Hyoungjun Lee   
3/27/2020   
Computer Org,Programming   
Project A4 Task 3

Task 3

Foundation

What is race condition?   
-this condition is when a software, electronics or other system where the system’s function is in dependent on sequence, if you do not use properly, it causes a bug

What race condition is difficult to reproduce and debug?   
-when debugging race conditions, we cannot correct the programming and hard to reproduce time and energy in software development, interdependent interactions

8p) How can it be fixed? Provide an example from your Project\_A3(see spmd2.c)

-when we looking at spmd2.c the output was not coming out correctly, because how Raspberry PI organized memory, to fix this issue, we commented out line 5 and declared id and numThreads as the int data type

Change to below

#pragma omp parallel {

int id = omp\_get\_thread\_num();

int numThreads = omp\_get\_num\_threads();

printf("Hello from thread %d of %d\n", id, numThreads);

}

 Summaries the Parallel Programming Patterns section in the “Introduction to Parallel Computing\_3.pdf”(two  pages)  in  your  own  words  (one  paragraph,  no  more  than  150 words).-(12p)

-parallel programming divide into two categories , implementation and Algorithmic. Parallel code can be difficult to write, so programmers need to created parallel patterns, roughly guideline, This pattern uses both openMP and Message Passing interface. Concurrent execution mechanism divided into two major categories, Process and Thread control patterns and coordination patterns.

In  the  section  “Categorizing  Patterns”  in  the  “Introduction  to  Parallel Computing\_3.pdf” comparethe following:

oCollective synchronization (barrier) with Collective communication (reduction)

-collective communication, all the process have a specific point before execution, reduction is an operation that process can be organized to execute in efficient way, uses concurrent execution mechanism for parallel execution, in collective Synchronization, blocks all the processes until processes are reached specific point, it uses MPI\_Barrier()function, this is type of parallel application of computing

 oMaster-worker with fork join

-Master-work, main process is being divided into small chunks, distributed to several worker processes, in Fork join , used to execute parallel light weight processes and threads

-(26p) Dependency: Using your own words and explanation, answer the following:

(3p) Where can we find parallelism in programming?

-in programs, server machines, virtual reality, data bases

(6p) What is dependency and what are its types (provide one example for each)?

-input of one execution depends on the output, other elements, we have true dependence, output dependence, anti dependence

(3p) When  a  statement  is  dependent  and  when  it  is  independent  (Provide two examples)?

-statement is dependent when its values is getting computed, or it relies on statement for some output

Example: a = 5 b = 10 c = a+b   c = a + b is dependent on a = 5 b = 10 statements   
statement is independent when we do not need previous values   
a = 5 b = 10 c = 30 + 20    where c = 30+10 does not need value of a and b

(3p) When can two statements be executed in parallel?

-two statement in parallel can be executed if they don’t share any common data item, no dependency between cycle

(3p) How can dependency be removed?

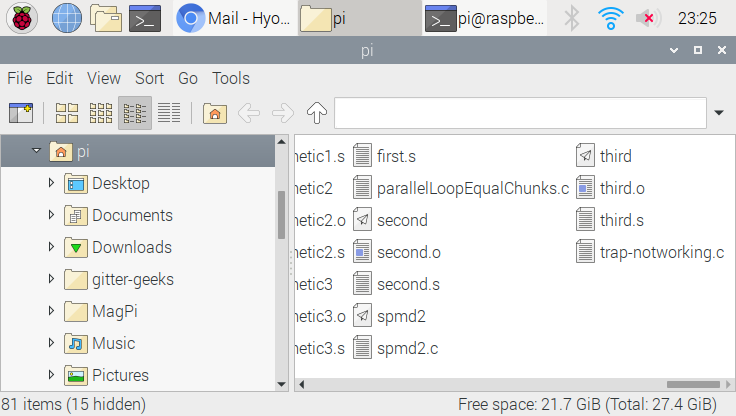
-rearranging and eliminating statements before execution

(8p) How do we compute dependency for the following two loops and what type/s of dependency

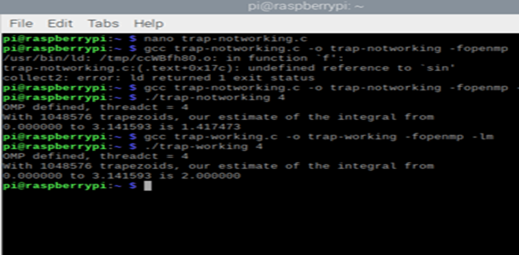
-examining in and out the loops, use two basic strategies to find dependency for the statement

Study the relationship between statement, unrolling of the loop into iterations, In the first loop, there is true dependency between loop index and a, and loop index a and be in second loop

Part B



Here’s program that I made for trap- not working



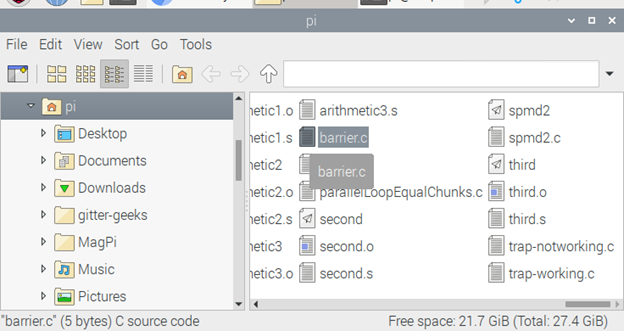
Trap-notworking compiled successfully and ran

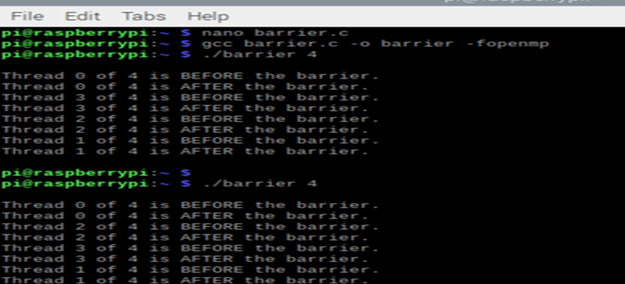
But the result was not correct since value of our integral at the first

After fixing the problem about integral, I ran it again with gcc trap-working-c

We get the correct result output

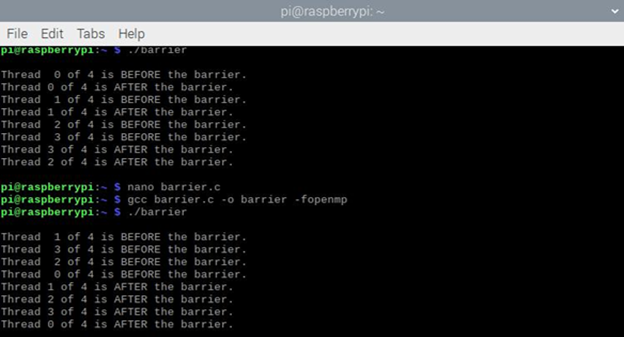
Reason for giving an incorrect output is because of race condition. So we used \ at the end of #pragma, save the last iteration.



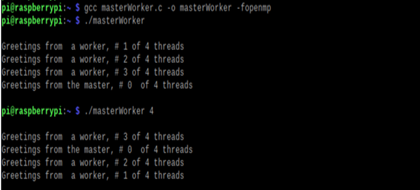


Without barrier above,

There is a difference between without barrier and with barrier, thread that without barrier print out before and after statement 1 by 1, but when running with barrier , all the threads stopped, until all other threads caught up



This is with the barrier



Worker master without pragma and with pragma

After creating code executable, ran the code,  one with pragmas, there’s no thread are being used, just sequential computation. Without parallel, I can see only 1 line output of the master