CSE 325 - Project 1

Project Overview

This project will help you become familiar with the computing environment that will be used in CSE 325. You will complete the exercises below and submit your work for grading via the D2L system. Make sure to read the entire document before starting to work on the assignment.

Assignment Deliverable

The deliverable for this assignment is the following file:

proj01.cpp – your program code

This means you should only submit a file named "**proj01.cpp**" to D2L.

IMPORTANT (READ CAREFULLY BEFORE YOU LOSE POINTS): Your submission file for this and any future projects must be exactly as specified above, or it cannot be graded. To be more specific:

- In this project, your program code should be in a single '.cpp' file named "**proj01.cpp**". DO NOT include a Makefile or structure your code in multiple files.
- DO NOT submit a compressed file (e.g., '.zip', '.rar', '.7z', etc.). Directly submit the file(s) specified above. D2L accepts multiple submissions if needed.
- DO NOT submit a compiled executable!
- For example, a submitted file such as "proj01 (02).cpp" is not acceptable

Project Specifications

- The purpose of the program is to receive a memory address in hexadecimal as input through the command line/terminal and output information about the memory system in a specific format.
- The program will process the command-line arguments to receive a single input, which is a memory address in hexadecimal. For example, here's a sample call to the program:

```
./proj01 FA01BD06
```

Note that 'proj01' is the name of the executable after compilation, and 'FA01BD06' is a single input passed to the program through the command-line arguments. The input can include capital or non-capital letters. The following's a tutorial for processing command-line arguments in C++:

https://cplusplus.com/articles/DEN36Up4/

- The program should count the number of hexadecimal digits in the provided input and output the following information:

  1. Address size in bits (Each hexadecimal digit is 4 bits).
  2. Total memory size (2 to the power of the address size in bits).

  For example, for the address in the example above, 'FA01BD06', the output would be:

  ```
  Address Size: 32 bits
  Memory Size: 4294967296 bytes
  ```

  Make sure your program's output is identical to the example above for a valid input. You are free to handle invalid cases with any message you think best describes the issue. There are some test cases that are provided with the specification PDF.

- An important programming skill is to think about edge cases. In future assignments, edge cases will not be explicitly provided. For this assignment, however, to give you a sense of what you should consider when handling edge cases, the edge cases are as follows:

  ```
  ./proj01 hello   (wrong address)
  ./proj01         (no address)
  ```

  Your code should handle such cases gracefully. This means your code should acknowledge the error, print an appropriate message, and either exit the application or continue execution. Whether to exit or not is a design decision you need to make, but outputting a proper error message is a must. In fact, any detail not specified in the assignment specifications is up to you to decide.

- The maximum allowed input address size should be 32 bits.

- Once you are done with your code, follow these steps (Refer to the D2L supporting materials for guidance.):

  1. If you coded on your local computer, transfer your code to scully using a system program such as 'scp' or a tool like FileZilla.
  2. Log onto the scully servers using your EGR account.
  3. Navigate to the directory that contains your "proj01.cpp" file.
  4. Compile your code by executing the following command:

     ```
     g++ -Wall proj01.cpp -o proj01
     ```

  5. If you notice any errors or warnings, resolve them and compile your code again.
  6. Test your code by running the executable using the following command:

```
./proj01 ADDRESS
```

where you should replace 'ADDRESS' with your test input addresses.

Additional Notes

- When you don't know something, try searching for it. Don't just rely on AI tools. A strong programmer should be skilled at searching!
- Do not give away any part of the solution on Ed discussion.
- You can use VSCode to connect to compute servers like CSE 320. Please refer to the Supporting Document on D2L contents.
- You can find instructions about commenting your code here:

  https://edstem.org/us/courses/91271/discussion/7512441
- It's up to you which IDE to use when coding. **However, you must ensure that your code supports command-line inputs and works in the scully environment. C++ compilers are compatible with DIFFERENT header files for Windows, macOS, or Linux distributions.**
- You are only allowed to use standard C++ libraries.