Name: Ali Anser Reg # 0603

Parallel and Distributed Computing

Activity

CPU Specifications:

```
Architecture:          x86_64
  CPU op-mode(s):      32-bit, 64-bit
  Address sizes:       39 bits physical, 48 bits virtual
  Byte Order:          Little Endian
CPU(s):                4
  On-line CPU(s) list: 0-3
Vendor ID:             GenuineIntel
  Model name:          Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz
    CPU family:        6
    Model:             61
    Thread(s) per core: 2
    Core(s) per socket: 2
    Socket(s):         1
    Stepping:          4
    CPU(s) scaling MHz: 92%
    CPU max MHz:       2700.0000
    CPU min MHz:       500.0000
```

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <math.h>
#include <sys/time.h>

#define MAX 500000
#define THREADS_TO_USE 4

void time_function(const char * label, void (*func)()) {
	struct timeval start, end;
	gettimeofday(&start, NULL);

	func();

	gettimeofday(&end, NULL);

	double time_taken = (end.tv_sec - start.tv_sec) * 1e6;
	time_taken = (time_taken + (end.tv_usec - start.tv_usec)) / 1e6;

	printf("%s took %f seconds\n", label, time_taken);
	printf("------------------------\n");
}

// number of primes in the range 1 to MAX
void func1() {
	int count = 0;

	for(int i = 1; i <= MAX; ++i) {
		if(i < 2) {
			continue;
		}

		int prime = 1;

		for(int j = 2; j < i; ++j) {
```

```
                if(i % j == 0) {
                        prime = 0;
                        break;
                }
        }

        if(prime) {
                count++;
        }
    }

    printf("Func1: Total primes = %d\n", count);
}

// number of primes from 4 till MAX
void func2() {
    int count = 2;

    for(int i = 4; i <= MAX; ++i) {
        int prime = 1;

        for(int j = 2; j < i; ++j) {

            if(i % j == 0) {
                prime = 0;
                break;
            }
        }

        if(prime) {
            count++;
        }
    }

    printf("Func2: Total primes = %d\n", count);
}

// number of primes from 4 till MAX using sqrt optimization
void func3() {
    int count = 2;
```

```
	for(int i = 4; i <= MAX; ++i) {
		int prime = 1;

		int limit = sqrt(i);

		for(int j = 2; j <= limit; ++j) {

			if(i % j == 0) {
				prime = 0;
				break;
			}
		}

		if(prime) {
			count++;
		}
	}

	printf("Func3: Total primes = %d\n", count);
}

typedef struct {
	int start, end, count;
} Thread_data;

// Thread function to count primes in a range
void * thread_func(void * arg) {
	Thread_data * data = (Thread_data *)arg;
	data->count = 0;

	for(int i = data->start; i <= data->end; ++i) {

		if(i < 2) {
			continue;
		}

		int prime = 1;
		int root = (int)sqrt(i);
```

```
            for(int j = 2; j <= root; ++j) {

                if(i % j == 0) {
                    prime = 0;
                    break;
                }
            }

            if(prime) {
                data->count++;
            }
        }

        return NULL;
}

// number of primes from 4 till MAX using pthreads
void func4() {
        pthread_t threads[THREADS_TO_USE];
        Thread_data data[THREADS_TO_USE];
        int range = MAX / THREADS_TO_USE;
        int total = 0;

        for(int i = 0; i < THREADS_TO_USE; ++i) {
            data[i].start = i * range + 1;
            data[i].end = (i == THREADS_TO_USE - 1) ? MAX : (i + 1) * range;
            pthread_create(&threads[i], NULL, thread_func, &data[i]);
        }

        for(int i = 0; i < THREADS_TO_USE; ++i) {
            pthread_join(threads[i], NULL);
            total += data[i].count;
        }

        printf("Func4 (pthreads): Total primes = %d\n", total);
}

int main() {
        time_function("Func1", func1);
        time_function("Func2", func2);
```

```
    time_function("Func3", func3);
    time_function("Func4", func4);
}
```

Output:

```
Func1: Total primes = 41538
Func1 took 44.859748 seconds
------------------------
Func2: Total primes = 41538
Func2 took 40.565163 seconds
------------------------
Func3: Total primes = 41538
Func3 took 0.112759 seconds
------------------------
Func4 (pthreads): Total primes = 41538
Func4 took 0.057541 seconds
------------------------
```

1	Simple Version	44 seconds
2	Simple Version starting with 4	40 seconds
3	Sqrt Optimization	112 milliseconds
4	Using pthreads	57 milliseconds