CIS 415 Operating Systems

Assignment <3> Report Collection

Submitted to:
Prof. Allen Malony

Author:
Jay (Hyo Jae) Shin

Report

To be honest, I really enjoyed this project, in fact it was very hard and very stressful and took me so long, still not fully done yet. I have imagined how the instagram server would work, how this server works driving people crazy. Synchronization is what CS students must know before graduate, it is that important. We will use it in any side of the computer area (back-end or front-end, wherever).

This insta quack will create pools of threads and assign different calculations to each thread. After checking availability of each thread pool space, a new calculator setting will be set up to perform concurrently. Because all my calculators are sharing one resource, we need to implement synchronization to use safely. Users want to upload how hot they are, how rich they are, and how cool they are asap, and somehow Users also want to see how hot others are, how rich others are and how cool others are asap, but it is the Instagram, not grams.

I was able to finish until step 4 which is reading from the input file, running and creating thread from it. I was not able to copy the returning data and send it to html which is part 5. I did not make other c or h files bc I realized that going up and down was more efficient for this project. So when you grade my code shorten and open when you have to read it.

```
143 › void pool_init () {…}
149 › void q_init (TEQ *queue, char *TEQ_ID, int size) {…}
159 › void te_init (topicEntry *TE, char *url, char *cap) {…}
164 › void freeTEQ(TEQ *queue) {…}
```

My main will read input.txt in the input file, and tokenize each line to read commands. Once it reads the command, it will create a thread when the user enters add pub or sub. Main will initialize requirements, create threads, and signal to it's threads. As the picture below shows part 3 and it will go to each thread to perform different calculations.

```
jay@jay-VirtualBox:~/Desktop/cis-415/projects/project3$ valgrind --track-origins=yes ./a.out input.txt
==186726== Memcheck, a memory error detector
==186726== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==186726== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==186726== Command: ./a.out input.txt
==186726==
CREATION SUCCESS
CREATION SUCCESS
CREATION SUCCESS
input/pub1.txt
ADD SUCCESS
input/pub2.txt
type: publisher thread ID: 90666752    waiting
type: publisher thread ID: 99059456    waiting
ADD SUCCESS
input/pub3.txt
ADD SUCCESS
15.000000
type: publisher thread ID: 107452160    waiting
DELTA SUCCESS
input/sub1.txt
ADD SUCCESS
input/sub2.txt
type: sublisher thread ID: 115844864    waiting
ADD SUCCESS
input/sub3.txt
type: sublisher thread ID: 124237568    waiting
ADD SUCCESS
type: sublisher thread ID: 132630272    waiting
BROADCASTING
type: subscriber        thread ID: 132630272    begin
type: subscriber        thread ID: 124237568    begin
type: subscriber        thread ID: 115844864    begin
type: publisher thread ID: 107452160    begin
pushing: Queue goats -URL: https://cdn.pixabay.com/photo/2018/03/07/19/51/animal-3206941_960_720.jpg -Caption:
 friends of a goat
type: publisher thread ID: 99059456    begin
```

```
type: cleanup   thread ID: 141022976    waiting for 15
popping: Queue Funny_dogs -Queue is empty, nothing to pop
type: cleanup   thread ID: 141022976    waiting for 15
popping: Queue Funny_dogs -Queue is empty, nothing to pop
type: cleanup   thread ID: 141022976    waiting for 15
popping: Queue Funny_dogs -Queue is empty, nothing to pop
type: cleanup   thread ID: 141022976    waiting for 15
popping: Queue Funny_dogs -Queue is empty, nothing to pop
^C==186726==
==186726== Process terminating with default action of signal 2 (SIGINT)
==186726==    at 0x4869CD7: __pthread_clockjoin_ex (pthread_join_common.c:145)
==186726==    by 0x10973F: main (in /home/jay/Desktop/cis-415/projects/project3/a.out)
==186726==
==186726== HEAP SUMMARY:
==186726==     in use at exit: 127,520 bytes in 131 blocks
==186726==   total heap usage: 527 allocs, 396 frees, 143,941 bytes allocated
==186726==
==186726== LEAK SUMMARY:
==186726==    definitely lost: 80,456 bytes in 60 blocks
==186726==    indirectly lost: 306 bytes in 41 blocks
==186726==      possibly lost: 272 bytes in 1 blocks
==186726==    still reachable: 46,486 bytes in 29 blocks
==186726==         suppressed: 0 bytes in 0 blocks
==186726== Rerun with --leak-check=full to see details of leaked memory
==186726==
==186726== For lists of detected and suppressed errors, rerun with: -s
==186726== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

jay@jay-VirtualBox:~/Desktop/cis-415/projects/project3$
```

However, because of creation of clean-up-thread after start, I end up having an infinite loop in emptiness. Due to this conclusion, I was not able to finish the getEntry and part 5.

I have spent most of my week fixing this issue (infinite loop, I believe). I am still confused with where I am having this trouble, but I think this is coming from calling cleaning wrong. This happens, I believe, because the main won't let clean to stop or clean does not stop.

I made a global variable work to check whether works are not done yet.

```
void *cleanup() {
  sleep(delta);
  int ret, idx = 0;
  while (work) {
    printf("\n\n\nEXTERMINATOR\n");
    if (idx == MAXQUEUES) idx = 0;
    ret = dequeue(registry[++idx]->topic);
    while (ret == 0) {
      printf("type: cleanup\tthread ID: %ld\twaiting for 15\n", pthread_self());
      ret = dequeue(registry[idx]->topic);
      sched_yield();
      sleep(delta);
    }
  }
  pthread_exit(0);
}
```

```
int work;
pthread_t cleanup_thread;
float delta;
pthread_cond_t ptc = PTHREAD_COND_INITIALIZER;
pthread_mutex_t ptm = PTHREAD_MUTEX_INITIALIZER;
```

```
  }
  //Join the thread-pools
  for (int i = 0; i < MAXQUEUES; i++) {
    pthread_join(pubpool[i].tid, NULL);
    pthread_join(subpool[i].tid, NULL);
  }
  work = 0;
  pthread_join(cleanup_thread, NULL);
  //Free the registry
  for (int i = 0; i < MAXQUEUES; i++) {
    freeTEQ(registry[i]);
  }
  return EXIT_SUCCESS;
}
```

This is my end of main function. I wanted the clean thread to stop working when other threads are gone, then main quits as follows.

I am waiting for the feedback on canvas after this gets graded. I very enjoyed coding this project earlier, but then I realized that This is a big size program, and lost thread of my understanding at the end. I think this project would be easier to understand and start with, if it asks us to do part 3 and 4 then 1 and 2. I was stuck at part 1 for a long time, but after finishing 3 I started understanding more and built up the code easily. I definitely did my best, I hope I did the rest of parts correctly at least, because I am not so sure about other parts besides getEntry and part 5 (very hard to track the connection).