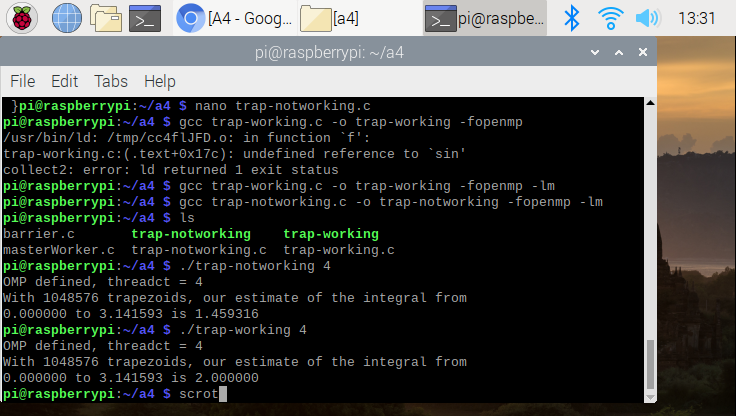
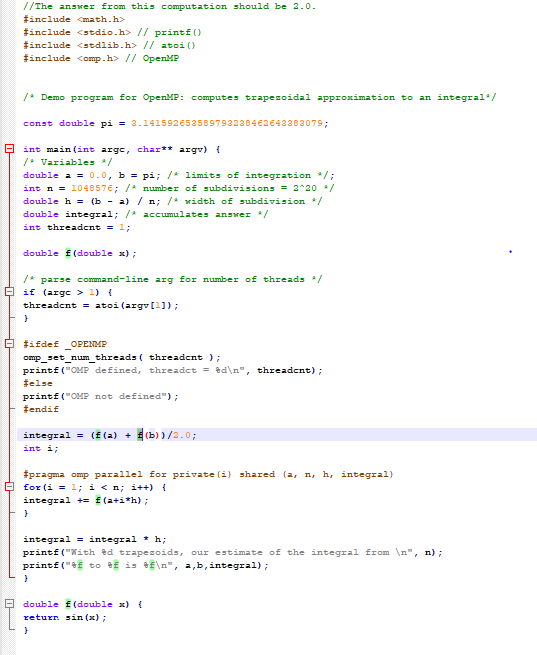
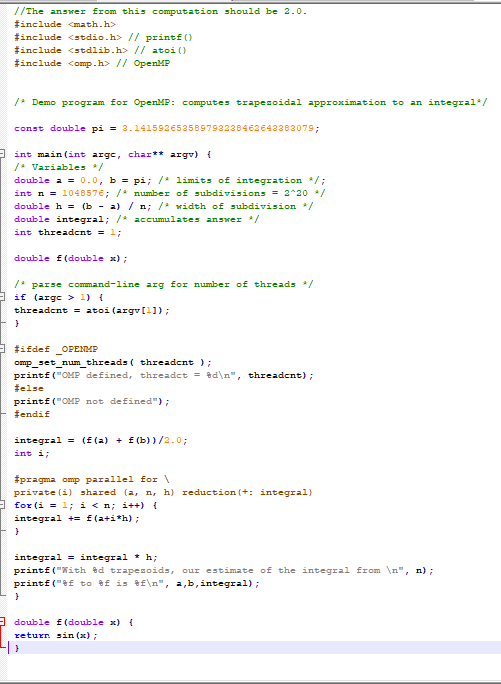
TrapWork & TrapNotWork ScreenShot



The photos are included both “TrapWork” and “TrapNotWork” result. This program use a 4 threads, since raspberry pi contains 4 core processor. Both programs are designed to compute the integral of sin of x from 0 to pi. Both programs are nearly equal program except line 37 and 38. The difference for those program is “private(i) shared (a, n, h) reduction(+: integral) “(in trapWork), this statement would allow the processor to perform the reduction parallel algorithm. This is the main reason why the program for Work and NotWork shows the different result. Overall, the correct answer for integral equation is shown in “Trap-Work” result(2.000). The answer for trap-notWork is incorrect.



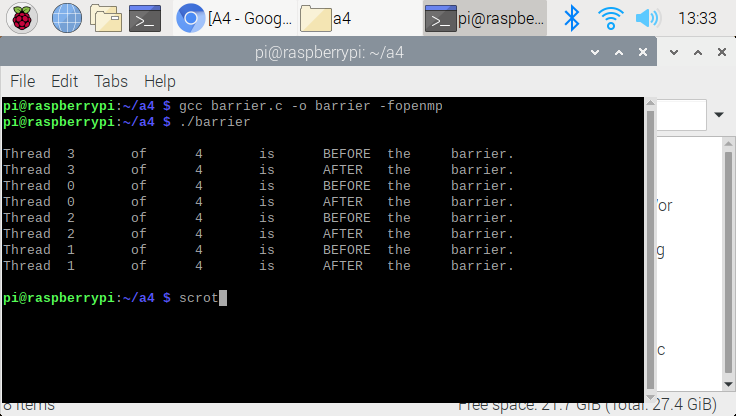
This is a code for “trap-notworking”. A you could see, the only difference from “trap-Working” is a #pragma omp.



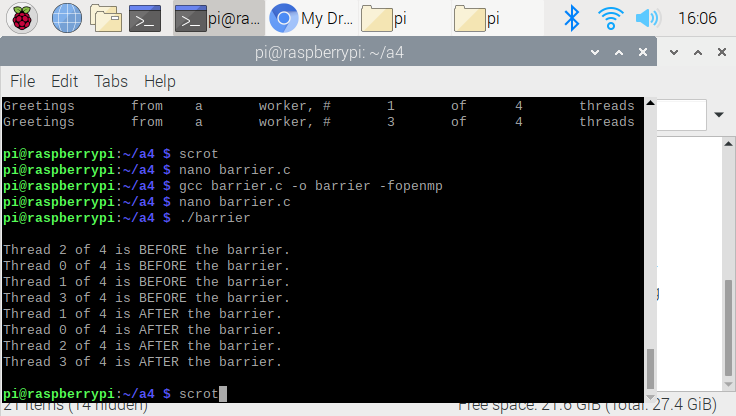
This is code for “trap-working”. The line “private(i) shared (a,n,h) reduction was added in this program. This makes the difference, and allow us to retrieve the correct result

Barrier Screenshot

Without pragma



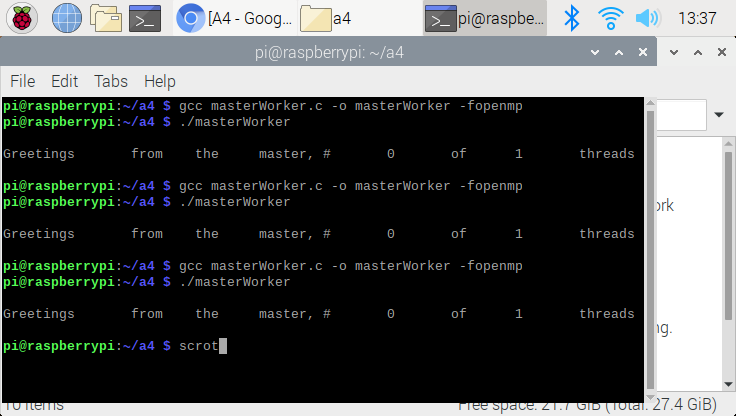
With pragma

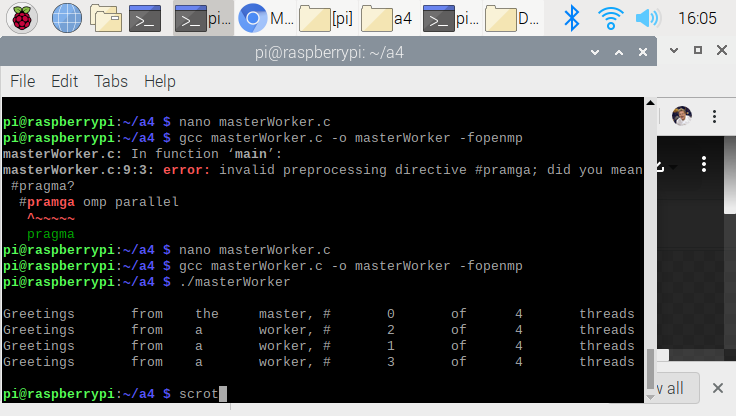


Barrier program is to ensure all designated parallel program is completed before moving to another section. From thefirst photo, the threads (4 threads from 0 to 3) are not being checked by the barrier. All threads are moved from “BEFORE” to “AFTER” then completed the program, then moved to another thread.

The second photo is with the barrier pragma. All threads are being checked by barrier,. This mean the thread would checked whether individual threads are completed their task. Once they complete all the task, then they move to another trask. That’s why all threads are “BEFORE” before moving to “AFTER.”

masterwork ScreenShot with pragma and without pragma.





The very first photo output is without the pragma. Next is with pragma. With pragma, we could see that the master is working with the worker to define the necessary pattern., we defined on difference between with and without pragma. The main purpose of this code is to define the common pattern and check whether the which threads are working on the program. When all workers are done with the processor, the master thread (in this case thread 0) will compile all the code and finished the program by giving us a result.