EECS3221 Winter 2017
Assignment 2

POSIX Threads

Due Date: Tuesday, February 28, 2017, 14:30.

A. <u>Description of the Assignment</u>

A1. You are required to read and fully understand the first 4 chapters, that is, pages 1-129, of the book "Programming with POSIX Threads" by (This book is currently on reserve at Steacie Science Library, Call Number QA 76.76 T55 B88 1997).

A2. You are required to download the program "alarm_mutex.c" and the file "errors.h" and "README" from the directory /cs/course/3221/assign2, and then try to compile and execute this program by following the instructions in the "README" file. (This program is explained in pages 52-58 of the book by David R. Butenhof).

A3. You are required to make the following changes to the program "alarm_mutex.c" to produce a program named "My_Alarm.c".

A3.1. In your program "My_Alarm.c", in addition to creating the alarm_thread, the main thread will also create two additional threads, called "display_thread_1", and "display_thread_2", respectively.

A3.2. In your program "My_Alarm.c", immediately after receiving each alarm request, the main thread will print:
"***Main Thread Received Alarm Request at <time>: <alarm_request>***",
where <time> is the actual time at which the alarm was received (<time> is expressed as the number of seconds from the Unix Epoch Jan 1 1970 00:00); <alarm_request> is the original alarm request, including the number of seconds and the alarm message.

A3.3. In your program "My_Alarm.c", after the alarm_thread has retrieved each alarm request from the alarm_list, instead of waiting for the specified number of n seconds and then printing out the alarm message as done in "alarm_mutex.c", the alarm_thread will do the following:

A3.3 (a) if the alarm expiry time of the alarm request is closer to an ***<u>odd</u>*** integer number of seconds from the Unix Epoch Jan 1 1970 00:00, then the alarm thread will pass on the alarm request to display_thread_1;

A3.3 (b) if the alarm expiry time of the alarm request is closer to an ***<u>even</u>*** integer number of seconds from the Unix Epoch Jan 1 1970 00:00, then the alarm thread will pass on the alarm request to display_thread_2;

(For example, if $0 \leq t < 0.5$ then t will be considered as closer to 0. if $0.5 \leq t \leq 1$ then t will be considered as closer to 1.)

A3.3 (c) Immediately after the alarm_thread has passed on the alarm request to one of the display threads, the alarm_thread will print:
"***Alarm Thread Passed on Alarm Request to Display Thread <number> at <time>: <alarm_request>***",
where <number> is 1 if the alarm thread passed on the alarm request to display_thread_1; <number> is 2 if the alarm thread passed on the alarm request to display_thread_2; <time> is the actual time at which the alarm was received (<time> is expressed as the number of seconds from the Unix Epoch Jan 1 1970 00:00); <alarm_request> is the original alarm request, including the number of seconds and the alarm message.

A3.3 (d) Then the alarm_thread will be free to continue to retrieve and process new alarm requests from the alarm_list.

A3.4. Both display_thread_1 and display_thread_2 will repeatedly check to see whether the alarm thread has passed on an alarm request to itself.

After display_thread_1 or display_thread_2 has received an alarm request from the alarm_thread, the display thread to which the alarm thread has passed on an alarm request will do the following:

A3.4 (a) It will immediately print:
"***Display Thread <number>: Received Alarm Request at <time1>: <alarm_request>, ExpiryTime is <time2>*** ",
where <number> is 1 if display_thread_1 is printing this message; <number> is 2 if display_thread_2 is printing this message; <time1> is the actual time at which the alarm was received; <time2> is the time at which the alarm will expire (both <time1> and <time2> are expressed as the number of seconds from the Unix Epoch Jan 1 1970 00:00); <alarm_request> is the original alarm request, including the number of seconds and the alarm message.

A3.4 (b) It will print the following repeatedly, once every 2 seconds, until the specified number of n seconds has expired:
"***Display Thread <number>: Number of SecondsLeft <nsec>: Time: <time>: <alarm_request>*** ",
where <number> is 1 if display_thread_1 is printing this message; <number> is 2 if display_thread_2 is printing this message; <nsec> is the number of seconds left before the alarm is set to expire; <time> is the actual time at which the alarm was retrieved (<time> is expressed as the number of seconds from the Unix Epoch Jan 1 1970 00:00); <alarm_request> is the original alarm request, including the number of seconds and the alarm message.

A3.4 (c) After the specified number of n seconds has expired, it will print:
"***Display Thread <number>: Alarm Expired at <time>: <alarm_request>*** ",
where <number> is 1 if display_thread_1 is printing this message; <number> is 2 if display_thread_2 is printing this message; <time> is the actual time at which this was

printed (<time> is expressed as the number of seconds from the Unix Epoch Jan 1 1970 00:00); <alarm_request> is the original alarm request, including the number of seconds and the alarm message.

B. <u>Platform on Which The Programs Are to be Implemented</u>

The programs should to be implemented using the ANSI C programming language and using the Prism Linux system at York. You should use POSIX system calls or POSIX functions whenever possible.

C. <u>Additional Requirements</u>

(a) You must make sure that your code has very detailed comments.

(b) You must make sure that your code compiles correctly with your Make file.

(c) You must make sure that your code does not generate segmentation faults.

(d) You must make sure that your code is able to handle incorrect input.

(e) You must describe in detail any problems or difficulties that you had encountered, and how you solved or were able to overcome those problems or difficulties in the report.

Failure to satisfy the additional requirements above will result in a very low mark for the assignment.

D. <u>What to Hand In</u>

Each group is required to hand in both a hard copy and an electronic copy of the following:

1. A written report that identifies and addresses all the important aspects and issues in the design and implementation of the programs for the problem described above.

2. The C source programs.

3. A "Test_output" file containing the output of any testing your group has done.

4. A "makefile" file to make it easier for the marker to compile and run your group's program.

5. A "README" file explaining how to compile and run your group's program.

Each group is required to use the utility "submit" to submit the electronic version of the above 5 files plus the “errors.h” file to the course directory /cs/course/3221/submit/a2

(The file “errors.h” should be included among the files submitted so that the marker can test whether your group’s programs can compile and run correctly or not.)

E. Evaluation of the Assignment

1. The report part of your assignment (50%) will be evaluated according to:

(a) Whether all important design and implementation aspects and issues of your programs related to the problem above have been identified and appropriately addressed.

(b) How well you have justified your design decisions.

(c) The quality of your design.

(d) How well you have designed and explained the testing.

(e) The clarity, and readability of the report.

2. The program and testing part of your assignment (50%) will be evaluated according to:

(a) The quality of the design and implementation of your programs.

(b) The quality of the testing of your programs.

(c) Whether your programs satisfy the Additional Requirements in section C above.

F. Notes

Please note that the requirements specified in section A. Description of the Assignment above, are the *minimum requirements* that must be satisfied by your program. Obviously, there are many other possible details of the alarm system that have been left unspecified. It is your responsibility to make appropriate design and implementation choices concerning the unspecified details of the alarm system, and justify those decisions in your report.