CS3510 Operating Systems 1
Report

Abhinav Gupta
ES15BTECH11002

November 25, 2017

1 Design Of the program

1.1 Primes-Proc.c

- We have used fork () defined in unistd.h to create k child processes.
- Each child processes created is assigned the work to check for the primality of a subset of numbers less than n.
- Child process exits independent of other processes once it finishes its task.
- wait command is added after the creation of all the child processes so that the parent waits for the child processes to complete before writing the primes to the file.
- A separate function is created to check for the primality of a number , we use the AKS primality check for doing this.
- A Boolean array of size N is maintained to store which of the numbers less than N are prime.
- After all the completion of all the tasks , the parent process writes the prime numbers evaluated into a file ('primes-proc.txt').

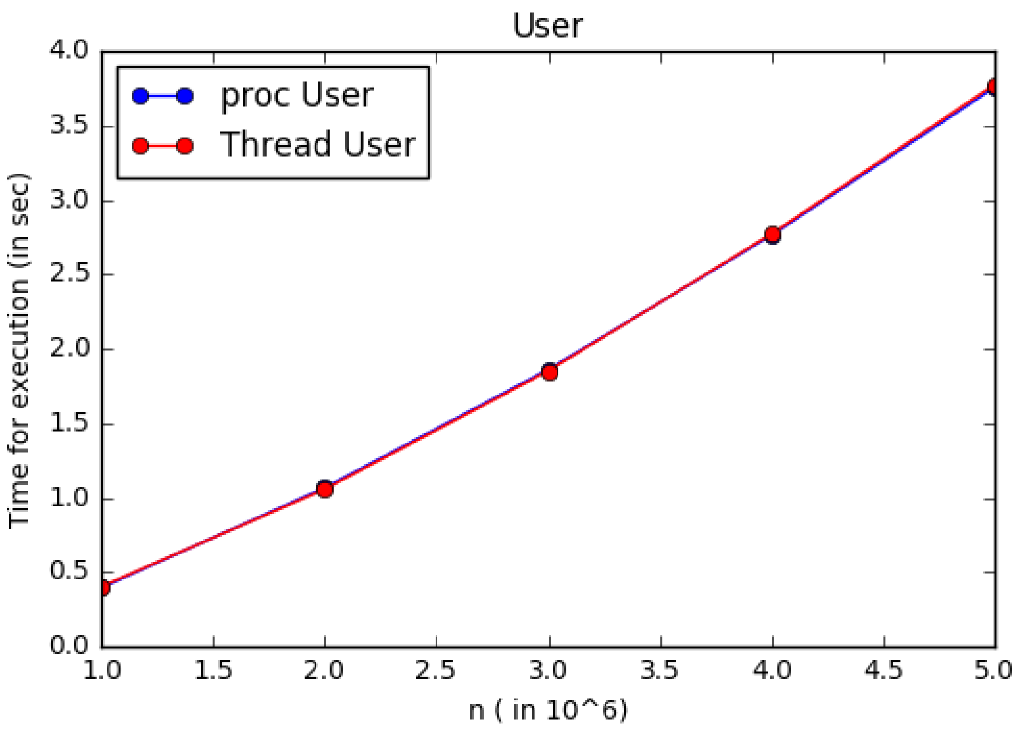
1.2 Primes-th.c

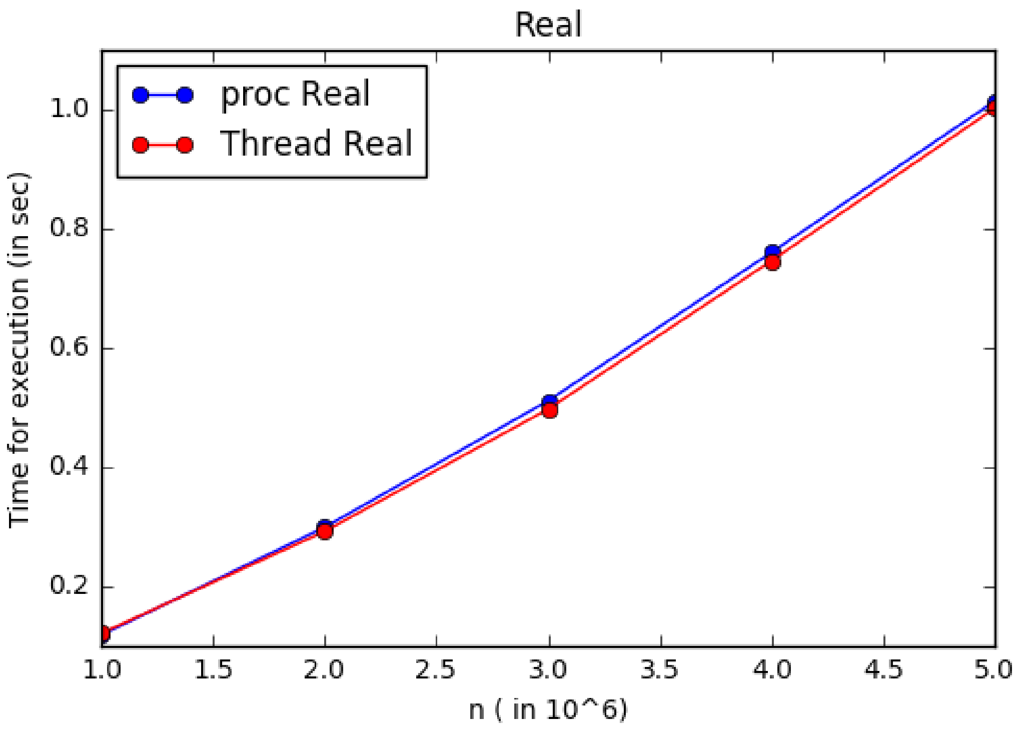
- We have used the pthread.h library to create k threads which perform the function specified to them.
- Each thread created performs primality test on a subset of numbers less than n independent of other threads , as the threads operate on different memory locations , mutex lock is not used.
- Threads exits independent of other threads once it finishes its task.
- pthread_join command is added after the creation of all the threads so that the parent thread waits for all the threads to complete their task before writing the primes to the file.

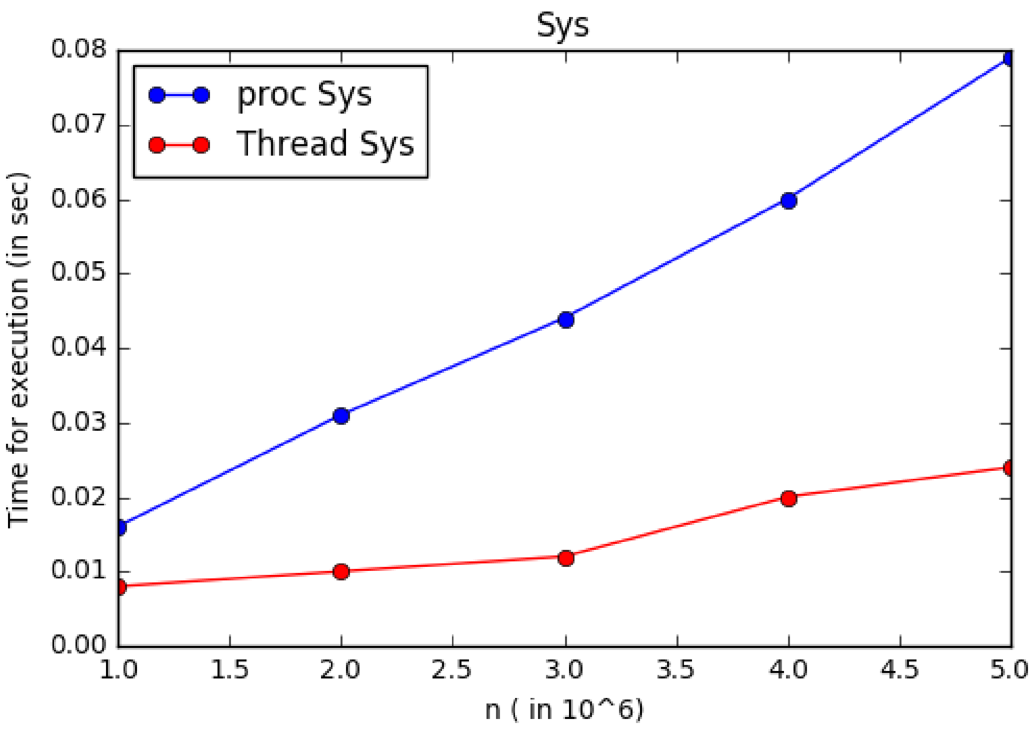
- A separate function is created to check for the primality of a number , we use the AKS primality check for doing this.
- A Boolean array of size N is maintained to store which of the numbers less than N are prime.
- After all the completion of all the tasks , the parent thread writes the prime numbers evaluated into a file ('primes-th.txt').
- No mutex locks are used for this program , as the threads operate independently on memory without interfering with each other.

2 Analysis Of the Output

- I have used the time module to calculate the Real,System ,User times for the entire process to run for both the methods.
- Below is the graph ,later the obesrvations are noted.







- The vaue used for comparision were for k=10 threads/processes.

- On Comparing the outputs from both primes-th.c and primes-proc.c we find that the performance of both the methods are almost the same for small values of n. As n grows the time taken for multi processing becomes more than that of multi threading.

- We see that in this case both the programs can execute the task parallely as we each thread/process acts on a different subset of data which do not interfere.
- Multi Threading takes less time as
  - CPU has lesser overhead to establish and terminate a thread.
  - It is faster for an operating system to switch between threads for the active CPU task than it is to switch between different processes.