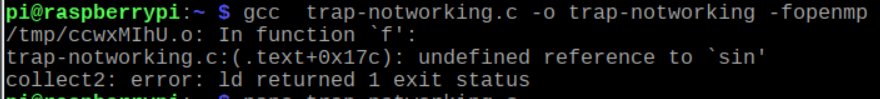
**Parallel Programming Basics**

**Part 1:**

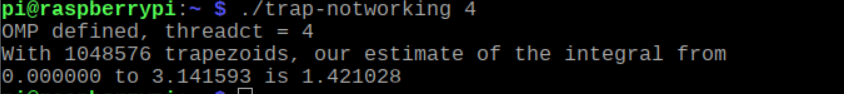
We created the file, trap-notworking.c, and wrote the code provided. When we compiled the program, we didn’t import the math library, so there were errors.



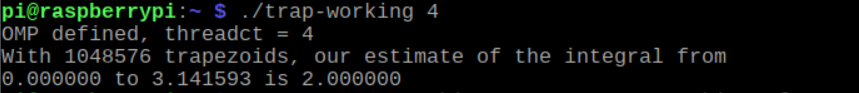
We searched online and realized we need to include -lm at the end of our compile command in order to use the sin function from the math library and trap-notworking compiled successfully.



We ran trap-notworking and it gave us an output of 1.421028. The correct output should have been 2.00.

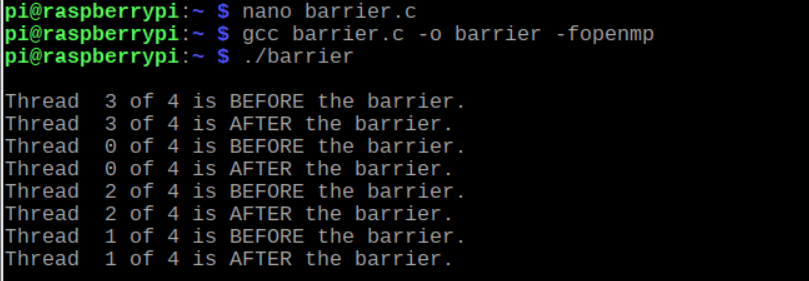


We followed the code provided and realized that integral is an accumulator variable, so we need to use “reduction: +integral”. This will make sure the individual values of integral after each thread is finished will be added together. The i is declared as private so that each thread uses its own I variable, and a, n, and h were declared as shared so that all the threads could use them globally. We created a file, trap-working.c, and wrote the code like the lines in trap-notworking.c and made the necessary changes. When we compiled and ran trap-working, we got the correct output, 2.000000.

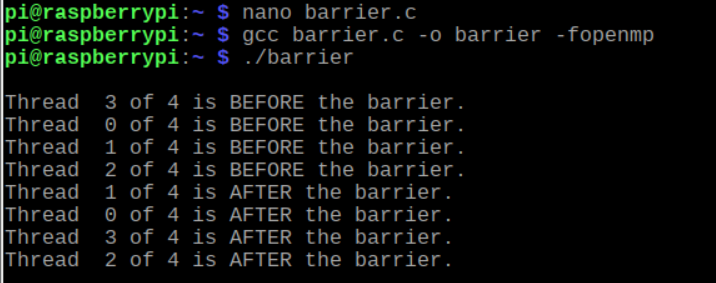


**Part 2:**

We created a file named barrier.c. Then, we created an executable program ‘barrier’ using gcc barrier.c -o barrier – fopenmp. We then proceed to run the program with ./barrier. We then proceeded to run the file and received the following output. We noticed that the same thread would output “before the barrier” and “after the barrier”, but there was no clear divide.

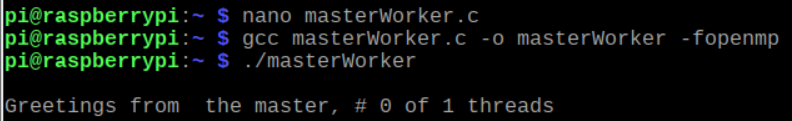


We uncommented “#pragma omp barrier” at line 31. We recompiled the program, ran it, and received the following output. “#pragma omp barrier” makes sure the program waits until all the threads finished outputting “Thread # of # is BEFORE the barrier.” before outputting “Thread # of # is AFTER the barrier.”



**Part 3:**

We created a file named masterWorker.c. Then, we created an executable program ‘masterWorker using gcc masterWorker.c -o masterWorker – fopenmp. We then proceed to run the program with ./ masterWorker. We then proceeded to run the file and received the following output.



We uncommented “#pragma omp parallel” at line 24. We recompiled the program, ran it, and received the following output. The master prints “Greetings from the master, #0 of 4 threads” and the rest of the worker threads executes the worker message.

