Operating Systems–1: Autumn 2023

Programming Assignment 2: Multi-threaded Solution for finding Vampire Numbers

Submission Deadline: 2nd December 2023, 9:00 pm

Goal:- The objective of this assignment is to develop an **efficient** multi-threaded program in **C** to find a list of vampire numbers in the first **N** numbers given as input.

Details of Vampire Numbers:- A number is known as a vampire number if you can take its digits, rearrange them into two new numbers with the same number of digits as each other, and then multiply them together to get back to the original number.

For example, look at 1260. These four digits can be rearranged into two 2-digit numbers, 21 and 60, which, if multiplied together, answer 1260. That makes 1260 a vampire number, with 21 and 60 being fangs. One other example is 1395 = 15 × 93. There are bigger vampire numbers and sometimes numbers that can have multiple pairs of fangs. Consider 125460: 204 × 615 = 125460 and 246 × 510 = 125460.

You can find more details about vampire numbers on this link : Vampire-Numbers.

Pseudocode for finding a vampire number :

```
// This program takes an integer ‘num’ and checks if it is a vampire number or not
bool VampireNum(int num) {
  digitCount = number of digits in num;
  if (digitCount is odd)
    return false;
  if (digitCount is 2)
    return false;
  for (A = each permutation of length digitCount/2) {
    for (B = each permutation of the remaining digits) {
      if (A or B starts with a zero)
        continue;
      if (both A and B end in a zero)
        continue;
      if (A*B == num)
        return true;
    }
  }
}
```

Multi-threaded Program Design: As a part of this assignment, you need to implement a **C** program to find vampire numbers till **N** and list them into a single file. The program partitions the numbers from 1 to N among **M** threads so that two threads do not work on the same number. Thus, each thread, T, will be responsible for a set of numbers. For each number in its set, the thread T will determine if the number is a perfect number or not. If it is, T will store it locally. After completion, T will share the set of numbers it identified with the main thread.

The logic for partitioning the N numbers among the M can be decided by you. It can be seen that intelligent partitioning of these numbers among the threads can greatly affect the performance of your program.

Input File:- As mentioned above, the input will consist of two parameters, **N** and **M**.

Output File:- For ease of understanding, a log file named OutFile will be created, and each thread will store information about the vampire number in the log file. Suppose Thread T tests some numbers and finds some of them to be vampire numbers; then, it will print the vampire number and its thread ID. In the end, the log file should also contain the total count of the vampire numbers in the given range, N. A sample log file can be as follows:

11: Found by Thread 1
101: Found by Thread 3
111: Found by Thread 2
.
.
.
1551: Found by Thread 7
.
.
.

Total Vampire numbers: 32

Report Details:- As a part of this assignment, you have to prepare a report that will describe the low-level design of your program and give an analysis of its output.

You have to submit a report for this assignment. The report should first explain the design of your program while explaining any complications that arose in the course of programming. Specifically, it must explain the logic of partitioning the N numbers among the M threads.

The report should contain the following important graphs which shows the performance of your program:

1. Time vs Size, N: In this graph, the y-axis will show the time taken by your algorithms. The x-axis will show the values of **k** varying from 10 to 20 in increments of 2 where $\mathbf{N=2^k}$. Have M,

the number of threads fixed at 8 for all these experiments.

2. Time vs Number of Threads, M: In this graph, like the previous graph, the y-axis will show the time taken by your algorithm. The x-axis will be the values of M, the number of threads varying from 1 to 16 (in powers of 2 *i.e*, 1,2,4,8,16). Have N fixed at 1000000 (10 lakhs) for all these experiments.

Submission Format:- You have to upload: (1) The source code in the following format: Assgn2Src-<RollNo>.c (2) Readme: Assgn2Readme-<RollNo>.txt, which contains the instructions for executing the program. (3) Report: Assgn2Report-<RollNo>.pdf, which contains all the details explained above.

Name the zipped document as: Assgn2-<RollNo>.zip. Please follow this naming convention. Otherwise, your assignment will not be graded.

Grading Policy:- The policy for grading this assignment will be -
(1) Design as described in report and analysis of the results: 50%; (2) Execution of the tasks based on description in readme: 40% (3) Code documentation and indentation: 10%.

Evaluation: The best of the marks obtained in Assignments 1 and 2 will be considered for the final grading.

Late Submission and Plagiarism Check: The late submission policy for this assignment is a penalty of 20% each day after the deadline for two days. Submissions after two days will not be considered for the evaluation.

Please keep in mind that all the submissions are subjected to plagiarism checks.