

Prof. John Svadlenka
Date: 10-01-2020
CSCI 340 14[48182]
Author: Rojan K C

Project 1 write up and program design

Table of Contents

Overview and Purpose	2
Program Design.....	2
Part2 and part3 Program Design	3
Struct	3
Runner	3
Main	4
Results/Output	4
Compilation Instructions.....	4
Issues Encountered and Resolved	4
Tools / Library Used.....	4

Overview and Purpose

This project is designed to simulate a threading operation over summation. This program reflects on how multi-threading works while sharing a memory. We find a sum up to certain target value by dividing the work into the threads. Part2 of the project will have the target value divided up to 10 threads while part3 of the project will have the same target value divided up to 100 threads.

Program Design

The program takes an arguments as below:

`./rkc_thd.exe <target value> <num of child> <num of grandchild> <outputfile>`

Below is the logistics that are considered in program design.

- i) Divide program to one thread operation (given the code).
- ii) Use void runner to run the thread multiple times.
- iii) Use an array to store struct args including the ThreadID.
- iv) Use STRUCT to communicate the arguments between the threads and work as a middleware between thread creation method and runner method.

- v) Use fprintf() method from FILE to write all the printf() into the files as a final output. (rkc_rslt.txt)
- vi) Since it's easier to divide the work among threads, we use pipeline like architecture to compute individual summation using different threads.

The main program executes part2() and part3() function. Below is further explanation of each program design.

Part2 and part3 Program Design

Run_part2(argc, argv[1], argv[2], argv[4]);

argv[1] is the target sum, argv[2] is number of thread and argv[4] is outputfile.

We use pipeline architecture to compute the

- i) Take arguments and convert it into int using atol();
- ii) initialize struct created outside main as an array of length given in arguments.
- iii) Initialize and create an array of threads and attributes.
- iv) Since we are dividing the summation work equally among child, we divide the number we want to sum up to by number of threads and if we subtract 1 to it we get difference between start and end index.
- v) We use for loop to assign struct args the initial and ending value and create thread.
- vi) Thread created will use the values from struct args to do summation.
- vii) We continue to do so until all the threads are created.
- viii) Finally, for output, we print out the values that's been saved in struct args and write the output into the file using fprintf()

For part3, we follow the same rules as part2 which will divide the total summation into 100 threads which are grandchildren threads.

Struct

We want to communicate the start and end value of each thread to next, so we create a struct arguments array which will have our sum, start index and end index.

It is essential to have total sum within struct because it will help us keep track of all the sum that's been done upto.

Runner

We use runner function (which was given) to calculate the sum and store it into the answer variable of struct arguments. We get the start and end index from struct and do the summation in Runner method.

Main

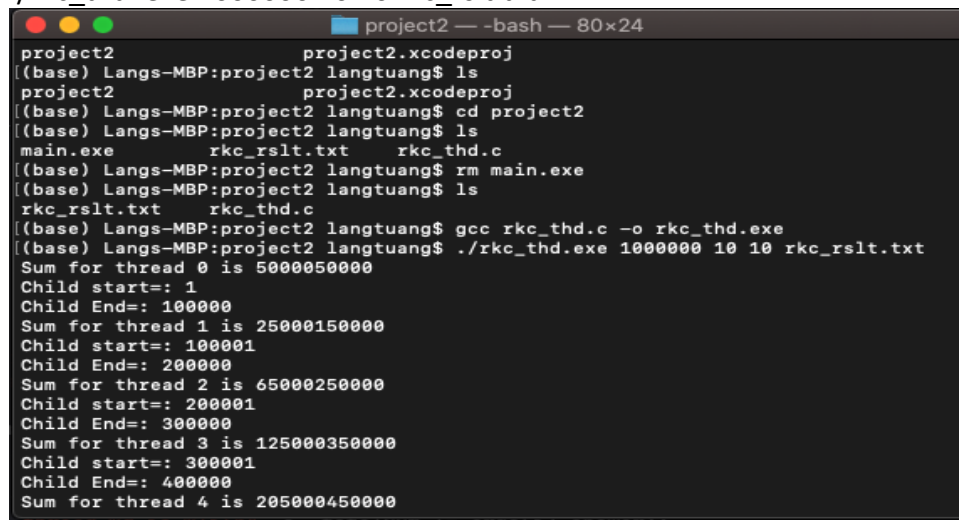
Our main method simply calls two function, run_part2 and run_part3 with all the arguments passed from command line.

Results/Output

All the results are printed out in console and written out in file given as an argument by the user. (For our experiment we use: rkc_rslt.txt)

Compilation Instructions

- i) Use terminal(for MAC) to navigate to the project folder.
- ii) gcc rkc_thd.c -o rkc_thd.exe
- iii) ./rkc_thd.exe 1000000 10 10 rkc_rslt.txt



```
project2 project2.xcodeproj
((base) Langs-MBP:project2 langtuang$ ls
project2 project2.xcodeproj
((base) Langs-MBP:project2 langtuang$ cd project2
((base) Langs-MBP:project2 langtuang$ ls
main.exe      rkc_rslt.txt  rkc_thd.c
((base) Langs-MBP:project2 langtuang$ rm main.exe
((base) Langs-MBP:project2 langtuang$ ls
rkc_rslt.txt  rkc_thd.c
((base) Langs-MBP:project2 langtuang$ gcc rkc_thd.c -o rkc_thd.exe
((base) Langs-MBP:project2 langtuang$ ./rkc_thd.exe 1000000 10 10 rkc_rslt.txt
Sum for thread 0 is 5000050000
Child start=: 1
Child End=: 100000
Sum for thread 1 is 25000150000
Child start=: 100001
Child End=: 200000
Sum for thread 2 is 65000250000
Child start=: 200001
Child End=: 300000
Sum for thread 3 is 125000350000
Child start=: 300001
Child End=: 400000
Sum for thread 4 is 205000450000
```

Issues Encountered and Resolved

- a) Thread Communication of (void *param) of runner function while creating thread
We used STRUCT to save those parameters as an arguments array and we keep track of start and end index for each thread.
- b) Writing in output file
We used fprintf() to write it into the output file.

Tools / Library Used

XCode, Command Line tool, pthread, fcntl, Struct