**Introduction to Systems Programming (System I)**

**Lab Exercise #2**

|  |
| --- |
| **You should save/rename this document using the naming convention MUid.docx (example: guerinf.docx). Please note that in most examples, the database.csi server must be**  **used instead of ceclnx01.cec.**  **Objective**: The objective of this exercise is to learn the basics of developing and running C++ programs on a Linux machine.  Fill in answers to all of the questions. For some of the questions you can simply copy-paste appropriate text from the shell/output window into this document. You may discuss the questions with your instructor. |

|  |  |
| --- | --- |
| **Name:** | **Noah Dunn** |

# Developing C++ programs using NetBeans

|  |  |
| --- | --- |
|  | Wait for your instructor to cover the basics of using NetBeans for C++ programming prior to proceeding with rest of this exercise. |

In this course we will be primarily using NetBeans IDE from Oracle for developing programs on Linux (and eventually on the Red Hawk cluster). Note that you will need NetBeans version 8.1 and it is available for free (<https://netbeans.org/downloads/>). Use the procedure introduced by your instructor to develop the C++ program described below. You may review the process of developing C++ programs by watching the videos off Cascade.

# Finding documentation for C++ API

The C++ language API is reasonably compact and yet fast and versatile. In this course we will be using only the core language constructs. Additional libraries maybe included as needed. One of the most widely used libraries for C++ is BOOST and you may find BOOST useful for some of your programs (as directed by your instructor).

The best site to find documentation for C++ API with examples and ability to run code online is <http://en.cppreference.com/>. When searching for C++ API documentation it is best to prefix your methods/classes with the std:: (pronounced as “stood”) namespace. For example, avoid searching for just “string” and instead search for “std::string”. There are other documentation sites like [http://www.cplusplus.com](http://www.cplusplus.com/), that you may use.

**Documentation review exercise**

Using the online C++ documentation available at <http://en.cppreference.com/>, study the operation of the std::string::find() method(s) and respond to questions in the space below:

|  |  |
| --- | --- |
| What is the data type of the return value of std::string::find()? | str or String |
|  |  |
| What is the value returned by std::string::find() method when a search string is not found? | not found |
|  |  |
| In the example C++ program supplied for the std::string::find() method change the value of string s to: std::string const s = "This is Miami";  Run the program (in the browser) and paste a screenshot of your browser window showing the modified line of code in the example and the resulting output in the space below: | |
|  | |
|  | |

# Exercise: Simple string operations in C++

|  |  |
| --- | --- |
| C:\Users\Victoria\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\MARLET97\MP900314351[1].jpg | Ensure that the C++ source for this exercise is named with the convention *MUid*\_ex2\_1.cpp (example: jordanm\_ex2\_1.cpp), where *MUid* is your Miami University unique ID. |

**Background**: Similar to almost all programming languages, C++ provides a standard set of string functions that can be used to perform standard string operations.

**Exercise**: This exercise expects you to develop a simple program that performs the following string:

1. Read a line of input (using std::getline method) from the user after prompting for input (see sample output below for message details).
2. The program must display the following information about the line entered by the user (see sample output for message details):
   1. Number of characters in the input.
   2. The first and last character in the string.
   3. The first and last word in the string. You may assume that there is only one blank space between words (user will input at least 1 word) and there are no leading or trailing blanks. (use substr, find, and rfind methods defined on std::string)

**Style checking C++ programs**

In this course, we will be adhering to the CSE programming style conventions (<http://miamioh.edu/cec/academics/departments/cse/academics/programming-style/index.html>). The style conventions will be checked and enforced using the cpplint tool (a slightly modified version of Google’s lint). In homework (but not labs), points will be deducted if cpplint reports style violations on your source code.

**Sample Output**:

User inputs in two different runs of the program are shown in red color in the sample outputs below:

|  |  |  |
| --- | --- | --- |
| [raodm@mualhpcp01 exercise2]$ ./raodm\_ex2\_1  Enter a line: **i will go to spring ice**  Length of line : 23  First character: 'i'  Last character : 'e'  First word : 'i'  Last word : 'ice' |  | [raodm@mualhpcp01 exercise2]$ ./raodm\_ex2\_1  Enter a line: **checking 1 two 3**  Length of line : 16  First character: 'c'  Last character : '3'  First word : 'checking'  Last word : '3' |

**Actual Output:** Once you have successfully developed and tested your program, copy-paste the output from your program into the space provided below for the input string “oneWord”:

|  |
| --- |
| . |

# Running programs in a terminal window

Initially, we will run the programs in NetBeans. However, you should also get comfortable with running programs from the terminal window. Consequently, it is important to understand the interplay between NetBeans and the way it stores programs. Specifically, run your program directly in the terminal (at the shell prompt) and place a screenshot of the terminal window (showing outputs from your program) below using the following skills from Exercise #1:

1. In NetBeans, determine the directory where your program is stored on the server.
2. Next, SSH to the Linux server database01.cec.miamiOH.edu from a Terminal
3. cd to the project directory (that you noted in step #1)
4. Run the ls command to observe the files and structure
5. Locate your executable and run it in the terminal (example: **./exercise1** -- don’t forget the leading period!)

|  |
| --- |
|  |

**Copy source from remote machine for submission**

Once you have done all the development and testing, you will need to copy the file from the remote machine to the local machine to upload to Canvas. Obviously, the copy is needed because the browser is running on your local machine and it cannot access files on the remote server. Use scp command introduced in Exercise #1 to copy the files. You will need to suitably modify the command below to reflect path and file name:

|  |
| --- |
| $ scp [MUid@database.csi.miamiOH.edu:NetBeans/exercise1/exercise1.cpp](about:blank) **.** |

Notice this also can be done with *sftp*, and the **mget/get** command.

**Important note: Homework**

Basically , you have practice now in all you need to do to submit a homework. But take

in account that:

1. You’ll put your required files in a specific directory in **database.csi ,** as indicate in

the *guidelines.pdf* document.

2. You must make sure that your program compiles and run correctly in the new environment.

You’ll find more practice on how to do this on the supplementary exercises.

# Submit to Canvas

Once you successfully completed the aforementioned exercises, save this MS-Word document as a PDF file. Next upload the following:

* 1. This MS-Word document (duly filled-in) saved as a PDF document.
  2. The C++ source file of the program you developed.

Ensure you actually **submit** the files after uploading them to Canvas.

Please note: For **homeworks,** there is no need to upload in Canvas. But youu’ll need to put your programs in a specified file. Make sure you carefully read the document *guidelines.pdf* for rules.