CS39002 Operating Systems Laboratory
Spring 2024

Lab Assignment: 6
Date of submission: 06–Mar –2024

Mutithreaded applications using pthread

The busy doctor Foosycian (MBBS, MD, FRCP) takes a heavy breakfast, and enters his chamber at sharp 09:00am. During his morning session, he attends visitors of three categories.

1. **Patients** come for consultation with the doctor. The doctor needs 5 – 15 minutes of consultation per patient.

2. **Reporters** come to meet the doctor to show the reports of the tests suggested by the doctor in earlier consultations (they have nothing to do with newspapers or news channels). Each reporter takes 1 – 2 minutes of the doctor's time.

3. **Sales representatives** come to the doctor for promoting their medicine and medical equipment. A sales rep can talk for hours, but the doctor restricts each sales-rep visit to 5 – 10 minutes.

Because of time limitations, the doctor can attend to only 25 patients and 3 sales representatives in each morning session (assume that at least these many patients and sales reps come to meet the doctor every morning). He however entertains all reporters so long as he is in his chamber. For the doctor, the priorities are ordered as: **reporter > patient > sales rep**. That is, he attends to a waiting patient only when no reporters are waiting. Moreover, a waiting sales rep can enter the doctor's chamber only when no reporters or patients are waiting. At any point of time, at most one visitor can be in the doctor's chamber. No ongoing meeting with the doctor can be preempted (even if a higher-priority visitor arrives during the meeting). In order to handle the logistics, the doctor has appointed a personal assistant Ms. Barina to perform the following tasks.

- When a new visitor comes, she gives the next token number of the appropriate category to the visitor. The visitor then keeps on waiting outside the doctor's chamber. However, if the quota in the visitor's category is already full, the visitor is not given any token, and must leave.

- When the doctor is done with his current visitor, she sends the next visitor to the doctor's chamber (provided that there are waiting visitor(s)). The choice of the next visitor is governed by the doctor's priority ordering.

- The doctor is done when he has already attended to 25 patients and 3 sales reps, and no reporters are waiting. (Of course, he must have attended all the reporters that arrived so far.) Whenever this happens, the assistant notifies the doctor, and the hungry doctor hurriedly ends the session to go for lunch. Any visitor (of any category) that comes after the session must leave. Notice that before the doctor leaves, he may have to stay idle during some periods (his daily quota is not yet full, and a suitable next visitor is yet to come).

In this assignment, you simulate the working of a morning session of the doctor. Write a single C source file `session.c` for that purpose. The input to this simulation is a text file `arrival.txt` storing the categories (P/R/S for patients/reporters/sales reps), arrival times (integers in minutes relative to 09:00am), and doctor's times they take (integers in the ranges specified above). You should use the pthread API in your simulation. A thread is to be used to stand for each person involved. These threads should work as follows.

The main thread: This is the first thread that runs the `main()` function. This simulates the working of Barina, and is also called the assistant thread. This creates the other threads in due course of time. The working of the main thread consists of the following tasks.

- The assistant thread reads the arrival schedule from the text file `arrival.txt`, and stores the events in a (min) priority queue E ordered with respect to the times when the events happen (ties are broken by the types of the events). An implementation of E using a (min) binary heap is supplied to you as `event.h` and `event.c`. Compile this along with `session.c` to generate the final simulator. The arrival time of a visitor is not the same as the time (s)he goes to the doctor's chamber. The assistant thread inserts events of an additional type (the doctor is done with the current visitor) to E. Maintenance of the event queue E is the chief task and the sole responsibility of the assistant thread.

- The second task of the assistant thread is to keep track of the time. No actual clock needs to be maintained. The events define the time. Nothing relevant happens between two successive events. Whenever a new event e is extracted from the event queue E, the time is updated by the time stored in e. This is, in essence, akin to sweep algorithms in computational geometry.

- The assistant creates the doctor thread at the beginning, and keeps on notifying the doctor thread whenever he is ready to accept the next visitor (or the session ends). This is triggered by a doctor-done-with-current-visitor event.

- Whenever a new visitor arrives, the assistant checks whether that visitor can be served by the doctor. If not (that is, if the quota for the visitor's category is full or the doctor has terminated the session), then nothing special needs to be done (except for a printing that the visitor is denied service). Otherwise, a visitor thread of the appropriate category is created. The thread is not immediately allowed to meet the doctor, and is supposed to wait until its turn comes.

- Whenever the doctor is done with the current visitor, the assistant picks a waiting visitor of the appropriate category (based on the doctor's priorities), and ends the waiting of that visitor thread.

The doctor thread: This thread runs the function `doctor()` which runs a loop. Each iteration of the loop begins by waiting until it is woken up by the assistant. The assistant wakes up the doctor when she is sending a new visitor to the doctor, or when the doctor's session is over for that morning. When woken up, the doctor thread prints an appropriate message, and goes to the next iteration of the loop (or breaks if the end-of-session notification is received).

The visitor threads: There are three types of visitor threads based on the category. Each thread runs the appropriate function `patient()`, `reporter()`, or `salesrep()`. It waits (after creation) until woken up by the assistant. It then prints the time interval during which the visitor will be in the doctor's chamber, and exits.

In essence, this is similar to the solution supplied in the MidSem question paper posted to the course website. There are quite a few major differences though.

- Reporters constitute a new type of visitors not handled by the MidSem question.

- The MidSem solution assumes that various threads (or processes) arrive spontaneously. In your case, your program needs to create the threads at appropriate times. This is done by the assistant thread, which also needs to keep track of the simulated time.

- The MidSem solution works with semaphores. Counting semaphores are not directly supported by pthread. Binary semaphores work like pthread mutexes, but there are quite a few restrictions. First, a mutex can be unlocked only by the thread that locks it. Second, an attempt to unlock an unlocked mutex is an error in pthread. Condition variables can address many of these issues. But then, they have a limitation too. If a signal (or broadcast) is sent on a condition variable and no thread(s) is/are waiting on that variable, that signal (or broadcast) is lost. It is not a good idea to keep on sending signals (or broadcasts) in a loop until it is received by other(s). Pthread barriers may be used to work around this problem. You can implement counting semaphores using pthread mutexes and condition variables, but better think in the pthread way when you are supposed to do so.

- You have the assistant here (this entity was absent in the MidSem question). This is indeed the most critical thread doing the synchronization and sequentialization works.

Other files

A few other files are provided to you to relieve you from the non-essential tasks.

An arrival generator `arrival.c` is supplied. This generates a random sequence of arrivals of patients (at least 25), sales reps (at least 3), and reporters. The program saves the arrival records (category, arrival time, duration of doctor's service), one in each line. A line containing `E` indicates the end of the records. Work on various randomly generated random schedules. That your code works on a specific sample is no guarantee that there are no bugs in your code. Run on several samples before submitting your code.

The functions needed for managing the event queue *E* are already supplied as `event.h` and `event.c`. Use these without any modifications. Note that there are four types of events: P (the arrival of a patient), R (the arrival of a reporter), S (the arrival of a sales rep), and D (the doctor is done with the current visitor). Each event is a structure consisting of three fields: type (P/R/S/D), time (when the event occurs), and duration (for P, R or S, this is the duration of the meeting with the doctor; for D, this field is not relevant). The events are ordered first with respect to time and then with respect to type (with the ordering R < P < S < D).

A makefile is supplied too, so you do not have to enter the compilation and running commands. Here are some of the targets that you can use.

`make visitors`	Compile and run `arrival.c` to generate a random sample `arrival.txt`.
`make run`	Compile and run `session.c` assuming that `arrival.txt` is present in the current directory.
`make all`	First `make visitors` and then `make run`.
`make clean`	The quintessential cleaner.

What you need to submit

Submit the single C source file `session.c`. Do not change or rename the other files supplied to you. We will download you submission, and run with the other files as they are.

Sample

On the left, the input file is printed. The input file is not needed to be sorted in any sequence. This file is supplied to you as `ARRIVAL.txt`. You may copy this to `arrival.txt`, and make initial runs on this. When this works, generate other samples using `make arrival` (or `make all`). On the right, the transcript of the session follows. The lines printed by the doctor thread are marked in red. The lines printed by the visitor threads are marked in blue. All the other lines (the indented ones) are printed by the main (that is, assistant) thread. The printing should be sorted with respect to the time of occurrence. The integer times (minutes relative to 09:00am) are printed as clock times in the 12-hour format.

arrival.txt	Output of session.c
P 200 7 P -15 5 P 218 8 P 107 14 P 77 11 P 71 10 P 206 7 P 126 6 P 292 10 P 113 11 P 265 13 P 329 7 P 287 14 P 62 10 P 75 6 P 204 12 P 115 11 P 238 14 P 216 7 P 84 7 P 86 11 P 195 15 P 26 14 P 137 8 P 117 10 P 248 10 P 271 12 P -24 8 P 50 8 P 220 6 R 326 2 R 172 2 R 194 1 R 3 2 R 203 1 R 203 2 R 136 1 R 304 2 R 165 2 R 64 2 R 161 2 S 105 6 S 257 5 S 211 10 S -3 6 S 199 8 E	[08:36am] Patient 1 arrives [08:45am] Patient 2 arrives [08:57am] Sales representative 1 arrives [09:00am] Doctor has next visitor [09:00am – 09:08am] Patient 1 is in doctor's chamber [09:03am] Reporter 1 arrives [09:08am] Doctor has next visitor [09:08am – 09:10am] Reporter 1 is in doctor's chamber [09:10am] Doctor has next visitor [09:10am – 09:15am] Patient 2 is in doctor's chamber [09:15am] Doctor has next visitor [09:15am – 09:21am] Sales representative 1 is in doctor's chamber [09:26am] Patient 3 arrives [09:26am] Doctor has next visitor [09:26am – 09:40am] Patient 3 is in doctor's chamber [09:50am] Patient 4 arrives [09:50am] Doctor has next visitor [09:50am – 09:58am] Patient 4 is in doctor's chamber [10:02am] Patient 5 arrives [10:02am] Doctor has next visitor [10:02am – 10:12am] Patient 5 is in doctor's chamber [10:04am] Reporter 2 arrives [10:11am] Patient 6 arrives [10:12am] Doctor has next visitor [10:12am – 10:14am] Reporter 2 is in doctor's chamber [10:14am] Doctor has next visitor [10:14am – 10:24am] Patient 6 is in doctor's chamber [10:15am] Patient 7 arrives [10:17am] Patient 8 arrives [10:24am] Patient 9 arrives [10:24am] Doctor has next visitor [10:24am – 10:30am] Patient 7 is in doctor's chamber [10:26am] Patient 10 arrives [10:30am] Doctor has next visitor [10:30am – 10:41am] Patient 8 is in doctor's chamber [10:41am] Doctor has next visitor [10:41am – 10:48am] Patient 9 is in doctor's chamber [10:45am] Sales representative 2 arrives [10:47am] Patient 11 arrives [10:48am] Doctor has next visitor [10:48am – 10:59am] Patient 10 is in doctor's chamber [10:53am] Patient 12 arrives [10:55am] Patient 13 arrives [10:57am] Patient 14 arrives [10:59am] Doctor has next visitor [10:59am – 11:13am] Patient 11 is in doctor's chamber [11:06am] Patient 15 arrives [11:13am] Doctor has next visitor [11:13am – 11:24am] Patient 12 is in doctor's chamber [11:16am] Reporter 3 arrives [11:17am] Patient 16 arrives [11:24am] Doctor has next visitor [11:24am – 11:25am] Reporter 3 is in doctor's chamber [11:25am] Doctor has next visitor [11:25am – 11:36am] Patient 13 is in doctor's chamber [11:36am] Doctor has next visitor [11:36am – 11:46am] Patient 14 is in doctor's chamber [11:41am] Reporter 4 arrives [11:45am] Reporter 5 arrives [11:46am] Doctor has next visitor [11:46am – 11:48am] Reporter 4 is in doctor's chamber [11:48am] Doctor has next visitor [11:48am – 11:50am] Reporter 5 is in doctor's chamber [11:50am] Doctor has next visitor [11:50am – 11:56am] Patient 15 is in doctor's chamber [11:52am] Reporter 6 arrives [11:56am] Doctor has next visitor [11:56am – 11:58am] Reporter 6 is in doctor's chamber [11:58am] Doctor has next visitor [11:58am – 12:06pm] Patient 16 is in doctor's chamber [12:06pm] Doctor has next visitor [12:06pm – 12:12pm] Sales representative 2 is in doctor's chamber

```
          [12:14pm] Reporter 7 arrives
[12:14pm] Doctor has next visitor
[12:14pm – 12:15pm] Reporter 7 is in doctor's chamber
          [12:15pm] Patient 17 arrives
[12:15pm] Doctor has next visitor
[12:15pm – 12:30pm] Patient 17 is in doctor's chamber
          [12:19pm] Sales representative 3 arrives
          [12:20pm] Patient 18 arrives
          [12:23pm] Reporter 8 arrives
          [12:23pm] Reporter 9 arrives
          [12:24pm] Patient 19 arrives
          [12:26pm] Patient 20 arrives
[12:30pm] Doctor has next visitor
[12:30pm – 12:31pm] Reporter 8 is in doctor's chamber
          [12:31pm] Sales representative 4 arrives
          [12:31pm] Sales representative 4 leaves (quota full)
[12:31pm] Doctor has next visitor
[12:31pm – 12:33pm] Reporter 9 is in doctor's chamber
[12:33pm] Doctor has next visitor
[12:33pm – 12:40pm] Patient 18 is in doctor's chamber
          [12:36pm] Patient 21 arrives
          [12:38pm] Patient 22 arrives
          [12:40pm] Patient 23 arrives
[12:40pm] Doctor has next visitor
[12:40pm – 12:52pm] Patient 19 is in doctor's chamber
[12:52pm] Doctor has next visitor
[12:52pm – 12:59pm] Patient 20 is in doctor's chamber
          [12:58pm] Patient 24 arrives
[12:59pm] Doctor has next visitor
[12:59pm – 01:06pm] Patient 21 is in doctor's chamber
[01:06pm] Doctor has next visitor
[01:06pm – 01:14pm] Patient 22 is in doctor's chamber
          [01:08pm] Patient 25 arrives
[01:14pm] Doctor has next visitor
[01:14pm – 01:20pm] Patient 23 is in doctor's chamber
          [01:17pm] Sales representative 5 arrives
          [01:17pm] Sales representative 5 leaves (quota full)
[01:20pm] Doctor has next visitor
[01:20pm – 01:34pm] Patient 24 is in doctor's chamber
          [01:25pm] Patient 26 arrives
          [01:25pm] Patient 26 leaves (quota full)
          [01:31pm] Patient 27 arrives
          [01:31pm] Patient 27 leaves (quota full)
[01:34pm] Doctor has next visitor
[01:34pm – 01:44pm] Patient 25 is in doctor's chamber
[01:44pm] Doctor has next visitor
[01:44pm – 01:52pm] Sales representative 3 is in doctor's chamber
          [01:47pm] Patient 28 arrives
          [01:47pm] Patient 28 leaves (quota full)
          [01:52pm] Patient 29 arrives
          [01:52pm] Patient 29 leaves (quota full)
[01:52pm] Doctor leaves
          [02:04pm] Reporter 10 arrives
          [02:04pm] Reporter 10 leaves (session over)
          [02:26pm] Reporter 11 arrives
          [02:26pm] Reporter 11 leaves (session over)
          [02:29pm] Patient 30 arrives
          [02:29pm] Patient 30 leaves (session over)
```