21:198:332:01 Operating Systems

Project I - Multithreading Programming (20 points)

Part I - Compile and Run C Program with Pthreads

Compile and run the example of summation of integers discussed in class.

1. The C program code of summation of integers is provided. (***thrd-posix.c***)

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

int sum; /* this data is shared by the thread(s) */

void *runner(void *param); /* the thread */

int main(int argc, char *argv[])
{
pthread_t tid; /* the thread identifier */
pthread_attr_t attr; /* set of attributes for the thread */

if (argc != 2) {
        fprintf(stderr,"usage: a.out <integer value>\n");
        /*exit(1);*/
        return -1;
}

if (atoi(argv[1]) < 0) {
        fprintf(stderr,"Argument %d must be non-negative\n",atoi(argv[1]));
        /*exit(1);*/
        return -1;
}

/* get the default attributes */
pthread_attr_init(&attr);

/* create the thread */
pthread_create(&tid,&attr,runner,argv[1]);

/* now wait for the thread to exit */
pthread_join(tid,NULL);

printf("sum = %d\n",sum);
}
                                                              52,1          50%
```

2. Compile the ***thrd-posix.c*** by using **gcc thrd-posix.c -lpthread**. (Note: the **-lpthread** is needed for linking the **Pthreads** library)

```
osc@ubuntu:~/p1$ gcc thrd-posix.c -lpthread_
```

3. The default output executable is ***a.out***. Run the ***a.out*** by typing **./a.out 10**, where **10** means the summation of integers from 1 to **10**. (You can change 10 to any positive integer.) Then, you should have the output "`sum = 55`".

```
osc@ubuntu:~/p1$ gcc thrd-posix.c -lpthread
osc@ubuntu:~/p1$ ./a.out 10
sum = 55
osc@ubuntu:~/p1$
```

Part II - Parallelize Summation with Multithreading

1. The C program in Part I only creates one new thread to calculate the summation (`pthread_create(&tid,&attr,runner,argv[1]);`). We can further improve the performance of the computation through parallelization.

2. The idea is that we can divide the integers we need to sum into multiple (M) partitions. For example, if we need to sum 1 to 10, we can divide them into two (M=2) partitions, i.e., 1 to 5 and 6 to 10. Then, we **`create`** (`pthread_create`) two new threads to calculate the summation of each partition simultaneously (The results are 15 and 40), and get the final summation (55) by adding the results from all partitions up (15 + 40 = 55) after **`join`** (`pthread_join`).

3. Design a C program with Pthreads to parallelize the summation of integers by modifying the code in Part I with the following requirements:

    a. The number of partitions (**M**) is determined through the parameters of the executable. The format is **./a.out N M**, where **N** means the summation from 1 to **N,** and **M** means that we divide the integers into **M** partitions. For the previous example in Part II.2, **N** is 10 and **M** is 2. (obviously, **N** $\geq$ **M**)
    b. Create one thread for each partition. For example, if **M** = 4, you should create four new threads to calculate the partial sum.
    c. Careful design your shared data among multiple threads to get the correct result.
    d. The program can print out partial sums and final result. For example, if you run **./a.out 10 2**, the printout should be like:

```
sum of 1 to 5 = 15
sum of 6 to 10 = 40
sum of 1 to 10 = 55
```

4. Test your program with N = 256, M = 8.

Extra Credits (5 Points)

1. Implement Part II in Java.

What to submit for this project?

1. You can work individually or in groups. If you choose to work in group, each group cannot have more than **two** students and only one of the group members needs to submit the project report. Please include the names of all your group members in the report.
2. For Part I, the screenshot of the output should be included in your report.
3. For Part II, a brief discussion on your design, your program source code, and the screenshot of the testing output (N = 256, M = 8) should be included in your report.
4. You also need to upload your source file (.c file) in Part II along with the report.
5. For the extra credits, the source file (.java file) and screenshot of the output are needed.