CSE344 – System Programming - Homework 4

Initiation to POSIX threads

Ozan GEÇKİN

1801042103

1) Problem

Our project, You will simulate a system where one student hires other students to do her homework in her place. Every actor in this system will be represented by a separate thread. Thread h, reads her homeworks one by one from an input file, and puts them in a queue, along with her priority (speed S, cost C or quality Q). There will also be an arbitrary number of students-for-hire, each of course represented as a separate thread. Each of them will possess 3 parameters: a price per homework, speed per homework and quality per homework. The main thread, depending on the requested priority, will give the task to the student-for-hire thread that is most suitable and available (i.e. not busy with another homework). My biggest problem here is synchronization. Because of the rule in the assignment, "you are not allowed to use mutexes or condition variables for synchronization. Anything else is fair game. I recommend avoiding signals." I was not allowed to use mutex, condition variables, additional files, even not signal so I decided to use semaphores to ensure synchronization.

2) How to solve issues and algorithm in this Homework ?

At first command line arguments are validated, if one of them are missing, program will display usage information and exit. After this exit handler is registered with sigaction() to catch and handle SIGINT (ctrl+c) signal and terminate working gracefully.

The next step is students file loading, program opens and reads whole file into buffer with open() and read() syscalls. After loading the buffer is scanned detect line amount by searching and counting new line character (‘\n’) in the buffer. With line count we can determine student count, initialize and run student threads with exact amount of students.

After running student threads, H thread context is initialized and program runs H thread. H thread work is very simple, it just reads one symbol from homework file and puts it back to the queue. The semaphore is used to ensure synchronization between main and g thread while accessing the same queue. Then there is no more homework in the file, H thread will exit from while loop and terminate working.

How student thread works:

Students structure (StudentThreadCtx) contains 12 variables:
pthread_t threadId; // Students thread id
sem_t semaphore; // semaphore for synchronization
sem_t *pSemNotify; // semaphore to notify main about availability
int nTerminated; // thread is terminated or not
char sName[NAME_MAX]; // name of the student
int nQuality; // quality value of this student
int nSpeed; // speed value of this student
int nCost; // cost value of this student
char cHomework; // homework this student is hired for
int nSolved; // homeworks solved by this student
int nEarned; // money earned by this student
int nBusy; // flag to check this student is currently busy or not This structure is passed to every student thread.

After running, student threads are waiting for main process for homework assignment. Wait is done with semaphore, when main process assigns homework for student it will unlock student semaphore to let him do the given homework. Student thread job is pretty simple, it just waits to the homework, then sleeps x seconds (where x = 6 – student speed) to simulate homework solving and increases earned money and solved homework counters.

After waking from sleep student will notify main process that he is ready for another homework with increasing value on the main process semaphore.

Main thread:

After running student and H threads, main thread is releasing semaphore to let H thread bring another homework, semaphore is used to ensure sync between main and H thread while accessing the same queue. Main thread will take first homework from queue and find relevant student by priority with findRelevantStudent() function. In this function every student context is iterated and students with busy flag are skipped to avoid homework assign to the busy students. Then appropriate student is selected according to the homework.

For example if homework is Q function selects available student with highest quality value. If all students are busy in the given moment, NULL pointer is returned by the function. If all students are busy, main thread will wait until one of them gets available. This is done with waiting on mainSemaphore. This semaphore will be unlocked by the student when he gets available so main will wake continue working. After finding relevant student, main process will assign homework to this student and release selected students semaphore to wake student and let him solve homework. If there is no enough available money or no more homework in queue, main process will break from while loop, terminate all threads and deallocate all resources, display statistics and finish working.

3) Testing and Running

```
cse312@ubuntu:~/Desktop/systemhw4/thread_simulator$ make
gcc -g -O2 -Wall program.c -o program -lpthread
cse312@ubuntu:~/Desktop/systemhw4/thread_simulator$ ./program homework students 10000
odtulu is waiting for a homework
bogazicili is waiting for a homework
itulu is waiting for a homework
ytulu is waiting for a homework
5 students-for-hire threads have been created.
Name Q S C
odtulu 5 3 900
bogazicili 4 5 1000
itulu 4 4 800
ytulu 3 4 650
sakaryali 1 2 150
sakaryali is waiting for a homework
H has a new homework C; remaining money is 10000TL
H has a new homework S; remaining money is 9850TL
sakaryali is solving homework C for 150, H has 9850TL left.
bogazicili is solving homework S for 1000, H has 8850TL left.
H has a new homework Q; remaining money is 8850TL
odtulu is solving homework Q for 900, H has 7950TL left.
H has a new homework C; remaining money is 7950TL
H has a new homework S; remaining money is 7300TL
ytulu is solving homework C for 650, H has 7300TL left.
itulu is solving homework S for 800, H has 6500TL left.
H has a new homework Q; remaining money is 6500TL
bogazicili is waiting for a homework
bogazicili is solving homework Q for 1000, H has 5500TL left.
H has a new homework C; remaining money is 5500TL
itulu is waiting for a homework
ytulu is waiting for a homework
H has a new homework S; remaining money is 4850TL
ytulu is solving homework C for 650, H has 4850TL left.
itulu is solving homework S for 800, H has 4050TL left.
H has a new homework Q; remaining money is 4050TL
bogazicili is waiting for a homework
bogazicili is solving homework Q for 1000, H has 3050TL left.
H has a new homework C; remaining money is 3050TL
odtulu is waiting for a homework
bogazicili is waiting for a homework
odtulu is solving homework C for 900, H has 2150TL left.
H has a new homework S; remaining money is 2150TL
bogazicili is solving homework S for 1000, H has 1150TL left.
H has a new homework C; remaining money is 1150TL
sakaryali is waiting for a homework
H has a new homework Q; remaining money is 1000TL
sakaryali is solving homework Q for 150, H has 850TL left.
H has a new homework S; remaining money is 850TL
ytulu is waiting for a homework
bogazicili is waiting for a homework
itulu is waiting for a homework
itulu is solving homework S for 800, H has 50TL left.
H has a new homework C; remaining money is 50TL
Money is over, closing.
H has no more money for homeworks, terminating.
odtulu is waiting for a homework
itulu is waiting for a homework
sakaryali is waiting for a homework
Homeworks solved and money made by the students:
odtulu 2 1800
bogazicili 4 4000
itulu 3 2400
ytulu 2 1300
sakaryali 2 300
cse312@ubuntu:~/Desktop/systemhw4/thread_simulator$
```