Writing an editor that can handle multiple files isn't that hard! In fact, it is quite easy once you know the trick.

First, you need to dimension some arrays:

```
dim TabName$(1), TabFile$(1), TabContents$(1)
dim tempArray1$(1), tempArray2$(1), tempArray3$(1)
```

TabName$() will hold the display name of the tab. For my demo, the user is asked to pick a name for each tab, but you could easily change it to display the filename only, or some other naming scheme. Note that while I refer to "tabs", the demo uses a listbox, which requires much less code. It wouldn't be too hard to do tabs in your own program, though.

TabFile$() contains the file name of the tab. Don't forget to store this! TabContents$() holds the contents of each tab's file in memory. This is important, because the files shouldn't be saved automatically when the user switches tabs.

tempArray1$(), tempArray2$(), and tempArray3$() are used in an important part of the program which dynamically rezizes the data arrays.

Now that the arrays are dimensioned, we should enter some sample data. Also, it is important to save the number of tabs, and the tab currently being edited. Here is the code for that:

```
TabCount = 1
CurrentTab = 1
TabName$(1) = "Sample Tab"
TabFile$(1) = ""
TabContents$(1) = "This is a tab!"
```

Now we can set up the GUI. Note that I am using an LB listbox, and not real tabs. If you'd like your program to have tabs, then it could be modified to do so.

ss.png

- Details
- Download
- 36 KB

```
WindowWidth = 650
```

```
WindowHeight = 540
UpperLeftX = int((DisplayWidth-WindowWidth)/2)
UpperLeftY = int((DisplayHeight-WindowHeight)/2)
menu #main, "&File", "New Tab", [NewTab], "&Open...", [OpenFile], |, "
E&xit", [Quit]
menu #main, "Edit"
menu #main, "Help", "About the Multi-File Editor", [About]
button #main.newtab, "New Tab", [NewTab], UL, 0, 0, 65, 25
button #main.openfile, "Open", [OpenFile], UL, 65, 0, 65, 25
button #main.tabfile, "File Name", [FileName], UL, 130, 0, 65, 25
listbox #main.tablist, TabName$(), [SelectTab], 0, 25, 220, 462
texteditor #main.te, 220, 25, 424, 462
open "Multi-File Editor Demo" for window as #main
#main "trapclose [Quit]"
#main "resizehandler [Resize]"
#main.tablist "selectindex 1"
gosub [updateTabs]
```

Now, the guts of the code. When a new tab is opened, we must:

- Ask the user to name it
- (if opening a file) Ask the user what file to open
- increase the size of the arrays to hold the extra tab
- add the data
- increment the tab count

This is pretty complex, at first glance. We can use LB's prompt and filedialog commands for the first two. The third can be accomplished with the temporary arrays and a for/next loop (see code below). The last two are simply setting variables. Here is the code used to create a new tab:

```
tabName$ = ""
prompt "What you like to call the new tab?"; tabName$
if tabName$ = "" then wait
TabCount = TabCount + 1
gosub [resizeArray]
TabName$(TabCount) = tabName$
CurrentTab = TabCount
#main.tablist "reload"
#main.tablist "selectindex "; CurrentTab
gosub [updateTabs]
```

The resizeArray subroutine, which resizes the data arrays:

```
[resizeArray]
redim tempArray1$(TabCount-1)
redim tempArray2$(TabCount-1)
redim tempArray3$(TabCount-1)
for x = 1 to TabCount - 1
tempArray1$(x) = TabName$(x)
tempArray2$(x) = TabFile$(x)
tempArray3$(x) = TabContents$(x)
next x
redim TabName$(TabCount)
redim TabFile$(TabCount)
redim TabContents$(TabCount)
for z = 1 to TabCount - 1
TabName$(z) = tempArray1$(z)
TabFile$(z) = tempArray2$(z)
TabContents$(z) = tempArray3$(z)
next z
return
```

The updateTabs subroutine, which updates the contents of the editor based on the selected tab:

```
[updateTabs] 'new tab selected... load it into the editor
#main.tablist "selectionindex? CurrentTab"
#main.te "!cls"
#main.te TabContents$(CurrentTab)
return
```

The rest is quite easy to grasp. Hopefully your knowledge has been expanded some. Here is the complete source code, ready to run:

```
nomainwin

'initialize the tab information
dim TabName$(1), TabFile$(1), TabContents$(1) 'data associated with a
praticular tab
dim tempArray1$(1), tempArray2$(1), tempArray3$(1) 'one for each data
array; used when resizing the arrays
TabCount = 1
CurrentTab = 1
TabName$(1) = "Sample Tab"
TabFile$(1) = ""
TabContents$(1) = "This is a tab!"

'struct to help resize the widgets
```

```
struct rcClient, top as long, left as long, right as long, bottom as l
ong

'setup the gui
WindowWidth = 650
WindowHeight = 540
UpperLeftX = int((DisplayWidth-WindowWidth)/2)
UpperLeftY = int((DisplayHeight-WindowHeight)/2)
menu #main, "&File", "New Tab", [NewTab], "&Open...", [OpenFile], |, "
E&xit", [Quit]
menu #main, "Edit"
menu #main, "Help", "About the Multi-File Editor", [About]
button #main.newtab, "New Tab", [NewTab], UL, 0, 0, 65, 25
button #main.openfile, "Open", [OpenFile], UL, 65, 0, 65, 25
button #main.tabfile, "File Name", [FileName], UL, 130, 0, 65, 25
listbox #main.tablist, TabName$(), [SelectTab], 0, 25, 220, 462
texteditor #main.te, 220, 25, 424, 462
open "Multi-File Editor Demo" for window as #main
#main "trapclose [Quit]"
#main "resizehandler [Resize]"
hMain = hwnd(#main)
#main.tablist "selectindex 1"
gosub [updateTabs]
wait

[Quit] 'close the program
close #main
end

[Resize] 'adjust the size of the window
calldll #user32, "GetClientRect", hMain as ulong, rcClient as struct,
result as long
nMainHeight = rcClient.bottom.struct-25
nEditorWidth = rcClient.right.struct-220
#main.te "!locate 220 25 "; nEditorWidth; " "; nMainHeight
#main.tablist "locate 0 25 220 "; nMainHeight
#main "refresh"
wait

[NewTab] 'create a new, empty tab
tabName$ = ""
prompt "What you like to call the new tab?"; tabName$
if tabName$ = "" then wait
TabCount = TabCount + 1
gosub [resizeArray]
TabName$(TabCount) = tabName$
```

```
CurrentTab = TabCount
#main.tablist "reload"
#main.tablist "selectindex "; CurrentTab
gosub [updateTabs]
wait

[OpenFile] 'open a file and put it in a new tab
tabName$ = ""
file$ = ""
filedialog "Open a file", "*.*", file$
prompt "What you like to call the file?"; tabName$
if file$ = "" or tabName$ = "" then wait
TabCount = TabCount + 1
gosub [resizeArray]
TabName$(TabCount) = tabName$
TabFile$(TabCount) = file$
TabContents$(TabCount) = loadFile$(file$)
CurrentTab = TabCount
#main.tablist "reload"
#main.tablist "selectindex "; CurrentTab
gosub [updateTabs]
wait

[SelectTab] 'A new tab has been selected
gosub [saveCurrentTab]
gosub [updateTabs]
wait

[FileName] 'Show the filename associated with the current tab
notice "This tab's filename is..." + chr$(13) + TabFile$(CurrentTab)
wait

[About] 'show an about message
message$ = "The Multi-File Editor is a demo on how to create a multipl
e-file editor in Liberty BASIC. "
message$ = message$ + "Kudos to the LB Community for putting up with m
y insane ramblings and letting "
message$ = message$ + "me try to help out. " + chr$(13) + chr$(13) + c
hr$(13) + "Use or abuse as you wish."
notice "About the Multi-File Editor" + chr$(13) + message$
wait

[resizeArray] 'resize the arrays for a new tab
redim tempArray1$(TabCount-1)
redim tempArray2$(TabCount-1)
redim tempArray3$(TabCount-1)
```

```
for x = 1 to TabCount - 1
 tempArray1$(x) = TabName$(x)
 tempArray2$(x) = TabFile$(x)
 tempArray3$(x) = TabContents$(x)
next x
redim TabName$(TabCount)
redim TabFile$(TabCount)
redim TabContents$(TabCount)
for z = 1 to TabCount - 1
 TabName$(z) = tempArray1$(z)
 TabFile$(z) = tempArray2$(z)
 TabContents$(z) = tempArray3$(z)
next z
return

[saveCurrentTab] 'save the contents of the editor to memory
#main.te "!contents? text$"
TabContents$(CurrentTab) = text$
return

[updateTabs] 'new tab selected... load it into the editor
#main.tablist "selectionindex? CurrentTab"
#main.te "!cls"
#main.te TabContents$(CurrentTab)
return

function loadFile$(file$) 'get the contents of a file
open file$ for binary as #f
loadFile$ = input$(#f, lof(#f))
close #f
end function

sub saveFile file$, text$ 'save text$ to file$
open file$ for output as #f
print #f, text$
close #f
end sub
```