**Heterogeneous and Cloud Computing**

**Exercise #2**

Max Points: 10

**Note: If you are using your personal machine then prior to commencing work on this exercise, you may need to install XMing, Putty, and WinScp as illustrated in LinuxEnvironment.pdf (and shown in the videos in the Handouts folder).**

|  |
| --- |
| **You should save and rename this document using the naming convention MUid.docx (example: raodm.docx).**  **Objective**: The objective of this exercise is to gain some familiarity with developing C/C++ programs on a Linux machine. This exercise will also serve to refresh the basic C++ programming that you have learnt from earlier courses.  **Submission**: Once you have completed this exercise, upload this MS-Word document (duly filled with the necessary information) and the C++ program that you developed with the following naming convention:   * ***MUid*\_ex2.cpp**: A C++ program to perform simple string operations in C++   Fill in answers to all of the questions as directed. For some of the questions that require outputs to be indicated, you can simply copy-paste appropriate text from the shell/PuTTY window into this document. You may discuss the questions with your instructor. |

|  |  |
| --- | --- |
| **Name:** |  |

# Preliminaries

You may use any editor or IDE that you are familiar with for programming. However, it is highly recommended you get comfortable using emacs. It is a powerful and rich environment and a good compromise between various tools, particularly when running over a network connection.

# Configuring Niihka Forums

In this course we will be using Niihka forums for soliciting help and conducting discussions. In order to ensure you get timely copies of postings and discussions, it is highly encouraged you receive email notifications upon postings. Consequently, enable email notifications in the following manner:

1. Log into Niihka and select the Niihka site corresponding to this course.
2. Click on the Forums tab (typically to the right hand side) to get to the forums settings as shown in the screenshot below:

|  |
| --- |
|  |

1. Next click on the Watch button (highlighted in the screenshot above) and select the radio button for “Send me an email whenever a new message is posted”
2. Next click on the Save button.

# Ccopy my emacs configuration

To provide consistent formatting and use of emacs, copy my emacs configuration into your home directory using the following commands:

|  |
| --- |
| $ cd  $ rm –f .emacs  $ wget -q http://pc2lab.cec.miamiOH.edu/documents/.emacs |

# Compiling C++ Programs:

We will be developing straightforward C++ programs that are relatively straightforward to compile and run. However, the compilation and style checking operations can be streamlined to make development easier. Accordingly, a simple generic Makefile is supplied off Niihka in the Resources🡪Handouts folder. Note that, “hand writing” Makefile is archaic and should be avoided in any “real” project. The supplied Makefile provides the following features:

* Use with emacs’s flymake mode (to dynamically highlight compile errors)
* Compile one C++ source file (you can modify it to compile many)
* Download (if needed) and run the style checker on the given source file

**Usage**:

The supplied Makefile can be used in the following manner in emacs:

* From the emacs menu choose Tools🡪Compile (or use keystore Ctrl-C C)
* In the mini buffer (at the bottom of emacs window) modify the make –k default command to:
  + Working with a one c++ source file: make SRC=hello.cpp. Change hello.cpp to the C++ source file you are working with.
  + Working with all source file(s) in current directory: make many EXE=hello. Change hello to the executable you would like to generate.

# Part #1: 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.cpp (example: raodm\_ex2.cpp), where *MUid* is your Miami University unique ID. |

**Background**: Similar to almost all object-oriented programming languages, C++ provides a std::string class and standard set of methods to operate on strings. In C++ a string can also be operated as an array of characters and is mutable.

**Exercise**: This part of the exercise expects you to develop a simple C++ program by completing a countVowels method in the skeleton code.

1. Download the supplied exercise2.cpp from Niihka.
2. Review the program briefly.
3. Without modifying the signature of the countVowels, implement the method to a return count of vowels in the given string. **Note**: If you are exceeding the 10 lines of code in this method, then you are going about it the wrong way.
4. Once you have implemented the method, run the program (as shown in the sample outputs) below to ensure it is operating correctly.

**Sample Output**:

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

|  |
| --- |
| $ ./raodm\_ex2 --str abcdefghijklmnopqrstuvwxyz  Vowel count: 5 |
|  |
| $ ./raodm\_ex2 --str abcdEfGhIjklMNopqrsTUVwxyz  Vowel count: 5 |
|  |
| $ ./raodm\_ex2 --file /usr/share/emacs/24.3/etc/JOKES  Vowels count: 3391 |
|  |
| $ ./raodm\_ex2 --file /boot/initrd.img-3.13.0-24-generic  Vowels count: 752921 |

**Verification**:

Once you have completed this part of the exercise, verify the solution with your instructor and demonstrate its operation to your instructor before proceeding to the next part of this exercise.

# Part #2: Understanding impact of parameter passing

|  |  |
| --- | --- |
| C:\Users\Victoria\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\MARLET97\MP900314351[1].jpg | Ensure that you have successfully completed the previous part of the exercise prior to proceeding with this part of the exercise. |

**Background**: Unlike many programming languages C++ permits parameters to be passed using three different conventions:

* Pass by value (arguments are copied prior to be being passed to the method)
* Pass by reference (copies are not made and only valid objects may be passed)
* Pass by pointer (copies are not made, and NULL pointers can be passed)

**Exercise**: This exercise expects you to observe the effects of passing parameters by value versus parameters by reference in the following manner:

1. The supplied countVowels method passes arguments by value. Measure the runtime characteristics of this version of the program using the following command and record the necessary observations in the table further below:

|  |
| --- |
| $ /usr/bin/time -v ./raodm\_ex2 --file /boot/initrd.img-3.13.0-24-generic |

1. Next modify, the countVowels method to accept arguments via reference (by making 1 character change shown below):

|  |
| --- |
| int countVowels(std::string& str) |

1. Compile, run, and record its runtime characteristics in the observations table below.

**Observation:**

Record the runtime characteristics from 3 independent runs of the program that uses pass-by-value approach for parameter passing:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter Passing Mode** | **User Time**  **(sec)** | **Elapsed Time**  **(sec)** | **%CPU** | **Max resident size**  **(KB)** |
| Pass by value (#1) |  |  |  |  |
| Pass by value (#2) |  |  |  |  |
| Pass by value (#3) |  |  |  |  |
| **Averages (of 3 runs)** |  |  |  |  |

Record the runtime characteristics from 3 independent runs of the program that uses pass-by-reference approach for parameter passing:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter Passing Mode** | **User Time**  **(sec)** | **Elapsed Time**  **(sec)** | **%CPU** | **Max resident size**  **(KB)** |
| Pass by reference (#1) |  |  |  |  |
| Pass by reference (#2) |  |  |  |  |
| Pass by reference (#3) |  |  |  |  |
| **Averages (of 3 runs)** |  |  |  |  |

**Inference**

Draw inferences about the different parameter-passing methods by comparing:

1. Percentage difference in average elapsed time
2. Max resident size (aka peak memory used)

|  |
| --- |
| Record your inference here. Use as much space as you need |

# Part #3: Submit files to Niihka

Upload the following files to Nihhka:

1. Upload this MS-Word document (duly filled with the necessary information) saved as PDF using the naming convention **MUid.pdf**.

* ***MUid*\_ex2.cpp**: The C++ program developed in this exercise.