Ayush Goyal
190905522 CSE D Roll 62

Operating Systems Week 6 Lab 8: Programs on Threads

1. Write a multithreaded program that generates the Fibonacci series. The program should work as follows: The user will enter on the command line the number of Fibonacci numbers that the program is to generate. The program then will create a separate thread that will generate the Fibonacci numbers,placing the sequence in data that is shared by the threads (an array is probably the most convinient data structure). When the thread finishes execution the parent will output the sequence generated by the child thread. Because the parent thread cannot begin outputting the Fibonacci sequence until the child thread finishes, this will require having the parent thread wait for the child thread to finish.

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>

void* thread_code(void* args){
	int *arr = (int*)args;
	int n = arr[0];
	arr[1] = 0;
	arr[2] = 1;
	for(int i=3;i<=n;i++){
		arr[i] = arr[i-1] + arr[i-2];
	}
}

int main(int argc, char* argv[]){
	if(argc < 2){
		printf("Not enough arguments!\n");
		exit(EXIT_FAILURE);
	}
	int n = atoi(argv[1]);
	int* arr = (int*)calloc(n+1, sizeof(int));
	arr[0] = n;
	pthread_t thread;
	pthread_create(&thread, 0, &thread_code, (void*)arr);
	pthread_join(thread, 0);
	printf("The Fibonacci Series is: ");
	for(int i=1;i<=n;i++){
		printf("%d ", arr[i]);
	}
	printf("\n");
	return 0;
}
```

We use the “gcc -pthread -o l6q1 l6q1.c” command in the terminal to compile this code and the output is as shown below:

Output:

2. Write a multithreaded program that calculates the summation of non-negative integers in a separate thread and passes the result to the main thread.

Code:

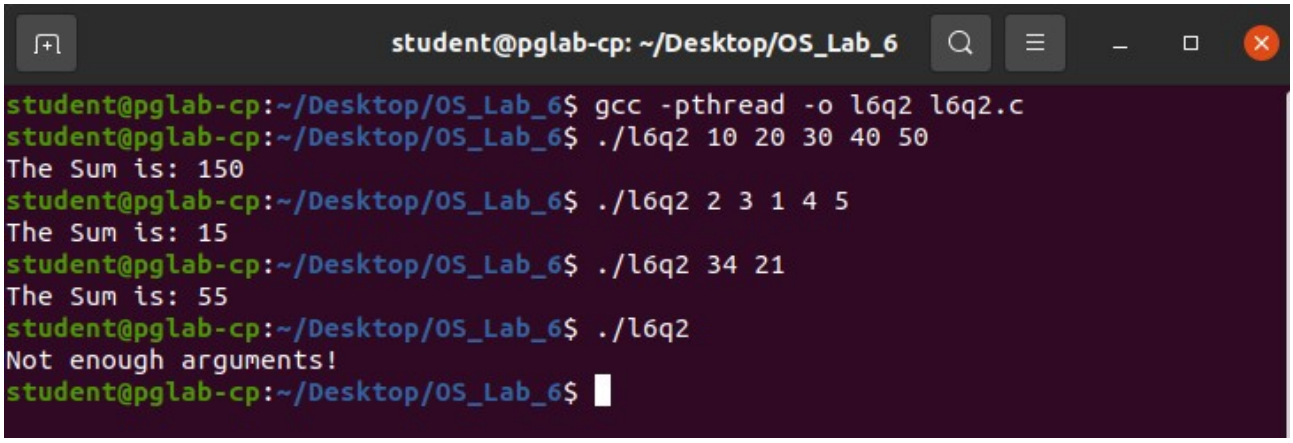
```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>

void* thread_code(void* args){
	int *arr = (int*)args;
	int n = arr[0];
	int *res = (int*)malloc(sizeof(int));
	*res = 0;
	for(int i=1;i<=n;i++){
		*res+=arr[i];
	}
	return (void*)res;
}

int main(int argc, char* argv[]){
	if(argc<2){
		printf("Not enough arguments!\n");
		exit(EXIT_FAILURE);
	}
	int n = argc - 1;
	int* arr = (int*)calloc(n+1, sizeof(int)), *res;
	arr[0] = n;
	for(int i=0;i<n;i++){
		arr[i+1] = atoi(argv[i+1]);
	}
	pthread_t thread;
```

```
	pthread_create(&thread, 0, &thread_code, (void*)arr);
	pthread_join(thread, (void**)&res);
	printf("The Sum is: %d\n", *res);
	return 0;
}
```

Ouput:



3. Write a multithreaded program for generating prime numbers from a given starting number to the given ending number.

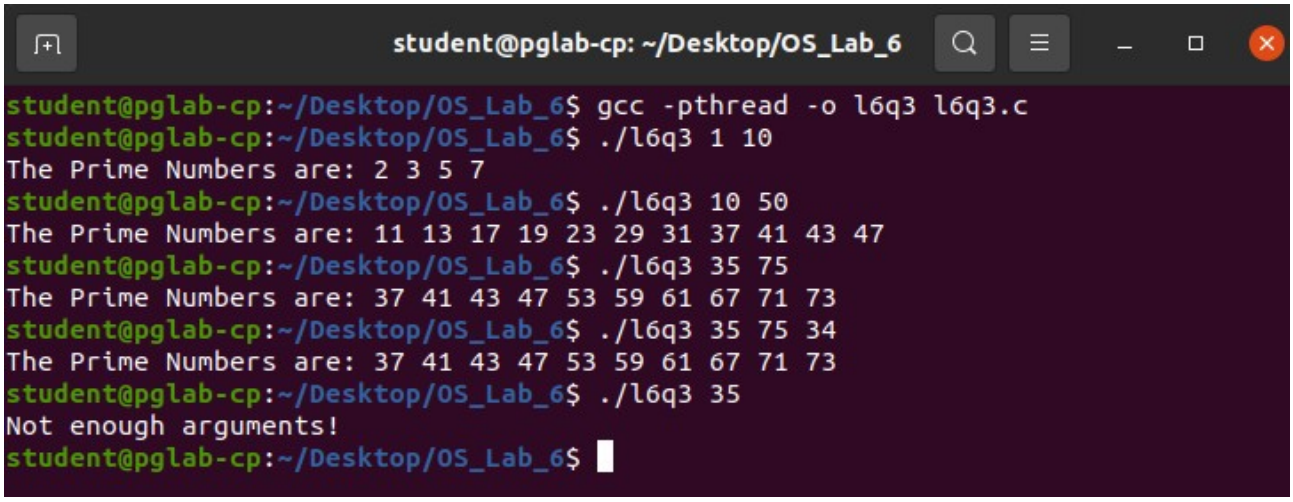
Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>

void* thread_code(void* args){
	int *arr = (int*)args;
	int lower_limit = arr[0];
	int upper_limit = arr[1];
	int *sieve = (int*)calloc(upper_limit+1, sizeof(int));
	for(int i=0;i<=upper_limit;i++){
		sieve[i] = 1;
	}
	sieve[0] = 0;
	sieve[1] = 0;
	for(int i=2;i*i<=upper_limit;i++){
		if(sieve[i] == 1){
			for(int j = i*i; j<=upper_limit; j+=i){
				sieve[j] = 0;
			}
		}
	}
	return (void*)sieve;
}
```

```
int main(int argc, char* argv[]){
	if(argc<3){
		printf("Not enough arguments!\n");
		exit(EXIT_FAILURE);
	}
	int lower_limit = atoi(argv[1]);
	int upper_limit = atoi(argv[2]);
	int packet[] = {lower_limit, upper_limit}, *result;
	pthread_t thread;
	pthread_create(&thread, 0, &thread_code, (void*)packet);
	pthread_join(thread, (void**)&result);
	printf("The Prime Numbers are: ");
	for(int i=lower_limit;i<upper_limit;i++){
		if((result)[i])
			printf("%d ", i);
	}
	printf("\n");
	return 0;
}
```

Output:



4. Write a multithreaded program that performs the sum of even numbers and odd numbers in an input array. Create a separate thread to perform the sum of even numbers and odd numbers. The parent thread has to wait until both the threads are done.

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
```

```
void* add_odd(void* args){
	int *arr = (int*)args;
	int n = arr[0];
	int *res = (int*)malloc(sizeof(int));
	*res = 0;
	for(int i=1;i<=n;i++){
		if(arr[i] & 1)
			*res += arr[i];
	}
	return (void*)res;
}


void* add_even(void* args){
	int *arr = (int*)args;
	int n = arr[0];
	int *res = (int*)malloc(sizeof(int));
	*res = 0;
	for(int i=1;i<=n;i++){
		if(!(arr[i] & 1))
			*res += arr[i];
	}
	return (void*)res;
}


int main(int argc, char* argv[]){
	if(argc < 2){
		printf("Not enough arguments!\n");
		exit(EXIT_FAILURE);
	}
	int n = argc - 1;
	int* arr = (int*)calloc(n+1, sizeof(int)), *resEven, *resOdd;
	arr[0] = n;
	for(int i=0;i<n;i++){
		arr[i+1] = atoi(argv[i+1]);
	}
	pthread_t threadOdd, threadEven;
	pthread_create(&threadOdd, 0, &add_odd, (void*)arr);
	pthread_create(&threadEven, 0, &add_even, (void*)arr);
	pthread_join(threadOdd, (void**)&resOdd);
	pthread_join(threadEven, (void**)&resEven);
	printf("The Sum of Odd numbers is: %d\n", *resOdd);
	printf("The Sum of Even numbers is: %d\n", *resEven);
	return 0;
}
```

Output:

```
student@pglab-cp: ~/Desktop/OS_Lab_6
student@pglab-cp:~/Desktop/OS_Lab_6$ gcc -pthread -o l6q4 l6q4.c
student@pglab-cp:~/Desktop/OS_Lab_6$ ./l6q4 1 2 3 4 5 6 7 8 9 10
The Sum of Odd numbers is: 25
The Sum of Even numbers is: 30
student@pglab-cp:~/Desktop/OS_Lab_6$ ./l6q4 1 2 3 4 5 6
The Sum of Odd numbers is: 9
The Sum of Even numbers is: 12
student@pglab-cp:~/Desktop/OS_Lab_6$ ./l6q4 1 2 3 4
The Sum of Odd numbers is: 4
The Sum of Even numbers is: 6
student@pglab-cp:~/Desktop/OS_Lab_6$ ./l6q4 1
The Sum of Odd numbers is: 1
The Sum of Even numbers is: 0
student@pglab-cp:~/Desktop/OS_Lab_6$ ./l6q4
Not enough arguments!
student@pglab-cp:~/Desktop/OS_Lab_6$
```

THE END