Operating Systems

Active Learning:
Multithread Programming

AI Software, Gachon University

Objective

- Practice with multi-thread
- Comparing computational time between single/multi-thread
- Apply mutex lock for multi-thread for synchronization
- Apply time check code

Thread handling

- Create
  - int pthread_create(pthread_t * tid, pthread_attr_t * attr, void * (*start_routine)(void *), void * arg);
  - tid: thread id
  - attr: attribute
    - Initialized by pthread_attr_init(&attr);
    - It can be NULL, when using default
    - (*start_routine)(void *): running function
    - arg: function parameters
- Created thread will begin at starting point of a running function.

Thread handling

- Terminate
  - int pthread_exit (void *status);
  - Implemented at the end of running function
- Wait
  - int pthread_join (pthread_t tid, void **status_ptr);
  - Similar with wait() system call

Mutex Lock and Synchronization

- POSIX pthread library

```
#include <pthread.h>

pthread_mutex_t mutex;

/* create the mutex lock */
pthread_mutex_init(&mutex,NULL);
```

```
/* acquire the mutex lock */
pthread_mutex_lock(&mutex);

/* critical section */

/* release the mutex lock */
pthread_mutex_unlock(&mutex);
```

Computational time check

- clock_gettime() function
  - Implemented in time.h (include <time.h>)
  - A return value is current tick of the computer.

  - struct timespec start, finish;
  - double elapsed;

  - clock_gettime(CLOCK_MONOTONIC, &start);
  - /* ... */
  - clock_gettime(CLOCK_MONOTONIC, &finish);

  - elapsed = (finish.tv_sec - start.tv_sec); // seconds
  - elapsed += (finish.tv_nsec - start.tv_nsec) / 1000000000.0; // nanoseconds

Today's mission

- Get the number of prime numbers from 2 to 2000000
- Single-thread version will be given.
- Produce a multi-thread counterpart (using 20 threads).
- When running the code,
  - Thread message
  - Processing time
  - The number of prime numbers within the range
  - Should be displayed

Today's mission

- Single-thread code
  - hw5_single.c
  - single thread (no thread creation).
  - Compile: gcc -o hw5_single hw5_single.c
- Multi-thread code
  - hw5_multi.c
  - Use 20 threads. Each thread should compute a divided range.
    (e.g. thread#1 counts from 0 to 9999999,
    thread#2 counts from 10000000 to 19999999
    thread#3 counts from 20000000 to 29999999 …)
  - Compile: gcc –o hw5_multi hw5_multi.c -lpthread

Today's mission

- Using input arguments when running single/multi-thread
  - Ex) ./hw5_single 2000000
  - Ex) ./hw5_multi 2000000

Examples

```
os @ubuntu:~/homework/hw5$ gcc -o hw5_single hw5_single.c
os @ubuntu:~/homework/hw5$ gcc -o hw5_multi hw5_multi.c -lpthread
os @ubuntu:~/homework/hw5$ ls
hw5_multi  hw5_multi.c  hw5_single  hw5_single.c
os @ubuntu:~/homework/hw5$
```

Examples

```
OS @ubuntu:~/homework/hw5$ ./hw5_single 2000000
elapsed time: 8.794132 sec
The number of prime numbers between 1~2000000 is 148933
OS @ubuntu:~/homework/hw5$ ./hw5_multi 2000000
thread for range (0 ~ 99999)
thread for range (100000 ~ 199999)
thread for range (1200000 ~ 1299999)
thread for range (300000 ~ 399999)
thread for range (1300000 ~ 1399999)
thread for range (200000 ~ 299999)
thread for range (1400000 ~ 1499999)
thread for range (1100000 ~ 1199999)
thread for range (1500000 ~ 1599999)
thread for range (1600000 ~ 1699999)
thread for range (1700000 ~ 1799999)
thread for range (1800000 ~ 1899999)
thread for range (1900000 ~ 1999999)
thread for range (400000 ~ 499999)
thread for range (500000 ~ 599999)
thread for range (600000 ~ 699999)
thread for range (1000000 ~ 1099999)
thread for range (900000 ~ 999999)
thread for range (800000 ~ 899999)
thread for range (700000 ~ 799999)
elapsed time: 8.925528 sec
The number of prime numbers between 1~2000000 is 148933
OS @ubuntu:~/homework/hw5$
```

Today's mission

- Single-thread code

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define true 1
#define false 0

int cntPrime(int num)
{
    int cnt = 0;
    int i;

    for (i=1; i*i<num; i++) {
        if (num%i == 0) cnt++;
    }
    if (i*i == num) cnt++;
    if (cnt == 1) return true;
    else return false;
}

int main(int argc, char* argv[])
{
    int cnt = 0;
    int range = atoi(argv[1]);

    struct timespec start, finish;
    double elapsed;
    clock_gettime(CLOCK_MONOTONIC, &start);


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

        if (cntPrime(i) == true) cnt++;
    }

    clock_gettime(CLOCK_MONOTONIC, &finish);
    elapsed = (finish.tv_sec - start.tv_sec);
    elapsed += (finish.tv_nsec - start.tv_nsec) / 1000000000.0;

    printf("elapsed time: %f sec \n", elapsed);
    printf("The number of prime numbers between 1~%d is %d\n", range, cnt);

    return 0;

}
```

Today's mission

- Multi-thread hint

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

#define true 1
#define false 0

int cnt = 0;
pthread_mutex_t mutex;

int cntPrime(int num)
{
    int cnt = 0;
    int i;
    if (num < 2) return false;

    for (i=1; i*i<num; i++) {
        if (num%i == 0) cnt++;
    }
    if (i*i == num) cnt++;
    if (cnt == 1) return true;
    else return false;
}

void *partial_prime(void* param)
{
    int range = *(int *)param;
    printf("thread for range (%d ~ %d)\n", range, range+99999);

    // add logic to count the number of prime numbers within the range (increase cnt with mutex lock)

    pthread_exit(0);
}
```

```c
int main(int argc, char* argv[])
{
    int range = atoi(argv[1]);

    struct timespec start, finish;
    double elapsed;

    // add logic to get time clock (i.e., start)

    // add logic to initialize mutex lock

    int num_thread = 0;
    pthread_t tids[1024];
    int limit[1024];

    int idx = 0;
    while (idx < range) {

        limit[num_thread] = idx;
        // add logic to create threads
        num_thread++;
        idx += 100000;
        if (idx+1 > range) idx = range;
    }

    // add logic to wait all the threads until they finish the prime number computation

    // add logic to measure the time clock difference

    printf("elapsed time: %f sec \n", elapsed);
    printf("The number of prime numbers between 1~%d is %d\n", range, cnt);


    return 0;

}
```

Due

- Due: 2022. 05. 25. 23:59

- Submission: ()
  - hw5_multi_”student id”.c: insert your id + name in the first line of the code!!
  - A word file where you should comment multi-thread and mutex related codes.

- Scoring criteria
  - Build code : 50
  - Correct multi-thread result (also code): 30
  - The word file with comments: 20