SEN-103 Programming Multi-Module Appllications
Assessment Activity SEN-103:00010, SEN-103:00020
Creating a Multi-Module Executable
Creating a System with Multiple Communicating Executables
Last update: 24 January 2024

Assignment Instructions

Write a program that solves **one** of the two problems described below. Test your program thoroughly. Bugs will reduce your assessment score.

Follow the coding standards. If you violate the coding standards, this will lower your assessment score.

Follow instructions. The detailed instructions below contain a variety of requirements and constraints that your program ***must*** satisfy. If your program does not satisfy these requirements, this will lower your assessment score.

Submit your work ***as a Zip file that contains all the code, screen shots, and other information necessary to build and run your application***. This must include a Makefile or a script that will compile and link the C, Python and/or Java components of your system. You must also include a file called **README.txt** that provides instructions for how to create and run your program, and how to use the user interface.

Problem 1: Command-Based Graphics Editor

System Capabilities

Build a software system that allows the user to create simple drawings composed of geometric shapes, edit those drawings, save them, and display them. The system must provide the following functions:

1. Start a new drawing, specifying the width and height of the drawing area in pixels and the background color.
2. Add a shape, specifying the type of shape, the x,y location of the shape in pixels, the color of the shape, and whether the shape is filled or unfilled. You must support the following types of shape: a) rectangle, b) circle, c) ellipse, d) triangle. You will need to ask for additional information depending on the shape. For instance, for a circle you need the radius (assuming the x,y position will be the center). For a triangle, you will need two additional points (assuming the x,y position will be one point). And so on.
3. Delete a shape from the drawing. To do this, you will need to have a way to identify each shape. For instance, you could give each shape a unique number.
4. Move a shape. Change the shape position, without changing the other attributes of the shape.
5. List all shapes in the drawing, with their identifiers and some other attributes so the user can tell which one is which.
6. Display the current drawing.
7. Save the current drawing as a PNG image file with a user-supplied filename.

User Interface

You should decide the type of user interface you want to use for this program. Since this will mostly be terminal-based program, there are three main possibilities:

1. Use a series of menus and sub-menus;
2. Create a command language;

3. Use a hybrid of commands plus prompts.

The first option is similar to what we've been doing in previous assignments. To do the second option, you need to create a simple command language that will allow the user to express their intentions. For example:

newdrawing 500 450 white – to start working on a new drawing

triangle blue F 120 30 120 70 75 40 – to create a blue, filled triangle

delete 11 - to delete shape with the identifier 11 from the drawing

display – to display the current drawing

save mydrawing.png - to save the current drawing

exit – to leave the program.

And so on. If you choose this type of UI, your program would run in a loop, asking for commands, then interpreting and executing them.

The advantage of this type of command-based UI is that it is much more efficient for expert users. They need to type just one line in order to perform any action.

There are two disadvantages: 1) the user needs to remember the commands; 2) you, as the programmer, need to check that the commands are legal, that they contain all the necessary information and that the arguments are valid.

The third, hybrid approach would define commands for the main operations, then ask the user for additional details if necessary. For example:

```
Enter command: delete
        Delete what shape: 11
   - - - Shape 11 was deleted.
Enter command: triangle
    Color? Red
    Unfilled or filled (U/F)? F
    First point (x y)? 120 30
        Second point (x y)? 120 70
    Third point (x y)? 75 40
    - - - Created triangle with ID 19
Enter command: display
Enter command:
```

This approach reduces the memory load on the user and the amount of validation, but takes longer to create a new drawing item.

Program Structure Requirements

In order to evaluate whether you have mastered the skills tested by this assessment, your program must follow certain requirements.

- You ***must*** use the GD library to create the drawing images. You may use either the C or the Python bindings for this assignment.

- Your editor program ***must*** be broken into sensible modules that maximize cohesion and minimize coupling. You should probably have three or four separate modules (source files), each of which handles a different aspect of the program functionality.
- The drawing display function ***must*** be provided by a **separate executable**, an image viewer program that you will write. It is quite easy to write an image viewer in either Python or Java. *Only one drawing should be viewable at a time.* That is, if you have a drawing being displayed, and the user of the editor program chooses the display operation again, the prior drawing should be replaced by the new, current state of the drawing.

Suggestions

The typical way to implement a drawing editor like the one described above is to use a **display list** data structure. A display list is an ordered list (probably a linked list) of individual structures that represent the objects in a drawing. Newly created objects get added to the end of the list. When an object is deleted, it will be removed from the list.

The structures that are the elements of the list contain all information that would be needed to draw the shape in the image: the shape type, its location, whether it is filled or unfilled, and any other shape-specific information. The items on the display list should also include the item ID that is used to locate and delete items.

To display the entire drawing using GD, your program would do the following:

1. Free the old in-memory image.
2. Create a new in-memory image.
3. Start at the first item on the display list. For each item, add the appropriate shape with the appropriate attributes to the GD in-memory image.
4. When the entire display list has been processed, save the in-memory image as a temporary file.
5. Give the viewer a command to display this temporary file.

There are several ways that you can communicate with the image viewer. One way is to create simple command files written to a known, shared directory, as in the taxi dispatching lab exercise. Another is to use the operating system facilities to execute a command from within your main editor program. However, this requires that your editor program keeps track of whether the viewer is already live and displaying an image. If it is, your editor program must somehow shut down the old version of the viewer, before starting a new one (in order to satisfy the requirement that only one image should be visible at a time).

Of course, you can also use the display list to print the current contents of the drawing in text form, so the user can decide (for instance) which shapes they want to delete.

Problem 2: Console-Based Tourism Information Entry

System Capabilities

Build a system that will allow the user to create individual HTML files according to a standard format, then display the created file in a browser. Each file will be a single page that contains information about some tourist attraction. The contents must include:

- Page title (attraction title);
- Image of the attraction;
- Category of the attraction (e.g. temple, historical building, natural site, market, etc.)
- Location of the attraction - province and district;

- Location of the attraction - latitude and longitude coordinates;
- Description of the attraction - at least one paragraph.

You can find an example below.

Your program should have three main operations:

- **Create new attraction page.** This should bring up an ***ncurses*** form for entering the information listed above. For the image you can use a URL or a local image file.
- **Display attraction page.** This should show a list of all the available HTML files, let the user choose one, then invoke a browser to display the page.
- **Exit the program.**

The program should keep asking the user what operation to execute until the user selects the exit option.

Program Structure Requirements

In order to evaluate whether you have mastered the skills tested by this assessment, your program must follow certain requirements.

- You ***must*** use the *ncurses* (https://en.wikipedia.org/wiki/Ncurses) library to create the form for entering data. The original version of ncurses is written in C. Apparently there are bindings for a variety of other programming languages including Python. (I would recommend you use C to reduce the overall complexity of your environment, however.)
- Your program ***must*** be broken into sensible modules that maximize cohesion and minimize coupling. You should probably have three or four separate modules (source files), each of which handles a different aspect of the program functionality. For instance, you probably should create a separate module to write the HTML file.
- You ***must*** use a separate executable to display the HTML results. You can use a browser (Firefox, Chrome, etc.) but you must provide a way for the user to configure the browser to be used, since different computers will have different browsers installed. Do not assume that every user will have (for instance) Chrome. Alternatively, you can write your own simple HTML display program. This is easy to do in Java and probably in Python as well.

Example Attraction File

You can find a very basic sample file in the assignment instructions on Canvas. You are welcome to make your HTML files fancier and more attractive, although the focus of this assignment is on your software structure, not your results.

Here is a rendering of the sample file in the Firefox browser.

Wat Benchamabophit (Marble Temple)

Attraction Type: Religious site

Location: Dusit District, Bangkok

Geographic Coordinates: 13°45'59.7"N 100°30'50.7"E

Description: Wat Benchamabophit Dusitvanaram (Thai: วัดเบญจมบพิตรดุสิตวนาราม) is a Buddhist temple (wat) in the Dusit District of Bangkok, Thailand. Also known as the Marble Temple, it is one of Bangkok's best-known temples and a major tourist attraction. It typifies Bangkok's ornate style of high gables, stepped-out roofs and elaborate finials.

Construction of the temple began in 1899 at the request of King Chulalongkorn after building his palace nearby. The temple's name literally means 'the Temple of the fifth King located near Dusit Palace'. It was designed by Prince Naris, a half-brother of the king, and is built of Italian marble. It has display of Carrara marble pillars, a marble courtyard and two large singhas (lions) guarding the entrance to the bot. The interiors are decorated with crossbeams of lacquer and gold, and in shallow niches in the walls of paintings of important stupas all over the country. The cloister around the assembly hall houses 52 images of Buddha.