OperatingSystems Assignment_1
Es21btech11003

Report : Finding Perfect Number

1. Input_handling:

```
// input_handling
   FILE *input_fp = NULL;
   input_fp = fopen("input.txt" , "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 named “input.txt” . Using fopen & fscanf function it read first two space separated integer as N and K values .

2. Forking child processes:

```
// forking child processes
       if(pro_id != 0){
           pro_id = fork()
       }
```

Fork is inside a while loop which runs K times , every time control flow of code at this part , it check it a processes is parent or a sibling process.
(check : (pid_process != 0) => parent process)
=> K times , this forks a child process from parent creating k child processes.

3. Child Process:

Implementation of child process :

1. Creates a output files , outputting list of numbers and whether if each of them are perfect or not.
2. It creates a shared memory object :

Name = “**Sharing**”

Size = `sizeof(int)*(N/K + 1);`

Using **shm_open()** and **mmap()** - it creates a sharable memory map with **ptr** as integer pointer . Outputs **“map failed_1”** if mmap fails to map the memory object

```
 /* create the shared memory object */
          fd = shm_open(name,O_CREAT | O_RDWR,0666);
```

```
            /* memory map the shared memory object */
            ptr = (int *)mmap(0, SIZE, PROT_READ | PROT_WRITE, MAP_SHARED,
fd, 0);
            if(ptr == MAP_FAILED){
                printf("Map Failed_1");
                exit(1);
            }
```

3. Each child process iterate through a set of numbers(**j**) as :

Starting with j = process_no

For each iteration , updating j = j + K while j is less than N

4. And “int i” is used for indexing for traversing through the memory map.
5. We call function Perfect_no for each j and update ptr of corresponding index i , 1 if j was perfect ,else 0.
6. Child process Outfiles handling :

```
// Printing in OutFiles(For each child process)
                if(ptr[i] == 1){
                    fprintf(child_fp , "%d : Is a perfect Number\n" , j);
                }
                else{
                    fprintf(child_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 .

7. Finally the child process exits.

4. Parent Process:

Implementation of Parent process

1. After fork() , parent process waits for each of its child process to complete before further execution
2. Similar to child process child process , we initialize all required for variables for sharing the memory buffer with the child process .
3. Using shm_open to open the shared memory object using its name and ptr for mapping . Parent process access memory buffer in read mode

```
// Creating/Opening MainOut file
            FILE *MainOut_fp = NULL;
            MainOut_fp = fopen("OutMain.txt" , "a");
```

4. MainOut file :

Iterating in a similar way as in child process , for each process it output the perfect numbers corresponding to that process

```
    Write Desired out in OutMain file */
fprintf(MainOut_fp , "Process %d : " , pro_no);
while(j <= N){
    if(ptr[i] == 1){
        // Printing in OutFiles
        fprintf(MainOut_fp, "%d " , j);
    }
    i++;
    j += K;
}
fprintf(MainOut_fp , "\n");
```

5. shm_unlink () unlinks the memory map

Further program closes the input file and returns

Comments :

=> The distribution of numbers across the child process in this way helps distribute the load more evenly

=> initially pro_id is initialized to -1 just to correctly correctly enter in the while there instantly it is updated to correct values

=> Perfect_no function evaluates and return 1 if a number is perfect or else 0

Output analysis :

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

For sixth process :
6th process.... :
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 files
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.