Assignment 3

Overview

This exercise is designed to give you practice using the `GtkTextView` widget and the `GtkHPaned` widget, among other things. Your program will open text files that contain special tags that either format the text or bookmark the text. The text itself will be displayed in the right pane of the horizontal paned widget. When the program finds formatting tags it will use them to display the text with the specified formatting. When it finds bookmark tags, it will display the list of bookmarked text items in the left pane. Both panes will be scrollable in both dimensions. The scrolling should be automatic – i.e., scroll bars are visible only if the text is larger than the display area.

The tags are bracketed, just like html, and will be one of:

Bold	<b> ... </b>
Italic	`<i> ... </i>`
Monospace	`<tt> ... </tt>`
Bookmark	`<h> ... </h>`

The tags in the program will not overlap but they may be nested. Thus, there will not be a sequence such as *<b>text<tt>text</b>text</tt>*, but there might be sequences such as in *<b>this is bold<i> and this is bold italic </i> but this is unitalicized bold <tt> and now it is bold Monospace</tt></b>*. Bookmark tags will not be nested and will not overlap, but bookmarks might enclose or be enclosed by formatting tags. When the program reads the input file, it should find all bookmarked text and create a list of the bookmarks and display them in the left pane. For example, if the text contains

```
    <h>Chapter 1</h> This is the start of it all. and so on
    <h>Chapter 2</h> and the saga continued, much to the chagrin of ...
```

then the program will store the text "Chapter 1" and "Chapter 2" as separate lines in the left pane. The text itself will be displayed in the right pane, without the tags. In other words, the tags will be invisible text, or perhaps deleted text, depending upon how you choose to implement this. In place of the tags, the text will have the formatting applied to it in accordance with the meaning of the tags.

There is no need for this text to be editable, so you should probably set the editable property to `FALSE` in the text view. If it is editable, then you have to deal with how the inserted or deleted text affects the formatting tags, which gets more complex.

The program should also have the following buttons either at the top or bottom of the top-level window:

Open opens a file chooser dialog to choose text files to open as input.

Close closes the currently open file, displaying just blank panes upon closing it.

Quit quits the application.

The buttons should be stock items, nonresizable. The main window title should change when a file is loaded, to the name "`File Viewer:`" followed by the name (not pathname) of the loaded file. If the selected file cannot be loaded, for whatever reason, the program should display an error dialog to the user.

Hints and Suggestions

The program does not have to, but ought to, make use of Glib's regex utilities. This will be the easiest way to find the tags. Look at the demo programs in the `demos/glib_demos` directory, and read the GLib API documentation.

You will have two choices of implementation – either loading the entire file into the `GtkTextBuffer` and hiding tags and searching for the tags after it is loaded, or storing it in memory first, parsing it for tags, and then loading it into the buffer. Think about which choice will be easier for you and then plan it all out.

Other Requirements

Your source code must satisfy the requirements stated in the *Programming Rules* document on the course website. Please review those rules if you are not familiar with them. It will be graded using the following rubric:

Component	**Percent of Grade**
Correctness (meeting the functional requirements)	60
Documentation	15
Structure (modularity and design)	15
Program Style	10

Submission

If the program consists of more than one source code file, create a directory named `hwk3_username`, with permission `0700`, and put all of your source code files and a makefile into it (and nothing else!) In he makefile, make sure the executable gets named `file_viewer`. If it is a single source code file, name it `hwk3_username.c` or `hwk3_username.cc`, where *username* is the username you have on our UNIX system, and the extensions are for C and C++ respectively, with permission `0600`. Put the directory or the program into the directory

```
/data/biocs/b/student.accounts/cs493.70/projects/project3
```

on our file system.