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

**Lab Exercise #3**

|  |
| --- |
| **You should save/rename this document using the naming convention MUid.docx (example: jaggerm.docx).**  **Objective**: The objective of this exercise is to:   1. Review reading data from standard-input and redirecting input/output 2. Complete a program that generates HTML formatted output 3. Review using the debugger in NetBeans   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:** |  |

# Experimenting with input/output (I/O) redirection

*Estimated time: 30 minutes*

**Background: Redirecting input** – Most program in Linux (and Mac/Windows as well) are designed to read inputs from standard input (std::cin) until the user enters logical end-of-file (EOF). In Linux, pressing CONTROL & D keys together on a separate line indicates logical EOF. Developing programs in this manner eases providing inputs to the program by simply redirecting data from a text file as shown below:

|  |
| --- |
| $ ./exercise3 < inputs.txt |

In the above example, rather than manually typing inputs via a keyboard each time the program is run, the input is redirected from a text file called input.txt that contains the inputs to be supplied to the program. This is a convenience feature that streamlines testing programs.

In this context, it is important to recollect the process of redirecting output of a program (*i.e.*, data written to std::cout) to a file as reviewed in Exercise #1 and covered below:

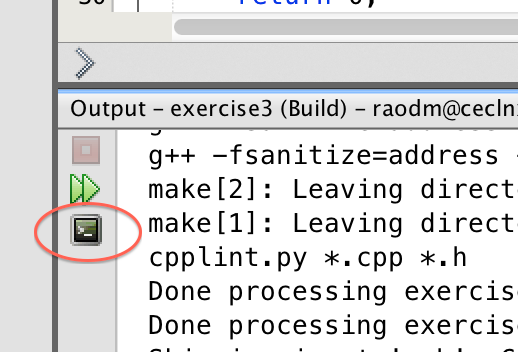
|  |
| --- |
| $ ./exercise3 > output.txt |

Yes, redirecting inputs and outputs can be combined as shown below:

|  |
| --- |
| $ ./exercise3 < inputs.txt > output.txt |

**Experimental procedure:**

Experiment with simulating EOF and I/O redirection using the following procedure:

1. Setup a remote project on our Linux server database.csi.miamioh.edu in NetBeans. For convenience name the project exercise3.
2. Next, copy-paste the supplied C++ source (replacing everything in the template file generated by NetBeans)
3. Compile the code in NetBeans (it should compile without any errors).
4. Next, open a **Terminal in NetBeans** by clicking the terminal icon in the output window as shown in the adjacent screenshot. Note that this terminal (in NetBeans) is the same as the Mac terminal or using PuTTY in Windows. It uses SSH to connect to database.csi.
5. In the terminal *run* (see step 6 if you have difficulties) the program, type in 3 input lines, namely “Love”, “and”, “Honor” (one word per line as shown in red letters – in your case you will not see color) and press control d (at the beginning of the line) to simulate EOF to obtain example output shown below:

|  |
| --- |
| $ ./exercise3  <!DOCTYPE html>  <html>  <head>  <link type='text/css' rel='stylesheet' href='http://ceclnx01.cec.miamiOH.edu/~raodm/ex3/ex3.css'/>  </head>  <body>  **Love**  Love  **and**  and  **Honor**  Honor  </body>  </html> |

1. You can also run command line in the terminal that you just open. But where is the executable? From your **project home** ~/NetBeansProjects/exercise3 the executable is

in ./dist/Debug/GNU-Linux. Make sure that you build using Debug, in the toolbar. If you are using C++ Application you may have “Release” instead of “Debug”.

1. Note that HTML is just a simple text format with open/close tags (of the form: <div> </div>). This course does not require you to know HTML per say, but it is something you can easily learn in a few hours using the tutorial at: <http://www.w3schools.com/html/>

**Experiment with input redirection**

1. In order to supply inputs from a text file, you need to create it. Create a text file in NetBeans using the following menu options: Right-click on leftmost Icon to create a file  New  Other  Other (in Categories)  Empty File  Next  lah.txt  Finish.

(Close Tab → Save).

1. The above menu options will create an empty text file. In it type the 3 words Love, and, Honor on 3 separate lines and save it. By default it will create it in your project home directory.
2. Copy the executable to your project home (so you do not have to specify a long path as in 6). In your **NetBeans terminal** you can now try redirecting inputs using the following command:

|  |
| --- |
| $ ./exercise3 < lah.txt |

If you get errors use the ls command to ensure you see exercise3 (typically in green color) and the lah.txt file you just created. The output from your program should be similar to the one shown earlier (except you will not see your input data)

**Experiment with out redirection**

1. Now you may also redirect to a file using > as in the introduction. Run again redirecting the output to file ex3*.txt.* View the file using your favorite text editor or the command *cat.*

|  |
| --- |
| $ ./exercise3 < lah.txt > ex3.txt |

1. Now login to **ceclnx01.cec** (remember this host runs a web server).In your home directory, create a directory

|  |
| --- |
| $ mkdir -p ~/public\_html |

If everything goes well, consistent with Linux commands (which do not print any information when things go well) there will be no outputs from the above commands.

1. Remote copy your ex3.txt into the directory above, on **ceclnx01.cec**. Make sure you

named it correctly and the file is there. You may be able to do this from NetBeans as well. Rename it **ex3.html.**

What you have created is a web page. As the server **database** does not have a Web Server, you need it to copy to ceclnx01, specifically to *~/public\_html/,* which you created in step 12. You may also view the html file via a web-browser via the following URL (you will need to replace MUid with your login id):

|  |
| --- |
| http://ceclnx01.cec.miamioh.edu/~MUid/ex3.html |

|  |  |
| --- | --- |
|  | In general, the input and output file name(s) or paths can be any valid paths. However, if you would like the file to be accessible via a URL, then you need to place those files in ~/public\_html directory. |

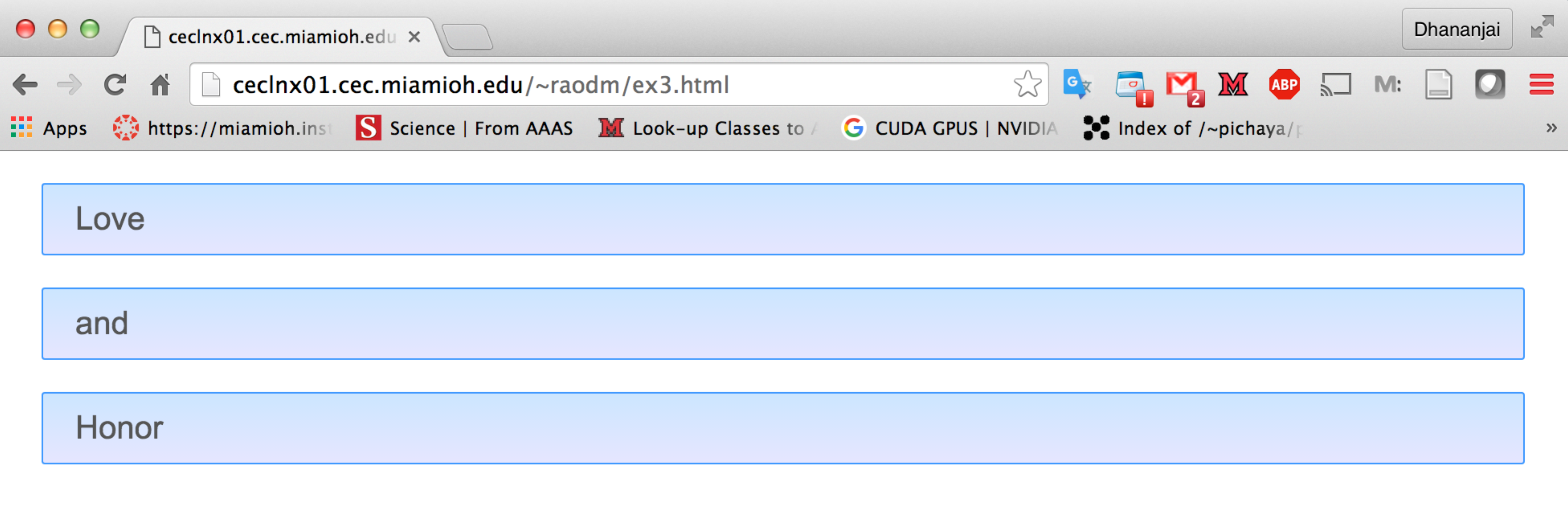
# HTML Formatting

*Estimated time: 15 minutes*

**Background**: Note that HTML is just a simple text format with open/close tags (of the form: <div> </div>). This course does not require you to know HTML per say, but it is something you can easily learn in a few hours using the tutorial at: <http://www.w3schools.com/html/>. We will may use this knowledge in the future.

**Exercise**: In this part of the exercise you will be modifying the convert method in the following manner:

1. Currently, the convert method simply returns the line pass in to it. Instead modify it to return the line wrapped in “<div>” and “</div>” tags. That is, if line is “Love”, the convert method should return the string “<div class=’line’>Love</div>”
2. Compile and run the program via the NetBeans terminal by redirecting standard input and output using the command from previous task. The same command is repeated below for your convenient reference. Repeat step 14, to get a new ext3.html on the *web server*.
3. View the file via your web-browser. If you have the implemented the convert method correctly, your output should appear similar to the output shown in the screenshot below:



1. Copy-paste your generated HTML file in the space below ..,:

|  |
| --- |
|  |

# Submit to Canvas

Once you successfully completed the aforementioned exercises upload the following files to Canvas.

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

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

# Optional: Using the debugger

*Estimated time: 15 minutes (after submitting to Canvas)*

Try getting hands on the debugger. It may come in handy in future laboratory exercises and homework. Try to gain familiarity with using the debugger, make sure you are in Debug mode. Here are operations you should try:

1. Try setting and removing breakpoints
2. Step line-by-line in a method and observe changes in a variable
3. Try navigating the stack frames in the debugger