you entered are not in our records.\n"
  L"What do you want to do? Click:\n"
  L"Yes\tto contact Human Resources\n"
  L"No\tto close the application\n"
  L"Cancel\tto try again\n",
  L"Failed Logon Attempt",
         MB_YESNOCANCEL | MB_ICONQUESTION);

        return TRUE;
            }
    };

    CExerciseApp theApp;
```

2. Test the application


Failed Logon Attempt

The credentials you entered are not in our records.
What do you want to do? Click:
Yes   to contact Human Resources
No   to close the application
Cancel   to try again

3. Close it and return to your programming environment

The Default Button

When a message box is configured to display more than one button, the operating system is set to decide which button is the default. The default button has a thick border that sets it apart from the other button(s). If the user presses Enter, the message box would behave as if the user had clicked the default button. Fortunately, if the message box has more than one button, you can decide what button would be the default.

To specify the default button, add one of the following constants to the *uType* combination:

Value	If the message box has more than one button, the default button would be
MB_DEFBUTTON1	The first button
MB_DEFBUTTON2	The second button
MB_DEFBUTTON3	The third button
MB_DEFBUTTON4	The fourth button

To specify the default button, use the bitwise OR operator "|" to combine the constant integer of the desired default button with the button's constant and the icon.

The Message's Return Value

After using the message box, the user must close it by clicking a button on it. Clicking OK usually means that the user acknowledged the message. Clicking Cancel usually means the user is changing his or her mind about the action performed previously. Clicking Yes instead of No usually indicates that the user agrees with whatever is going on.

In reality, the message box only displays a message and one or a few buttons. The function used to create the message box returns a natural number that you can use as you see fit. The return value itself is a registered constant integer and can be one of the following:

Displayed Button(s)	**If the user clicked**	**The return value is**
OK	OK	IDOK
OK Cancel	OK	IDOK
OK Cancel	Cancel	IDCANCEL
Abort Retry Ignore	Abort	IDABORT
Abort Retry Ignore	Retry	IDRETRY
Abort Retry Ignore	Ignore	IDIGNORE
Yes No	Yes	IDYES
Yes No	No	IDNO
Retry Cancel	Retry	IDRETRY
Retry Cancel	Cancel	IDCANCEL

❖Practical Learning: Returning a Value from on a Message Box

1. To get the returned value of a message box, change **InitInstance()** as follows:

```
#include <afxwin.h>
#include "resource.h"

class CExerciseApp : public CWinApp
{
    BOOL InitInstance()
    {
 int Answer;

 Answer = AfxMessageBox(L"Due to an unknown internal error, "
            L"this application will now close.\n"
            L"Do you want to save the file before closing?",
            MB_YESNO | MB_ICONWARNING | MB_DEFBUTTON2);

 if( Answer == IDYES )
  AfxMessageBox(L"You selected Yes");
 else // if( Answer == IDNO )
  AfxMessageBox(L"You selected No");

 return TRUE;
    }
};

CExerciseApp theApp;
```

2. Test the application
3. Close it and return to your programming environment