BTP305 Lab 9

C++ Binary Files and Threads

In this lab, you will develop and application you will read/write binary files and perform matrix multiplication using 2 threads.

# LEARNING OUTCOMESs

Upon successful completion of this lab, you will have demonstrated the ability to:

* Read/Writing of binary files using an fstream object
* Create and execute 2 threads using the thread class
* program components of quadratic complexity to solve non-linear problems

# SPECIFICATIONS

This workshop will consist of a single module spawning two threads as follows:

* Module w9 – Only using one file
  + –Open O strea/ binary writing
  + Create loop write the num 0 to 99 to the binary file using integer data type
* Thread T1
* Thread T2

## W9 Module

Write a w9.cpp file that contains the following functionality:

* Opens ***ofstream*** object using binary for a file called ***data.bin***
  + Create a loop and write the number 0 to 99 into ***data.bin*** using an ***int*** data tayp
  + *REFERENCE: http://www.cplusplus.com/reference/ostream/ostream/write/*
* Creates a thread **T1** that executes the ***RunT1()*** thread logic (see below)
* Creates a thread **T2** that executes the ***RunT2 (std::string filename)*** thread logic (see below)
* ***Join()*** threads ***T1*** and ***T2*** to synchronize them to the main process

Try and design your main function to run thread T1 before T2.

# Thread T1

Using the following code as a starting point, create a function ***RunT1()*** that will execute the logic for thread ***T1***. This function contains the following functionality:

* Loads the ***matrixA*** and ***matrixB*** 2D arrays with values from the following input files:
  + ***matrixa\_input.txt***
  + ***matrixb\_input.txt – need to put ‘B’***
* Displays the unique thread ID
* Performs a 2D matrix multiplication and stores the value in ***matrixC***
* Display the resulting ***matrixC*** to the screen

void RunT1()

{

int matrixA[5][5];

int matrixB[5][5];

int matrixC[5][5];

//TODO: Print out the threads unique ID here

//load the matrix data

std::ifstream infile("matrixa\_input.txt");

std::ifstream infile2("matrixb\_input.txt");

//TODO: Read the data from the open input files into matrixa and matrixb

infile.close();

infile2.close();

//TODO: Perform the matrix multiplication here

REFERENCE: http://www.cplusplus.com/forum/beginner/100491/

std::cout << "your matrix is" << std::endl;

//TODO: Print out the resulting matrixC 2D array to the screen

}

# Thread T2

Using the following code as a starting point, create a function ***RunT2(std::string inputfile)*** that will execute the logic for thread ***T1***. This function contains the following functionality:

* Displays the unique thread ID
* Creates an ***std::ifstream infile*** to load a binary file
* Reads the contents of ***infile*** to a ***char \*buffer***
* Displays the contents of ***char \*buffer*** to the screen

void RunT2(std::string inputfile)

{

char \*Buffer;

//TODO: Print out the threads unique ID

//TODO: create an ifstream infile to read the binary file inputfile

//get size of the file

infile.seekg(0, infile.end);

long size = infile.tellg();

infile.seekg(0);

std::cout << "Number of bytes in file is " << size << std::endl;

//Allocate buffer space

Buffer = new char[size];

//TODO: Read the data from the binary file

REFERENCE: http://www.cplusplus.com/reference/ostream/ostream/write/

//TODO: Display the contents of the binary file to the screen

HINT: You have a character buffer, but the values are of type int

infile.close();

}

(int)&buffer[x]

X+= size of int

Std::ifstream name(‘fielname”, std::ios::binary);

## Sample Output

**NOTE: Your thread ID’s will be different to this sample output.**

Thread ID = 772

your matrix is

69 49 54 42 38

169 134 159 132 108

269 219 264 222 178

369 304 369 312 248

469 389 474 402 318

Thread ID = 1180

Number of bytes in file is 400

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29

30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 5

57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82

3 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99

# SUBMISSION REQUIREMENTS

Once you have completed your lab create a single ZIP file that contains the following information and upload your ZIP file to Blackboard using the lab submission link.

* All your source code files (\*.h and \*.cpp)
* Execution instructions file (if there is anything special I need to know to successfully run your program write them down in a README file)
* Any input files required (test inputs, etc….)
* Any output files generated by your code
* A file summarizing to your instructor the concepts that you have learned in doing this particular workshop.  Add any other comments you wish to make in the comment box provided