Callbacks

-
Alyce

Callbacks are one of the more complex aspects or Liberty BASIC programming. If you are new to programming, or if you do not have a thorough knowledge of making API calls, then you might want to put this article away for future consideration.

A "callback" statement in Liberty BASIC is the address of a program function that is used as an argument in API call. The API function contacts your program's function and sends it information in the form of an argument list. Your program's function then performs the desired action. Your program's function may make use of the items in the list. When your program's function is finished it must return a value to the calling function. If it returns 0, the calling function will stop itself from running and return control to your program. Your program's function should return nonzero if it wants to let the API function to continue to execute.

API calls with "Enum" in their names generally require callbacks. These functions will enumerate some system entities, sending the list to your function one at a time. Your function processes each of these items as they are sent to it by the API function. When the input parameters from the API function give no further information to your program's function, or when your program has processed a set maximum number of items, then it returns 0 to the calling API function to cause the API function to quit execution and return control to your program.

Syntax

The syntax is:

```
callback addressPTR, functionName(type1, type2...), returnValue
```

Usage

The parts of the callback statement are explained individually below.

addressPTR

This assigns a name to the memory address of the function. This memory address is passed into the API function that requires a callback.

functionName

This is the name of the function in the Liberty BASIC program that will be called by the API function.

(type1, type2...)

The type list is a comma-separated list of parameters which will be specific to the function used. The parameters must be valid data TYPES such as "ulong" and "long". API functions require different numbers of parameters, depending upon the individual function.

returnValue

This is the type of the return value listed after the closing parenthesis. The Liberty BASIC function may return a value to the calling function. Some calling functions evaluate this return and when YOUR function returns 0 to the calling function the API stops itself from running and returns control to your program.

Functions Requiring Callbacks

Some API functions that require callbacks:

- EnumChildWindows
- EnumFontFamilies
- EnumWindows
- EnumPrinters

Look in the documentation provided by Microsoft in its Software Developers Kits or on the MSDN to discover the correct format for the function you must provide for the API function. The functions you must provide for the above sample list are:

- EnumChildWindowsProc
- EnumFontFamiliesProc
- EnumWindowsProc

- EnumPrintersProc

For a working example of using an Enum Function with a callback, see the helpfile "EnumWindows" example in the CALLBACK section.

Generic Demo

Here is a generic program to demonstrate callbacks. This is NOT a REAL program!

```
'generic, non-working code example:
texteditor #win.te, 10, 10, 250, 250
open "Enum Something Example" for window as #win
print #win, "trapclose [quit]"

'set the variable named address to be the memory address for
'EnumSomethingProc() using types long and ulong, and set
'the return type of EnumSomethingProc() to be a long

callback myAddress, EnumSomethingProc(long, ulong), long

open "mydll" for dll as #dummy

'call EnumWindows, which in turn calls back into the
'BASIC function at address.

calldll #dummy, "EnumSomething", _
    myAddress as ulong, _
    0 as long, _
    result as long

close #dummy

wait

[quit]
close #win
end

function EnumSomethingProc(wParam, lParam)
    print #win.te, "wParam is ";wParam
    print #win.te, "lParam is ";lParam
    'check conditions to choose a return
    'value based upon the param list values
    'returning 0 causes EnumSomething to
    'return control to the program

    if lParam=0 or wParam=0 then
        EnumSomethingProc = 0
      else
        EnumSomethingProc = 1
    end if

    'if your program returns nonzero to
    'EnumSomething, then it will continue
    'running and send your function
    'another callback and list of params.

end function
```

Working Demo

This is a variation on the enumwindows.bas program that comes with Liberty BASIC. The called function is allowed to continue processing the information about windows until no more windows are found. We check for this by looking for a handle value of 0. We give ourselves the handle information in a texteditor, just so we can see what is happening.

```
    texteditor #win.te, 10, 10, 250, 250
    open "Enum Windows Example" for window as #win
    print #win, "trapclose [quit]"

    'set the variable named address to be the memory address for
    'enumWndProc() using types handle and ulong, and set
    'the return type of enumWndProc() to be a long
    callback address, enumWndProc(handle, ulong), long

    'call EnumWindows, which in turn calls back into the
    'BASIC function at address.
    calldll #user32, "EnumWindows", _
        address as ulong, _
        0 as long, _
        result as long
    wait
```

```
[quit]
    close #win
    end

function enumWndProc(hwnd, lparam)
    if hwnd<>0 then
        #win.te "Window handle is ";hwnd
        enumWndProc = 1
    else
    'returning 0 causes EnumWindows to return
    ' control to your program
        enumWndProc = 0
    end if
    end function
```

Callbacks | Syntax | Usage | addressPTR | functionName | (type1, type2...) | returnValue | Functions Requiring Callbacks | Generic Demo | Working Demo