Comp 304 — Project 2

Gul Sena Altintas*& İdil Defne Çekin †

June 4, 2021

Contents

1. Overview

Every part of our project works properly.
Note: You may directly run and test our code by running `make run`, which will compile the source file and run the program with default parameters.

Overall, in the project we used 2 mutexes and 3 conditional mutexes and an array of conditional mutexes answer:

1. mutex: to lock the queue and moderate between the questions and answers.
2. breaking_mutex: for when the breaking event occurs.
3. question_asked: to broadcast that the question is asked (by the moderator).
4. next: to signal the next commentator in the queue to start talking.
5. breaking: to broadcast that the breaking event is over (by the main thread). The commentator cut short waits for this signal to signal *next*.
6. answer: one conditional mutex for each speaker, $answer[i]$ is signaled by the moderator when its the ith commentator's turn to speak.

*galtintas17@ku.edu.tr - 64284
†icekin17@ku.edu.tr - 64387

We have 3 types of threads:

1. Main_thread: Generates breaking event.
2. Moderator_thread: Asks questions and signals to commentators when it is their turn to speak.
3. Commentator_thread: Adds themselves to the queue if they want to answer, speaks when signaled by the moderator, and stops speaking if a breaking event occurs. Signals *next* when they are done talking. Increases *count* variable after they speak or decide not to speak for the current question for moderator to be able to wait for every commentator.

2. Utils

2.2.1. Parameter Parsing

We used `getopt` function to parse and validate command line parameters. Corresponding function and variables are defined in `src/utils.c`.

2.2.2. Queue

To manage the commentator queue, we implemented a basic Queue struct and utility functions in `src/queue.c`.

2.2.3. Time & Logging

To be able to log the time since the start, we defined `void get_time()` in `src/utils.c` which overrides the variable `log_time` with the difference between now and start_time.

3. Moderator

In each iteration of the while loop;

- Moderator locks the mutex.
- Moderator asks a question if:
  - The queue is empty.
  - There are still questions to ask.
  - count - indicating the number of speakers that has decided to answer or not answer the previous question - is equal to n. (initialized as n so if it is the first question this still works)
- After asking the question, moderator broadcasts question_asked to alert the speakers, and sets the count to 0.
- Moderator ends the session if:
  - All questions are asked and,
  - Queue is empty and,
  - count is equal to n - so that everyone has finished speaking.
- Moderator ends the session by leaving the for loop and increasing the current question number so each thread will leave their loop. (Also unlocks the mutex)
- Else:

  - Moderator wakes the first speaker in the queue by signaling their corresponding conditional mutex, in the answer array.
  - And waits for the next signal, which will be signaled after the speaker finishes talking.

- At the end of the loop, Moderator unlocks the mutex.

4. Commentator

In each iteration of the while loop;

- Locks the mutex.
- Waits for the question_asked conditional mutex - which will be broadcast when the moderator asks a question.
- Decides to answer the question with probability p.
  - Enqueues itself using the enqueue method of our custom queue structure.
  - Increments count - explained in the moderator section (Section 3). item Waits for the moderator to signal their conditional mutex meaning it is their turn to speak.
  - Until their speach is over, they check the breaking_event indicator every second to be aware if they need to stop.
  - If there is a breaking event, locks the breaking_mutex and waits for the breaking to be signaled indicating the end of the event, and unlocks the breaking_mutex. Only the currently speaking speaker needs to stop and wait for the event.
  - When the speech is completed - or the breaking event is over, signals next so moderator is waken up, unlocks the mutex.

5. Main

In each iteration of the while loop;

- Sleeps for one second.
- Generates a breaking event with probability b:
  - Sets the boolean indicator, breaking_event to 1 - so speakers will stop.
  - Sleeps for 5 seconds.
  - Ends the breaking event by signaling breaking conditional mutex.
  - Sets the breaking_event to 0 - so speakers won't stop.

6. Sample Run

```
$ time ./project2 -n 4 -p 0.75 -q 5 -t 3 -b 0.05
User chosen parameters are as follows:
  n: 4  p: 0.750000 q: 5
  t: 3  b: 0.050000
Hello, I am Commentator #0, glad to be here.
Hello, I am Commentator #1, glad to be here.
Hello, I am Commentator #2, glad to be here.
Hello, I am Commentator #3, glad to be here.
[0:0.948395]  Moderator asks question 1
[0:0.948565]  Commentator #1 wants to answer, position in queue 0
[0:0.948719]  Commentator #3 wants to answer, position in queue 1
```

```
[0:0.948830]  Commentator #1: I will be speaking for 1.186 seconds.
[0:2.135208]  Commentator #2 wants to answer, position in queue 1
[0:2.135422]  Commentator #3: I will be speaking for 1.038 seconds.
[0:3.173985]  Commentator #2: I will be speaking for 0.641 seconds.
[0:3.815470]  Moderator asks question 2
[0:3.815687]  Commentator #0 wants to answer, position in queue 0
[0:3.815853]  Commentator #1 wants to answer, position in queue 1
[0:3.815974]  Commentator #3 wants to answer, position in queue 2
[0:3.816137]  Commentator #2 wants to answer, position in queue 2
[0:3.816228]  Commentator #0: I will be speaking for 1.779 seconds.
[0:5.595804]  Commentator #1: I will be speaking for 0.610 seconds.
[0:6.206322]  Commentator #3: I will be speaking for 2.315 seconds.
[0:8.522465]  Commentator #2: I will be speaking for 2.006 seconds.
[0:8.949942]  Just in!!
[0:9.522835]  Commentator #2: I stopped 1.006 seconds before my end time due to breaking
    event
[0:13.950244] Wow, at least that's over!
[0:13.950374] Moderator asks question 3
[0:13.950442] Commentator #3 wants to answer, position in queue 0
[0:13.950539] Commentator #1 wants to answer, position in queue 1
[0:13.950626] Commentator #3: I will be speaking for 0.100 seconds.
[0:14.51043]  Commentator #2 wants to answer, position in queue 1
[0:14.51183]  Commentator #0 wants to answer, position in queue 2
[0:14.51309]  Commentator #1: I will be speaking for 2.178 seconds.
[0:16.230169] Commentator #2: I will be speaking for 2.484 seconds.
[0:18.715037] Commentator #0: I will be speaking for 0.125 seconds.
[0:18.840308] Moderator asks question 4
[0:18.840444] Commentator #3 wants to answer, position in queue 0
[0:18.840586] Commentator #1 wants to answer, position in queue 1
[0:18.840741] Commentator #0 wants to answer, position in queue 2
[0:18.840821] Commentator #3: I will be speaking for 0.929 seconds.
[0:19.770064] Commentator #2 wants to answer, position in queue 1
[0:19.770221] Commentator #1: I will be speaking for 2.596 seconds.
[0:22.366755] Commentator #0: I will be speaking for 2.602 seconds.
[0:24.969392] Commentator #2: I will be speaking for 2.940 seconds.
[0:27.910003] Moderator asks question 5
[0:27.910138] Commentator #1 wants to answer, position in queue 0
[0:27.910242] Commentator #1: I will be speaking for 2.552 seconds.
[0:27.952338] Just in!!
[0:28.910421] Commentator #1: I stopped 1.552 seconds before my end time due to breaking
    event
[0:32.952506] Wow, at least that's over!
[0:32.952788] Commentator #2 wants to answer, position in queue 0
[0:32.953048] Commentator #2: I will be speaking for 2.735 seconds.

real   0m35,009s
user   0m0,000s
sys 0m0,016s
```

Listing 1: Sample run