OperatingSystems Assignment_2
Es21btech11003

Report : Finding Perfect Number

1. Input_handling:

```
// input_handling
   FILE *input_fp = NULL;
   input_fp = fopen(argv[1] , "r");
   if(input_fp == NULL){
       printf("Error : Unable to open file");
       exit(1);
   }
   else{
       fscanf(input_fp , "%d %d" , &N , &K);
   }
```

Program takes input from a file name of which it take as a command line argument **argv[1]** . Using fopen & fscanf function it read first two space separated integer as N and K values .

2. Creating threads:

```
//creating K threads
   for(int i = 1 ; i <= K ; i++ ){

    // initializing and allocating memory to the elements of data_thread array
       thread_data_arr[i-1].thread_no = i;
       thread_data_arr[i-1].perfect_no = calloc((N/K + 1) , sizeof(int));

       /* create the thread */
       pthread_create(&tid[i-1], &attr, runner, (void*)&thread_data_arr[i-1]);
       /* wait for the thread to exit */
       pthread_join(tid[i-1],NULL);
   }
```

Using a loop:
1) we initialize the parameters required for each of the thread
2) Using **pthread_create** we create each thread and invoke **runner** function
3) pthread_join stops the main thread and waits for all other thread to finish before further executing.

3. Runner function:

Implementation of runner function :

1. Evaluates a subset of number such that work load is evenly distributed between thread.

2. Each thread iterate through a set of numbers(**j**) as :
   Starting with j = thread_no
   For each iteration , updating j = j + K while j is less than N

3. And “int i” is used for indexing for traversing through the perfect number array which corresponds to each thread and keeps track of perfect number found by that thread.

4. We call function Check_Perfect function for each j and update perfect_no array of corresponding index i , 1 if j was perfect ,else 0.

5. Each thread also create its own outfile , Outfiles handling :

```
// file_handling for each child
   FILE *thread_fp = NULL;
   char File_name[20];
   sprintf(File_name , "OutFile%d.txt" , thread_data->thread_no);
   thread_fp = fopen(File_name, "w" );
```

```
// Printing in OutFiles(For each thread)
       if(thread_data->perfect_no[i] == 1){
           fprintf(thread_fp , "%d : Is a perfect Number\n" , j);
       }
       else{
           fprintf(thread_fp , "%d : Is not a perfect Number\n" , j);
       }
```

   Updates Outfiles as soon as it checks for a number , for this it uses **fprintf() .**
   After iterating through the loop , using fclose() it closes the Outfile .

6. Finally each thread exits(pthread_exit(0);).

4. Outmain file:

After completion of all other thread , Main thread reads the perfect_no arrays corresponding to each thread and outputs a main file in format :
Thread1 : num1 num2
Thread2 : num3 num4

5. Check_perfect function:

Function evaluates an integer to be a perfect number or not.
It maintain a sum of factors(factor_sum)
Iterating through each number upto sqrt(n) and adding number and n/number to factor_sum
Checking the edge case , it returns 1 if n in perfect else 0.

(n is the number being evaluated)

Comments :

=> **`thread_data_arr`** is an array of struct data_thread for storing parameter for each thread.

int thread_no // keep thread number

int *perfect_no // used to keep track if each number if it is perfect or not

```
/* Keeps thread data which needs
to be passed as parameter */
typedef struct{
   int thread_no;
   int *perfect_no;
}data_thread;
```

=> pthread.h library is used for implementing and creating pthreads.

Output analysis :

Here we analyze output for N = 100 , K = 10

For sixth thread :

```
6th thread :
6 : Is a perfect Number
16 : Is not a perfect Number
26 : Is not a perfect Number
36 : Is not a perfect Number
46 : Is not a perfect Number
56 : Is not a perfect Number
66 : Is not a perfect Number
76 : Is not a perfect Number
86 : Is not a perfect Number
96 : Is not a perfect Number
```

Here each 6 is the only perfect number as given by output . the number evaluated by this process are incrementing with 10 and evaluation starts with process number.

MainOut file:

```
Process 1 :
Process 2 :
Process 3 :
Process 4 :
Process 5 :
Process 6 : 6
Process 7 :
```

```
Process 8 : 28
Process 9 :
Process 10 :
```

Here we can see for every process it outputs the perfect numbers corresponding to the process.
In this case 6 and 28 are the only perfect numbers from 1 to 100.
(For 6: factors(1,2,3) => sum = 6)
(For 28: factors(1,2,4,7,14)=> sum = 28)