Assignments

CS3307 Individual Assignment - In progress

In progress
Submitted
Returned

Assignment Details

Title
CS3307 Individual Assignment
Due
Oct 3, 2023, 11:55 PM
Number of resubmissions allowed
Unlimited
Accept Resubmission Until
Oct 5, 2023, 11:55 PM
Status
Not Started
Grade Scale
Points (max 100.00)

Instructions

CS3307 Individual Assignment

Fall Session 2023

Purpose of the Assignment

The general purpose of this assignment is to develop a simple C++ program for a Unix/Linux system (like the Raspberry Pi Desktop), given a number of requirements, making use of the principles and techniques discussed throughout the course. This assignment is designed to give you experience in:

- object-oriented programming using C++, using basic language constructs, classes, and data types
- looking through Unix/Linux manual pages and documentation, as you will likely need to do this in your projects later
- getting acquainted with Unix/Linux-based programming and services, which will help in project development on this environment later in the course

The assignment is intended to give you some freedom in design and programming the explore the subject matter, while still providing a solid foundation and preparation for the type of work you will later be doing in the group project.

Assigned

Tuesday, September 12, 2023 (please check the main course website regularly for any updates or revisions)

Due

The assignment is due Tuesday, October 3, 2023 by 11:55pm (midnight-ish) through an electronic submission through the OWL site. If you require assistance, help is available online through OWL.

Late Penalty

Late assignments will be accepted for up to two days after the due date, with weekends counting as a single day; the late penalty is 20% of the available marks per day. Lateness is based on the time the assignment is submitted.

Individual Effort

Your assignment is expected to be an individual effort. Feel free to discuss ideas with others in the class; however, your assignment submission must be your own work. If it is determined that you are guilty of cheating on the assignment, you could receive a grade of zero with a notice of this offence submitted to the Dean of your home faculty for inclusion in your academic record.

What to Hand in

Your assignment submission, as noted above, will be electronically through OWL. You are to submit all source code files, header files, and build files necessary to compile and execute your code. If any special instructions are required to build or run your submission, you must include a README file documenting details. (Keep in mind that if the TA cannot run your assignment, it becomes much harder to assign it a grade.)

Assignment Task

Your assignment task is to develop a simple web application leveraging Wt, and familiarize yourself with development on a Unix/Linux System (like Raspberry Pi Desktop). While this might seem complex on the surface, it's surprisingly straightforward to do so. Wt is a C++-based web framework, capable of creating web app and REST services for querying information over HTTP. This should give you a good exposure to C++ classes, STL packages, external libraries/packages, and more -- all of which will be quite handy for the group project to come!

Getting An Environment Set Up

For this assignment, you will need access to a Unix-like environment of some kind. Using something like Raspberry Pi Desktop will give you something that most closely resembles doing things on an actual Raspberry Pi, but that is not required. You can use whatever distribution you wish as long as it supports the needs of this assignment. The options at your disposal, depend on your computer and operating system:

- If you are running Linux on your computer already, you're off to a good start. You'll just need the C++ compiler tools, CMake, Boost, and Wt. The compiler is quite likely installed by default, depending on your distribution, but CMake and Boost will likely need to be installed if you haven't already done so. (Make sure to install the development packages and not just the run-times so you can create your own programs!) The packages needed depend on your distribution, but a quick Googling should tell you what you need. You will also need to install Wt, and it is best to build from source following the instructions found here, particularly under Building and installing the Wt library.

- If you are running MacOS, you are also in pretty good shape, and should be able to run things from a Terminal or using XQuartz. You will need the command line compiler tools installed, which might depend on Xcode, as well as CMake and Boost, which will likely need to be installed. (This largely depends on the version of MacOS that you are running, and if you are using MacPorts or Homebrew for managing packages. Googling around will again tell you what you need depending on your setup.) If you have virtualization software (like VMWare or VirtualBox) with a Linux VM handy, that should work just as well too. You will also need to install Wt, and it is best to build from source following the instructions found here, particularly under Building and installing the Wt library.

- If you are running Windows, it is likely best to virtualize in some fashion to get Linux up and running. Virtualization software (like VMWare or VirtualBox) could work for this, but these days, my preferred approach is to use the Windows Subsystem for Linux or WSL. You can get a number of Linux distributions running under Windows quite readily with this, assuming that your computer supports things well. Once it's up and running, you just need compiler tools, CMake, and Boost, and you're ready to set up Wt. You will also need to install Wt, and it is best to build from source following the instructions found here, particularly under Building and installing the Wt library. (You could always use a live CD or live USB key to boot into Linux, but that's often not the most convenient way of doing things.)

- If you are using a Chromebook, you should be able to get Linux on it, depending on the age of your device. Again, compiler tools, CMake, and Boost and you're set to install Wt as noted above.

- If none of these are an option or work for you (for example because you have a Windows computer that doesn't support virtualization), we have a Linux server that you can connect into and use, with all the packages installed and ready to go. Simply use ssh to connect into cs3307.gaul.csd.uwo.ca with your Western credentials and you should be all set. You'll be doing things remotely, but it should still work fine for this assignment. To connect with your server from home, you would need to tunnel in using ssh, as we will demonstrate below.

Alright! So, now you have a Unix/Linux environment set up with the necessary compiler tools, CMake, Boost, and Wt. (As a bonus, this should also give you a nice dev environment for your group project too!) You have everything you need to get started with this assignment, so let's dig into the details of what you need to be doing here.

A Sample C++ Wt Hello World Application

To get you started, you will find a C++ Wt Hello World application attached to this assignment in a .zip file. This should allow you to test out your Wt environment and give you an initial code base that can be used for your assignment. This .zip file contains the following:

- README.txt: Brief documentation describing the contents of the file.
- Makefile: A build script that will compile things for you.
- main.cpp: The main function for the application.
- HelloApplication.h, HelloApplication.cpp: The main class for the application containing the Hello World logic.
- resources: A folder containing various resources commonly used by Wt applications.

To get started, you might need to adjust the Makefile according to how you configured and installed Wt. (It might also work fine as it is, and works fine on cs3307.gaul.csd.uwo.ca without modifications.). Simply typing "make" in the folder containing the Makefile will build things for you.

To run things after building, you can execute:

```
./hello --docroot . --http-listen 0.0.0.0:8080
```

at the command line. This runs the hello executable in your current directory, giving it a couple of parameters. Specifying --docroot . tells the application to find files it needs from the current directory. (Specifically, it will search for things in the resources folder in the current directory.) The remaining parameters (--http-listen 0.0.0.0:8080) tells the server to listen for incoming HTTP requests on any network interface on the system at port number 8080.

This will have a Hello World server up and running. To interact with things, you will need to connect up to it using your favourite web browser. If you are using a local installation of Wt that you set up yourself, simply typing in the URL localhost:8080 into your browser's address bar should connect you to this server and you are all set!

Running Servers on cs3307.gaul.csd.uwo.ca

If you are using our server for this assignment, there's a couple of things to keep in mind when running things. Not big deals, but things to keep in mind when using shared resources and using them remotely.

First, the port number that you have your server listen to has to be unique. Above we used port 8080, but if everyone in the class tried to use that port number for their server, it would work for the first person, but everyone else would get an error that the port is already in use. So, how to get a unique port number? You could just pick one randomly, but a a good trick is to select one based off of the numeric user id attached to your account. If you execute the command:

```
id -u
```

you will be given the unique numeric id attached to your account. If you take the last four digits of that number and add a 1 to the front, you will have a number greater than 10000 that should be available as a valid port number and somewhat unique to you. For example, if you were given a result if of 1583700123 above, the last four digits would be 0123. Adding a 1 to the front of this would give you a port number of 10123 to use for your work. Just substitute that in place of 8080 in the instructions above, and you're good to go.

Now the remote use part. For security reasons, access to most port numbers of our cs3307.gaul.csd.uwo.ca server are restricted off campus, and you would not be able to connect to your server using a URL of cs3307.gaul.csd.uwo.ca:10123 using our port number from above when home. The server is running there just fine, but a firewall would prevent the connection. So, how do we get remote access to our servers then? To accomplish this, we use an ssh tunnel.

Basically, an ssh tunnel allows us to create a secure channel to our remote server. We can then route requests to our server using this channel, and it will work as well as if everything was running locally. Suppose we are running our server remotely on port 10123 on cs3307.gaul.csd.uwo.ca as above. From a command line on our local computer, we can execute the following to set up our tunnel:

```
ssh -N -L 10123:cs3307.gaul.csd.uwo.ca:10123 cs3307.gaul.csd.uwo.ca
```

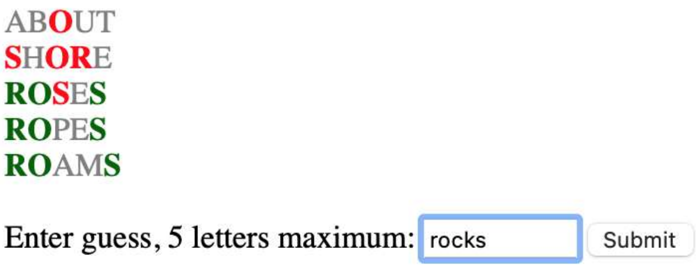
Let's break this down piece by piece. The -N option tells ssh that you do not want to create a remote shell with a command prompt on cs3307.gaul.csd.uwo.ca. We just want the tunnel, so we don't need that. The -L tells ssh that you're setting up a local tunnel, mapping a local port number to a remote one. In this case, we are mapping port 10123 on our local computer to port 10123 on cs3307.gaul.csd.uwo.ca. The last cs3307.gaul.csd.uwo.ca in the command is simply indicating the name of the server we are connecting to for this, cs3307.gaul.csd.uwo.ca. (If you need to specify an alternate user name when connecting to our server, you would do so on this second instance of cs3307.gaul.csd.uwo.ca, making it username@cs3307.gaul.csd.uwo.ca where username is your user id here.)

With this command running, you now have a tunnel set up. Any connections made to port 10123 on your local computer will now get redirected to our remote server. So, using localhost:10123 in the address bar of a web browser on your computer, will remotely connect up with our Wt server running on cs3307.gaul.csd.uwo.ca at that same port number. Doing this, you can access your server remotely without any difficulties!

The Application

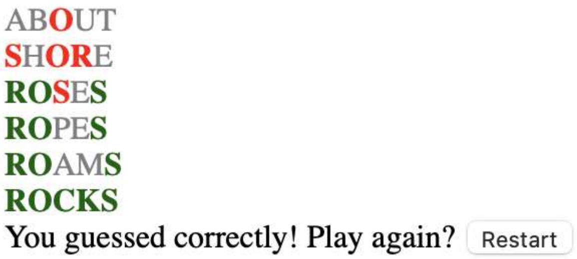
For this assignment, you are going to create a simplified version of the popular Wordle game running in a Wt server that you will connect with via a web browser. In essence, players get 6 chances to guess a 5 letter word. Every time a word is guessed, the player is given hints towards the target word by highlighting correct letters in the correct position using one colour and correct letters in incorrect positions with a different colour. (And completely incorrect letters are left unhighlighted.) This gives the player just enough information so that they can make better and better guesses each time until they've either correctly guessed the word or they have run out of guesses.

In creating your web application for this game, you will use appropriate web widgets to capture the essence of the game. How exactly you want to do this is up to you. As long as you show the player their previous guesses with letters highlighted accordingly, and give them a chance to enter new words, that's the main thing. For example, a screen shot of my implementation of things (and me about to win, by the way) is shown below:

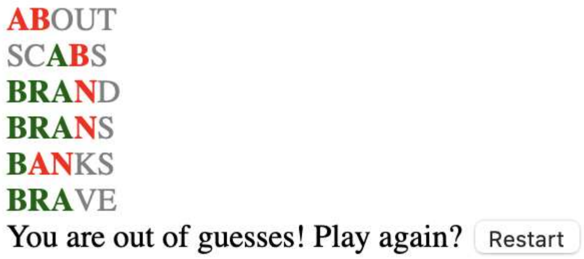

ABOUT
SHORE
ROSES
ROPES
ROAMS

Enter guess, 5 letters maximum: rocks

Here you can see my previous guesses, with green indicating correct letters in the correct places, red letters indicating correct letters in incorrect places, and grey letters indicating incorrect letters. A simple prompt/form is used to input and submit new guesses to the system. On submitting this last word, I would receive congratulations and be asked to play again, as shown below:


ABOUT
SHORE
ROSES
ROPES
ROAMS
ROCKS
You guessed correctly! Play again?

Likewise, if I were to run out of guesses from poor play, an appropriate message should be displayed, also prompting me to try again. This is shown in the example below. (Guess I should have used my BRAIN better there?)


ABOUT
SCABS
BRAND
BRANS
BANKS
BRAVE
You are out of guesses! Play again?

Particular requirements to follow when creating your Wordle application include the following:

- You are to follow the general Wordle rules as noted above. The player gets 6 attempts to guess a 5 letter word, with correct letters in the correct positions and correct letters in incorrect positions highlighted accordingly.

- Whether the player succeeds at guessing or runs out guesses, an appropriate message is given to the player and they are given the chance to try again at guessing a different word.
- You must use C++ and Wt to build your application as a web app so that it is playable from within a web browser. You can use the sample Hello World application attached to this assignment as a starting point.
- The aesthetics of your application is up to you. You can use whatever Wt widgets you would like, and can choose the layout, colours, styling and so on. Feel free to be creative here!
- You will have to provide your application a list of valid 5 letter words for the player to guess. Each time the player begins a game, a new target guess word should be randomly selected from this list. This list must not be hard coded within your application; it is likely easiest to supply this as a separate file in with the resources of your application. The format of this is up to you; a simple text file with one valid word per line is just fine, for example. Your list must contain at least 10 words, but could ultimately contain as many as you like. (You want to strip all valid 5 letter words out of a dictionary for your list? Go for it!)
- Your application should provide some sort of basic input filtering on words entered by the user. It should only accept letters as input and accept words of no more than 5 letters. (If you can filter it to only accept 5 letter words and nothing less, that's fine too, but not required. You must have an upper limit of 5 letters though.)
- Your word input should be case insensitive, so words like "about" and "ABOUT" are treated the same, for instance. Words should match regardless of their capitalization. (Notice that above, I entered words in lower case but output them in upper case -- I did that to make the letters and their colours more pronounced.)
- You are not required to ensure that the user only enters valid 5 letter words as input, as many Wordle apps do. You are free to do this if you like though, although this would likely be easiest if your word list consisted of all valid 5 letter words from the dictionary. (You could then screen input against the list and if it's not there, you don't count it as a guess and have the user input a valid word. Again, you don't have to do this, but you can if you'd like!)

Implementation Notes

For this assignment, you should package your application logic into its own class in a similar fashion to the Hello World sample application. You are free to have additional support classes too, if you find that to be helpful. Each class you create should be captured in two files: one header and one code file. Your app will also require its own main code file separate from these classes. (So you will have at least three C++ files in this assignment: a header and a code file for your application class, and one for your main app. You may have other files for other classes, as well as other support functions and such.)

Your code should also be written in adherence to the Coding Guidelines available here. Deviations may result in deductions from your assignment grade up to 10% of its overall value.

In addition to the above files, you must provide a Makefile or other appropriate build script to assist in the building of your classes and apps. The Makefile used for the Hello World sample application could be used as a starting point for things.

Additional resources for assignment

- wt4-hello.zip (1 MB; Sep 7, 2023, 1:53 pm)

Submission

Assignment Text

This assignment allows submissions using both the text box below and attached documents. Type your submission in the box below and/or use the Browse button or the "select files" button to include other documents. **Save frequently while working**.

Source

Styles ▾ | Format ▾ | Font ▾ | Size ▾ | ▾ ▾

Words: 0, Characters (with HTML): 0/1000000

Attachments

No attachments yet

Select a file from computer 选择文件 未选择任何文件 or select files from 'Home' or site

Proceed Preview Save Draft Cancel ❗ Don't forget to save or proceed!