

SCSE-443/543: High Performance Computing

Exercise #2

Max Points: 20

Note: If you are using your personal machine then prior to commencing work on this exercise, you need to install a SSH client (PuTTY) and NetBeans 8.2 with necessary plugins (see Files→Handouts folder on Canvas)

You should save/rename this document using the naming convention **MUId.docx** (example: raodm.docx).

Objective: The objective of this exercise is to:

- Create simple C++ class and implement key methods
- Gain familiarity with using the debugger to troubleshoot C++ programs

Fill in answers to all of the questions. For almost all the questions you can simply copy-paste appropriate text from the Terminal window into this document. You may discuss the questions with your neighbor or instructor. However, the key part of the exercise is to try the different commands and explore them via trial-and-error process.

Name: Sam Horn

Part #1: Creating a simple C++ class

Estimated time: 20 minutes

Background: NetBeans is the recommended IDE as it works reasonably well for remote projects (unfortunately, there are no good IDE's for C++ because experts hate IDEs). Note that when using NetBeans to develop remotely on **redhawk.hpc.miamioh.edu**, every operation runs on redhawk.



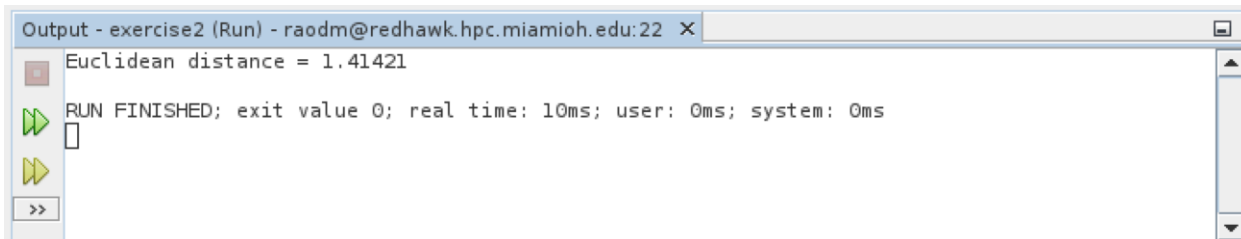
Video demonstration illustrating the use of NetBeans for remote C++ development is available on Canvas under: Pages → Video Demonstrations. You can review the video and use it as a guide for this exercise.

Exercise:

1. Using NetBeans, create a remote C++ project on **redhawk.hpc.miamioh.edu** using the "Miami University C++ project" as the project template. Ensure you name your project and your main source file `exercise2`.
2. Copy-paste the supplied starter code in `exercise2.cpp` from Canvas overwriting the skeleton code in NetBeans.

- Next create a C++ class using NetBeans via the following menu options: Right-click on Project→New→C++ class. **Ensure you name your class Point**. NetBeans will create 2 files for you, namely: `Point.h` (header file) and `Point.cpp` (source file).
- Next modify and implement the `Point` class to have the standard API methods shown in the adjacent UML diagram. The instance variables `x` and `y` contain the `x` and `y` coordinates of a point in a 2-D Euclidean plane. Note that the `dist` method returns the Euclidean distance between the 2 points, this and other. Euclidean distance is computing using the formula: $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$
- Compile the program and fix any style violations.
- Run the program with command-line arguments `1 1 2 2` to test it and verify you get the output shown below. Of course, video demonstration illustrating the use of NetBeans for remote C++ development is available on Canvas under: Pages → Video Demonstrations.

Point
- x: double - y: double
+ Point(double x = 0, double y = 0) + Point(const Point& orig) + ~Point() + dist(const Point& other): double



```
Output - exercise2 (Run) - raodm@redhawk.hpc.miamioh.edu:22 X
Euclidean distance = 1.41421
RUN FINISHED; exit value 0; real time: 10ms; user: 0ms; system: 0ms
```

Part #2: Using the debugger for troubleshooting

Estimated time: 10 minutes

Background: The debugger is one of **the** most important tool that every developer must master. Working with a debugger is a vital skill necessary for troubleshooting programs, particularly complex multithreaded programs where inserting print statements etc. may suppress bugs making it impossible to debug programs. Needless to add your instructor will ask you to show values of variables in a debugger when asked for help troubleshoot programs.

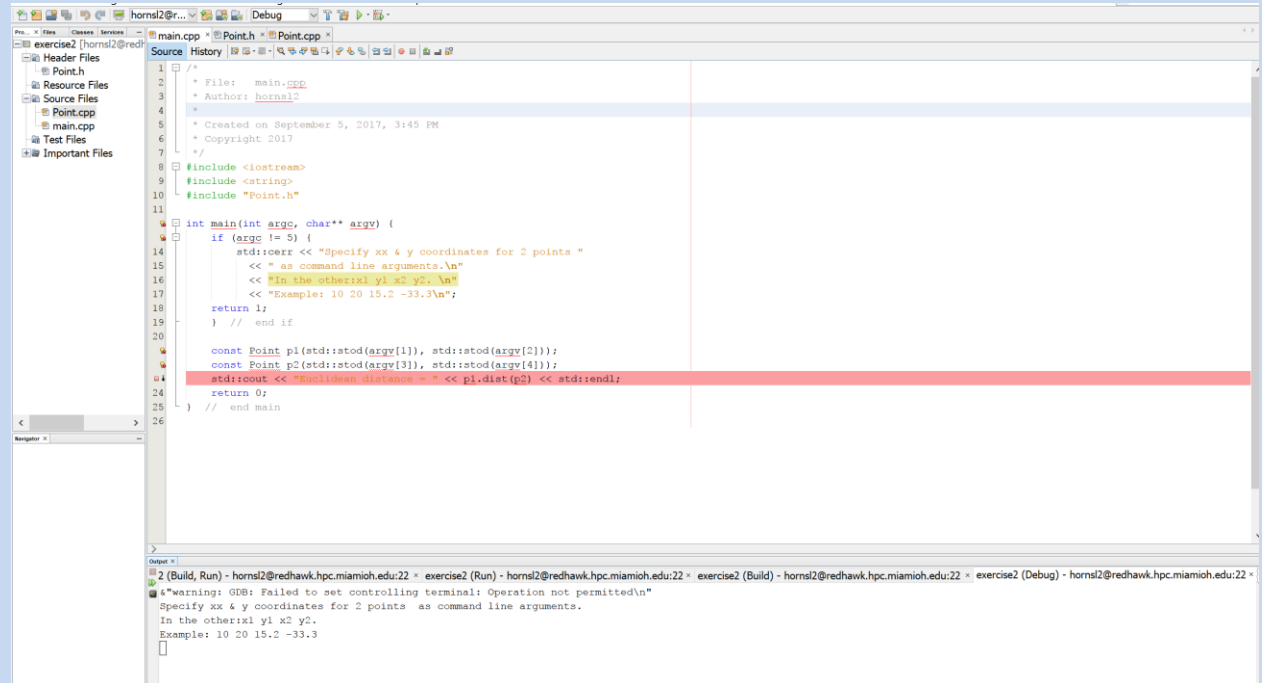


Working with a debugger is like using a fire extinguisher to put out a fire. You don't want to be reading the instructions on a fire extinguisher in the middle of a fire -- it could cost you your life. Debugger is similar. You want to be familiar with it so that you never have to think twice about using it for troubleshooting aka "fire fighting" in tech lingo.

Exercise:

Gain familiarity with compiling and running programs from the command-line via the following procedure:

- In NetBeans, set a breakpoint on the line in `exercise2.cpp` that prints the distance between 2 points.
- Run the program via the debugger.
- When the breakpoint is hit, make a screenshot showing the values of instance variables `x` & `y` for the objects `p1` and `p2` use in the `main` function and paste it below:



```
1  /*  
2  * File:   main.cpp  
3  * Author: horns12  
4  *  
5  * Created on September 5, 2017, 3:45 PM  
6  * Copyright 2017  
7  */  
8  #include <iostream>  
9  #include <string>  
10 #include "Point.h"  
11  
12 int main(int argc, char** argv) {  
13     if (argc != 5) {  
14         std::cerr << "Specify xx & y coordinates for 2 points "  
15             << "as command line arguments.\n"  
16             << "In the other: x1 y1 x2 y2. \n"  
17             << "Example: 10 20 15.2 -33.3\n";  
18         return 1;  
19     } // end if  
20  
21     const Point p1(std::stod(argv[1]), std::stod(argv[2]));  
22     const Point p2(std::stod(argv[3]), std::stod(argv[4]));  
23     std::cout << "Euclidean distance = " << p1.dist(p2) << std::endl;  
24     return 0;  
25 } // end main  
26
```

Output:
2 (Build, Run) - horns12@redhawk.hpc.miamioh.edu:22 * exercise2 (Run) - horns12@redhawk.hpc.miamioh.edu:22 * exercise2 (Build) - horns12@redhawk.hpc.miamioh.edu:22 * exercise2 (Debug) - horns12@redhawk.hpc.miamioh.edu:22 *
*warning: GDB: Failed to set controlling terminal: Operation not permitted\n
Specify xx & y coordinates for 2 points as command line arguments.
In the other: x1 y1 x2 y2.
Example: 10 20 15.2 -33.3
[]

Unfortunately my terminal was not configured correctly and was unable to get back to the computer lab in time to find a properly working Netbeans terminal due to work.

Submission

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

- This MS-Word document (duly filled-in) saved as a PDF document.
- The Point.h and Point.cpp files generated in this exercise (you need to scp it to your local machine for submission)

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