/\*

Course\_Name: OOP 345

Section: A

Title: Workshop 9 Part 2

Module: Process\_data

File\_Name: Process\_data.cpp

Student\_Name: Zhaokai\_Guan

Student\_ID: 130988215

\*/

This is the last workshop for OOP345, to be honest, I feel the workshop is getting slightly easier with the semester going on. I had the opportunity to practice the multi-thread skills learned during the lecture. Also, for the very first time, I was able to practice the std::bind() and binary read.

was there any benefit from using binary files in this workshop?

What I really like about binary files is I do not need to read the file based on the format. I can just simply read or write base on the size given by the requirement. As a result, the logic of my code is getting really simple. No more cin.ignore() methods or cin.getline() to set the rules of the reading.

Here is the logic of my read process:

std::ifstream ifstr(filename, std::ios\_base::binary);

if (ifstr) {

ifstr.read((char\*)&total\_items, sizeof(total\_items));

}

data = new int[total\_items];

int index = 0;

while (ifstr.good()) {

ifstr.read((char\*)&(data[index]), sizeof(data[index]));

index++;

}

ifstr.close();

// Binary write here//

std::ofstream ofstr(target, std::ios::binary);

ofstr.write((const char\*)&total\_items, sizeof(total\_items));

for (int i = 0; i < total\_items; i++) {

ofstr.write((const char\*)&data[i], sizeof(data[i]));

}

One thing need to be really careful is we have to use the .read or .write with the arguments met the requirement of the parameter. No matter what it is, we have to cast it to a char\* while reading.

why would be important to bind a function to its arguments, and how was it useful in this workshop?

When I first learned std::bind() I honestly think this is totally useless. But now I finally see the advantage of this. Std::bind() will return a new function that forged with the supplied values. As a result, there will be less argument passed while calling this new function. This is especially useful while we have some parameter values won’t be changed. Instead of overloading functions, we could just get a temporary function returned by the std:bind() in order to simplify the function calling.

Here I bind the function with the place holders and two of the supplied values.

// God!!!!!!!!!!!!!!!!!!!!!!!//

// Don't try to be a smart butt, naming the placeholder 5 because it is the fifth parameter of the function//

// The compiler will actually look for the missing 3 and 4//

// JUST NAME YOUR PLACE HOLDERS FOLLOW THE RULES OF MOTHER NATURE, 1 IS 1 2 IS 2//

std::vector<std::thread> var\_threads;

auto vari = std::bind(computeVarFactor, std::placeholders::\_1, std::placeholders::\_2,total\_items, average, std::placeholders::\_3);

for (size\_t i = 0; i < (size\_t)num\_threads; i++){

var\_threads.push\_back(std::thread(vari, &data[p\_indices[i]], (total\_items / num\_threads), std::ref(variances[i])));

}

iterator = 0;

for (size\_t i = 0; i < (size\_t)num\_threads; i++) {

var\_threads[i].join();

variance += variances[i];

}

It is worth to notice that placeholders naming is also important, we have to follow the sequence of the natural. Never overthink about naming ever.

The advantages of using multiple threads, and how did you accomplished multi-threading in this workshop.

The advantages of using multiple threads are very obvious. It makes the code run faster and more efficient. With one thread, it takes 215 ms to complete the same task compared to 135ms used 2 threads. I am very positive that this kind of skills are very welcomed by those places that are highly driven by the run time speed where I absolutely desire to work at.

As for how I completed this, I used prof’s way to create a vector of threads, then used a ranged based for loop to loop through all the thread element to allocate tasks on each thread.

Before the program exits, I used another ranged base for loop to join all the threads back.

ONE VERY IMPORTANT THING IS:

Using more threads do not mean your program will run faster than the other. Because in my case, using 2 threads run time speed is 10 ms faster than 4 threads for a few times. When we are designing a program, how many threads will be used should be definitely considered carefully and tested as well.