Help (... is on the way!)

by Jerry Muelver - -
jmuelver- http://hytext.com/

Table of Contents

Offering Help

A quick way to access Help files for your LB app is to point the user's Web browser to an HTML file with a RUN statement. (That is to say, it's quick if you are adept at writing HTML files. See Note 1)

Once you've got a productive, effective method for producing HTML (see Note 1), the hard work begins. How do you know what to put into a Help file, and how do you know it will actually help?

From your own experience with computer applications, you already know that a Help file's main sections are typically:

- Table of contents

- Overview, description of what the program is designed to do.
- Controls, detailed list and description of each menu item, control button, and entry field in the program
- Functions, how to make the program work to accomplish its intended tasks
- Errors and recovery, how know what's broken, and how to fix it
- Index

People who have done this sort of work before already have a good idea how to go about building Help files. Let's ignore those folks, and lay out the simplest way possible for complete beginners to build Help files. The basic assumption is that we'll create a Help system in HTML (to take advantage of Web-style built-in display features), and that those of us who do not automatically write error-free HTML markup in our sleep will use an HTML or Help file editor of some sort (see Note 1). I'll explain what to do with the tools, rather than how the tools work, because (1) I don't know which authoring tool you have chosen, and (2) those who choose WikiWriter will be productive in ten minutes or so, anyway.

Table of Contents

The process of writing Help files is iterative. You start small, and keep building and refining as you go until the job is done. The basic approach is to create a label for something, then describe the something you just labeled.

Begin with a Table of Contents (TOC). The minimum TOC will simply point to the main sections. Build a list of links for those sections like this to get started:

- Overview
- Menus and Controls
- Functions
- Troubleshooting
- Index

Now your TOC is done, at least for now. You can always add more links when you have more sections, sub-sections, and topic pages to play with.

Overview

On your overview page, start with just three paragraphs, each describing one of the following, in four sentences or

- What the program does
- What is special about this program compared to the competition
- Who developed the program, and how to contact the developer

If you keep this section short, people will actually read it.

Menus and Controls

We all know your program is designed with a completely intuitive interface which really needs no explanation. But just in case your program winds up in the US Congress some day, you should probably explain the obvious anyway.

Menus are easy to document. Just build a list in outline fashion that shows your menu structure:

- File
    - New
    - Open
    - Quit
- Edit
    - Copy

Now add an explanatory phrase after each menu command. Keep it short and sweet. Make the explanation read more like a ToolTip than a magazine article. Avoid lecturing: "This option, when dropped down from the menu bar and selected by the user, will activate the Operating System's file selection browser in 'Open' mode, enabling...."

Your users want information, not Victorian Romance, so try it like this:

- File
    - New - Create new file
    - Open - Find and open existing file
    - Quit - End program
- Edit
    - Copy - Copy selected text to memory

Do the same for controls. Use their button text or names if they are labeled. For graphics controls, take the time do do screen captures, or re-use the button graphics in your list so your user knows exactly what you're talking about. Group related controls together in sublists.

Functions

You might call this section Actions, or Tasks, or Features, if you like. Now that you've described the menus and controls, you are free to refer to them in a series of condensed instructions on their use. But, how do you explain how use the program's features without sounding like a geek out of water? You use

"TWIT" -- The Whispered Instruction Technique.

To use TWIT, imagine you have a clueless user sitting at the keyboard, all set to trash your program by doing the Wrong Thing at the Wrong Time. Your job is to prevent disaster by whispering instructions into the user's ear just before Something Awful happens. You can't grab the mouse, or poke at keys, or pull the power cord plug out of the wall. You can only whisper instructions, and jot them down for your Help file as you whisper.

Start with a list (familiar, comfortable refrain, isn't it?) of the jobs you want TWIT to do. Your list should cover the range of actions your program supports. If some of the actions are large-scale ("Build House"), use sublists to break them down for TWIT ("Build House > Clear land, dig basement, pour foundation, set footers, assemble wall frames..."). Two basic formats are the narrative:

- Open a file - Select File > Open. Use the file browser to find a file. Double-click on the file name to open the file, or single-click on the name to select the file and then click on Open.

and the sequence (or "recipe"):

    - Move text

1. Select the text with the mouse click-and-drag.
2. Release the mouse button.
3. Move the mouse cursor into the area of the selected text.
4. Click-and-hold the right mouse-button.
5. Holding the button down, move the cursor to drag the text to the new location.
6. When the text is positioned where you want want it, release the mouse button.
7. To undo the move, in case it didn't work the way you wanted, press Control-Z (press and hold Control while you press and release Z).

I usually build the outline of jobs as a list of links, and apply TWIT to each job on a separate page. It's easier to scan a list of jobs and pick out the specific one you need than it is to find it by scrolling though long, dense text like this article.

Troubleshooting

Name this section whatever you wish, as long as the users can identify this as the place to go if something unexpected happens. I once had a client who banned "troubleshooting" from our dictionary of authorized technical terms because that term could give someone the idea that people could have trouble with the product. That was about the time I came up with "TWIT".

This section holds your error messages and explanations, typical user-caused glitches and recoveries, and problem diagnostic procedures. Once again, use lists to organize your material, and keep it short and to-the-

point.

Index

Indexing is a special art, and not for timid to attempt. With a properly detailed TOC and readable lists for controls and functions, you can get by without an index. But for really large, complex programs and systems, an index is essential. Get experts to help on this effort, unless you have tools that can make indexing easy to do (see Note 1).

Packaging

One way to package your Help system is to deliver it as a bunch of HTML files. If you put your application in a Zip file, just add the HTML files and any supporting graphics files to your application's main directory. Hook your app's Help button up to a RUN call to the first file in your set. This works even if the user has some browser other than MS Internet Explorer.

Another way is to compile all the Help files into a single MHT-format file, and point your Help button to that file with a RUN command to trigger the user's Internet Explorer. Only MSIE reads MHT format smoothly. See Note 1.

You can go the whole 9 yards and compile your system into Windows HtmlHelp, with the proper tools. That process is easier to do than it is to explain, once you've acquired the tools and some mildly frustrating experience, so I won't cover it here.

What to do next

Since you started simply, and organized tightly, it's not difficult to enhance and expand your Help system with such goodies as Tutorials, Tips and Tricks, Templates, or whatever else your user base demands. Don't get seduced into a bunch of tricky layouts and magic secondary pop-up windows and tooltip glitz. If you need those things, you're already working for a big-big company with lots of resources and money to waste, which means it's probably a good idea to out-source the Help file production to someone who already knows all the tricks and secrets of tech writing, so you can concentrate on programming.

For an example of how a Help system can be simple and direct, and still do the job, take a look at http://hytext.com/iwiki/ which is the posted HTML version of the Help files for WikiWriter, generated by WikiWriter (see Note 1).

- Note 1: If you're not a wonderful HTML coder (or if you are, and you really want to fly!),

take a look at WikiWriter. It is a free utility that produces HTML files from your simple tag-marked text files. WikiWriter will also compile your entire directory full of Help files into a single MSIE-compatible MHT file, graphics and everything, so you only have one file to distribute.

Help files aren't the only thing you can do with the WikiWriter authoring system. You can run it to experiment with PIM applications, free-form textbase stuff, web site design, project and program specification and wire-frame design, and all the other writing jobs that settle on the shoulders of programmers.

See what the WikiWriter way of working means for your programming and your program users by checking it out at http://hytext.com/ww. WikiWriter can do for your writing what Liberty Basic has done for your programming.

Note 2: The rumor that Microsoft is using WikiWriter internally for project documentation on the .NET product line is, as yet, unverified, and must be regarded as possibly untrue.

Table of Contents