Displaying a Message Box

In Win32 console applications, you can use the MessageBoxA function to display a message box to the user. The MessageBoxA function takes four parameters:

hWnd: The handle to the window that owns the message box. If this parameter is NULL, the message box will be created as a top-level window.

lpText: A pointer to the text to display in the message box.

lpCaption: A pointer to the caption of the message box.

uType: A bit-mapped integer that specifies the type of message box to display. The uType parameter can be used to specify the buttons to display, the icon to display, and the default button.

The following table shows some of the possible values for the uType parameter:

Value	Integer Value	Description
MB_OK	0	Display a message box with an OK button.
MB_OKCANCEL	1	Display a message box with OK and Cancel buttons.
MB_YESNO	4	Display a message box with Yes and No buttons.
MB_YESNOCANCEL	3	Display a message box with Yes, No, and Cancel buttons.
MB_RETRYCANCEL	5	Display a message box with Retry and Cancel buttons.
MB_ABORTRETRYIGNORE	2	Display a message box with Abort, Retry, and Ignore buttons.
MB_CANCELTRYCONTINUE	6	Display a message box with Cancel, Try Again, and Continue buttons.
MB_ICONSTOP	16	Display a message box with a stop-sign icon.
MB_ICONQUESTION	32	Display a message box with a question-mark icon.
MB_ICONINFORMATION	64	Display a message box with an information-symbol icon.
MB_ICONEXCLAMATION	48	Display a message box with an exclamation-point icon.

The table describes various message box constants and their corresponding descriptions.

These constants are often used in programming to customize the appearance and behavior of message boxes, which are dialog boxes used to display information or request user input in applications.

Clearer Table:

Value	Description
MB_OK	Displays message box with OK button
MB_OKCANCEL	Displays message box with OK and Cancel buttons
MB_YESNO	Displays message box with Yes and No buttons
MB_YESNOCANCEL	Displays message box with Yes, No, and Cancel buttons
MB_RETRYCANCEL	Displays message box with Retry and Cancel buttons
MB_ABORTRETRYIGNORE	Displays message box with Abort, Retry, and Ignore buttons
MB_CANCELTRYCONTINUE	Displays message box with Cancel, Try Again, and Continue buttons
MB_ICONSTOP	Displays message box with stop-sign icon
MB_ICONQUESTION	Displays message box with question-mark icon
MB_ICONINFORMATION	Displays message box with information-symbol icon
MB_ICONEXCLAMATION	Displays message box with exclamation-point icon

Yes, the values listed in the table are often used as integer constants to specify the "uType" parameter when creating message boxes in programming.

The "uType" parameter is an integer value that determines the type and behavior of the message box.

You can use the uTypes in your programming language like:

```python
import ctypes

# Display a message box with an OK button
ctypes.windll.user32.MessageBoxW(0, 'This is an example', 'MessageBox', 0x00000000)

# Display a message box with Yes and No buttons
ctypes.windll.user32.MessageBoxW(0, 'Question?', 'MessageBox', 0x00000004)

# Display a message box with an exclamation-point icon
ctypes.windll.user32.MessageBoxW(0, 'Warning!', 'MessageBox', 0x00000030)
```

Or

```c
#include <windows.h>

int main() {
    ;Display a message box with an OK button
    MessageBoxW(NULL, L"This is an example", L"MessageBox", MB_OK);

    ;Display a message box with Yes and No buttons
    int result = MessageBoxW(NULL, L"Question?", L"MessageBox", MB_YESNO);

    ;Check the user's response
    if (result == IDYES) {
        ;User clicked Yes
    } else if (result == IDNO) {
        ;User clicked No
    }

    ;Display a message box with an exclamation-point icon
    MessageBoxW(NULL, L"Warning!", L"MessageBox", MB_ICONEXCLAMATION);

    return 0;
}
```

In this C program, we use different message box types, including MB_OK, MB_YESNO, and MB_ICONEXCLAMATION.

You can specify the desired message box type by passing the corresponding integer value as the third parameter to the MessageBoxW function.

The program also checks the user's response to the "Yes" and "No" buttons by examining the result returned by the function.

Or

```c
#include <windows.h>

int main() {
    ;Display a message box with an OK button
    MessageBoxW(NULL, L"This is an example", L"MessageBox", 0);

    ;Display a message box with Yes and No buttons
    int result = MessageBoxW(NULL, L"Question?", L"MessageBox", 4);

    ;Check the user's response
    if (result == 6) {
        ;User clicked Yes
    } else if (result == 7) {
        ;User clicked No
    }

    ;Display a message box with an exclamation-point icon
    MessageBoxW(NULL, L"Warning!", L"MessageBox", 48);

    return 0;
}
```

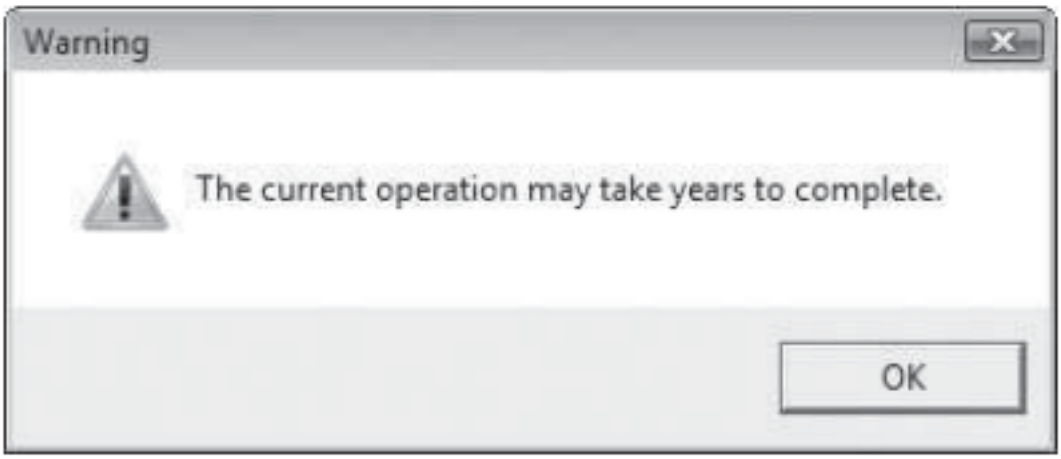
In this code, we're using the integer values directly to specify the message box types.

For example, 0 corresponds to MB_OK, 4 corresponds to MB_YESNO, and 48 corresponds to MB_ICONEXCLAMATION.

You can use these values to create message boxes with the desired type and behavior in your C program.

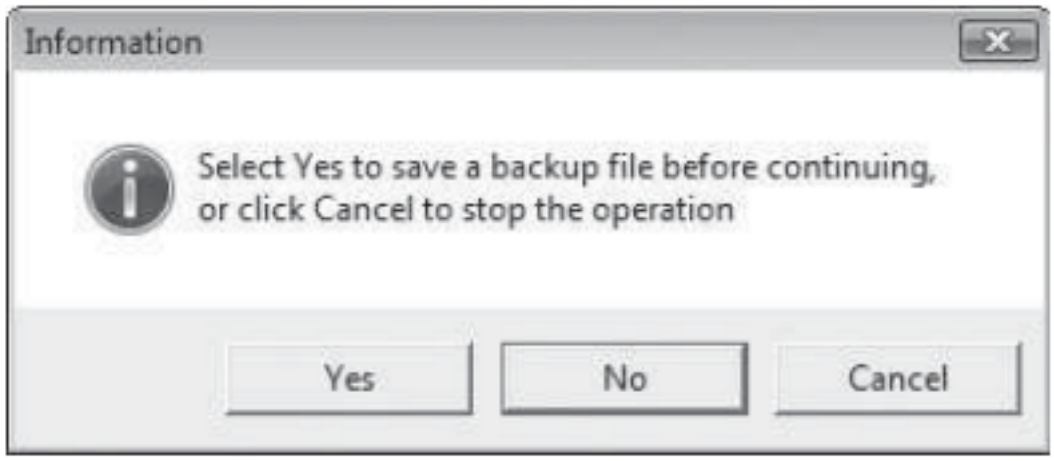
Demonstration Program

We will demonstrate a short program that demonstrates some capabilities of the MessageBoxA function. The first function call displays a warning message:

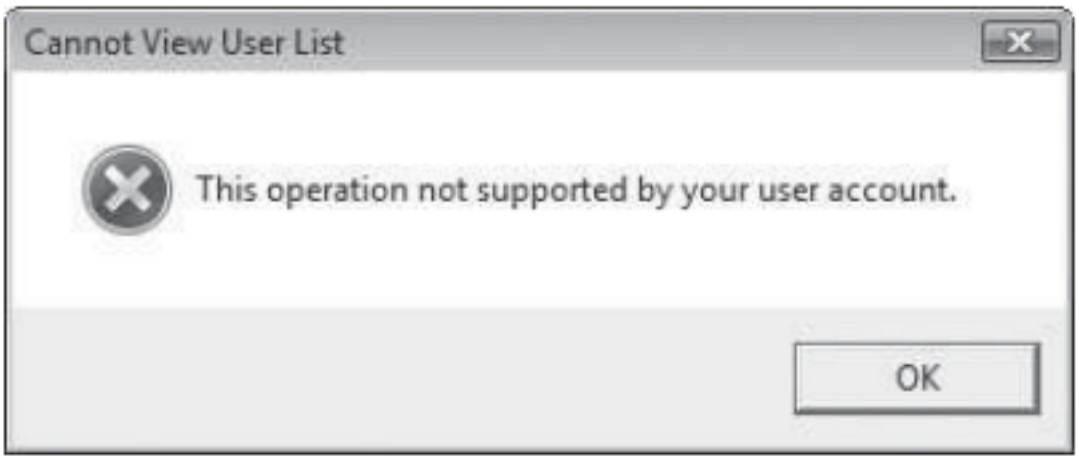


The second function call displays a question icon and Yes/No buttons. If the user selects the Yes button, the program could use the return value to select a course of action:

The third function call displays an information icon with three buttons:



The fourth function call displays a stop icon with an OK button:




```
; Demonstrate MessageBoxA (MessageBox.asm)
INCLUDE Irvine32.inc

.data
captionW BYTE "Warning",0
warningMsg BYTE "The current operation may take years to complete.",0

captionQ BYTE "Question",0
questionMsg BYTE "A matching user account was not found.",0dh,0ah,"Do
you wish to continue?",0

captionC BYTE "Information",0
infoMsg BYTE "Select Yes to save a backup file before continuing,",
0dh,0ah,"or click Cancel to stop the operation",0

captionH BYTE "Cannot View User List",0
haltMsg BYTE "This operation not supported by your user account.",0

.code
main PROC
; Display Exclamation icon with OK button
INVOKE MessageBox, NULL, ADDR warningMsg, ADDR captionW, MB_OK +
MB_ICONEXCLAMATION

; Display Question icon with Yes/No buttons
INVOKE MessageBox, NULL, ADDR questionMsg, ADDR captionQ, MB_YESNO +
MB_ICONQUESTION

; interpret the button clicked by the user
cmp eax,IDYES ; YES button clicked?

; Display Information icon with Yes/No/Cancel buttons
INVOKE MessageBox, NULL, ADDR infoMsg, ADDR captionC, MB_YESNOCANCEL
+ MB_ICONINFORMATION + MB_DEFBUTTON2

; Display stop icon with OK button
```

```
INVOKE MessageBox, NULL, ADDR haltMsg, ADDR captionH, MB_OK +
MB_ICONSTOP

exit
main ENDP
END main
```

The provided code is a demonstration program in assembly language for the Windows API. It showcases the use of the MessageBox function to create message boxes with different icons and button options. Here's a clear and concise explanation of the code:

The program starts with including the Irvine32 library, which provides various macros and functions for 32-bit assembly programming.

In the .data section, several strings are defined for different message boxes, including captions and messages.

These strings are null-terminated with a '0' byte at the end.
The .code section contains the main procedure (main PROC) where the demonstration begins.

Four message boxes are created using the INVOKE macro, each with different options: A warning message box with an exclamation icon and an OK button.

A question message box with a question icon and Yes/No buttons. An information message box with an information icon and Yes/No/Cancel buttons, with the default button set to No.

A stop message box with a stop icon and an OK button. After displaying each message box, the program checks the result (in the eax register) to determine which button was clicked by the user.

It compares the result with predefined constants like IDYES to check if the "Yes" button was clicked.

Finally, the program exits after displaying all the message boxes.

In summary, this program demonstrates how to create message boxes with different icons and button options using the MessageBox function

from the Windows API.

It also shows how to interpret the user's response by comparing the result with predefined constants for button IDs.