EECS 3221

Assignment 2 Report

Group Member

Yiming Shao	215159932
Yixi Zhao	214936298
Youngin Ko	214876924
Weinan Wu	212662011
Guohao Ouyang	214447817

Design and implementation

- Two types of "Alarm requests"

  "Start_Alarm" is basically borrowed from the "alarm_mutex.c" program, the only modification we have done is we modified the input part (sscanf) and attributes of an alarm (alarm_t), "alarm_id" is added to "alarm_t" as a new attribute.

  "Change_Alarm" has the similar input as "Start_Alarm", the difference is its functionality, the alarm that we would like to change will be found by its id by looping the alarm list, modify the alarm use the new inputs if the alarm is found.

- The main_thread

  Our program follows the instruction, all inputs will be checked before use, only pass the valid command into the program, otherwise, print "Bad command". The new alarm will be printed on the command line interface with exactly same format as the instruction mentioned. Alarms will be ordered by their id, this step is done by the alarm_insert function, and this function is called when the new alarm is valid to use.

- The alarm_thread

  A new thread called display_thread is newly implemented, display_thread is a conditional variable associated with a mutex, and it can tell the state of shared memory. There are totally up to three display_threads exist at the same time, we decide we need to create a new display_thread or insert the alarm into the exist thread by using the information stored in display_info (which contains the capacities of three display threads and how many of them are in use).

When the alarm is expired , the command line will print the information that it is removed, count_1, count_2, count_3 indicate the number of alarms associated on display_thread 1 to 3, when the alarm is expired, the count will minus 1, if the count equals 0, means there is no more alarms in the display, then that display_thread will be terminated.

- The display_alarm_thread

  The current_alarm is the alarm that is being selected at the time, the display_alarm_thread which the current_alarm associated with will print its state periodically every 5 seconds (We use sleep function to achieve this)

- The display_change_thread

  The change_alarm is a temporary alarm to save the changing information, and pass those value to Display_Changing thread, and then print this thread's information.

Testing

Test_input: Start_Alarm(1) 40 aaaaaaaa

Start_Alarm(2) 20 aaaaaaa

Start_Alarm(3) 20 aaaaaaa

Start_Alarm(4) 20 aaaaaaa

Start_Alarm(5) 20 aaaaaaa

Change_Alarm(4) 5 AAAAA

Change_Alarm(5) 5 AAAAA

Change_Alarm(6) 5 AAAAAA

In our testing, first input is Start_Alarm(1) 40 aaaaaaaa, alarm_thread will assign this alarm to display_thread 1, and it will start automatically print per 5 sec. After that, when we type second input, the first input is still working, alarm thread will assign second input to display_thread 2. When we type third input, the first input was still working and second was waiting. Input 3 would be assigned to display_thread3. After it, the fourth input and fifth input will be assign to wait list until first input finished. At this time, when we type change command, main thread will searching the alarm from Alarm_list by Alarm id. If alarm is found, it will go to the display_change_thread and print the changing information. If alarm is not found, it will print not No such alarm.

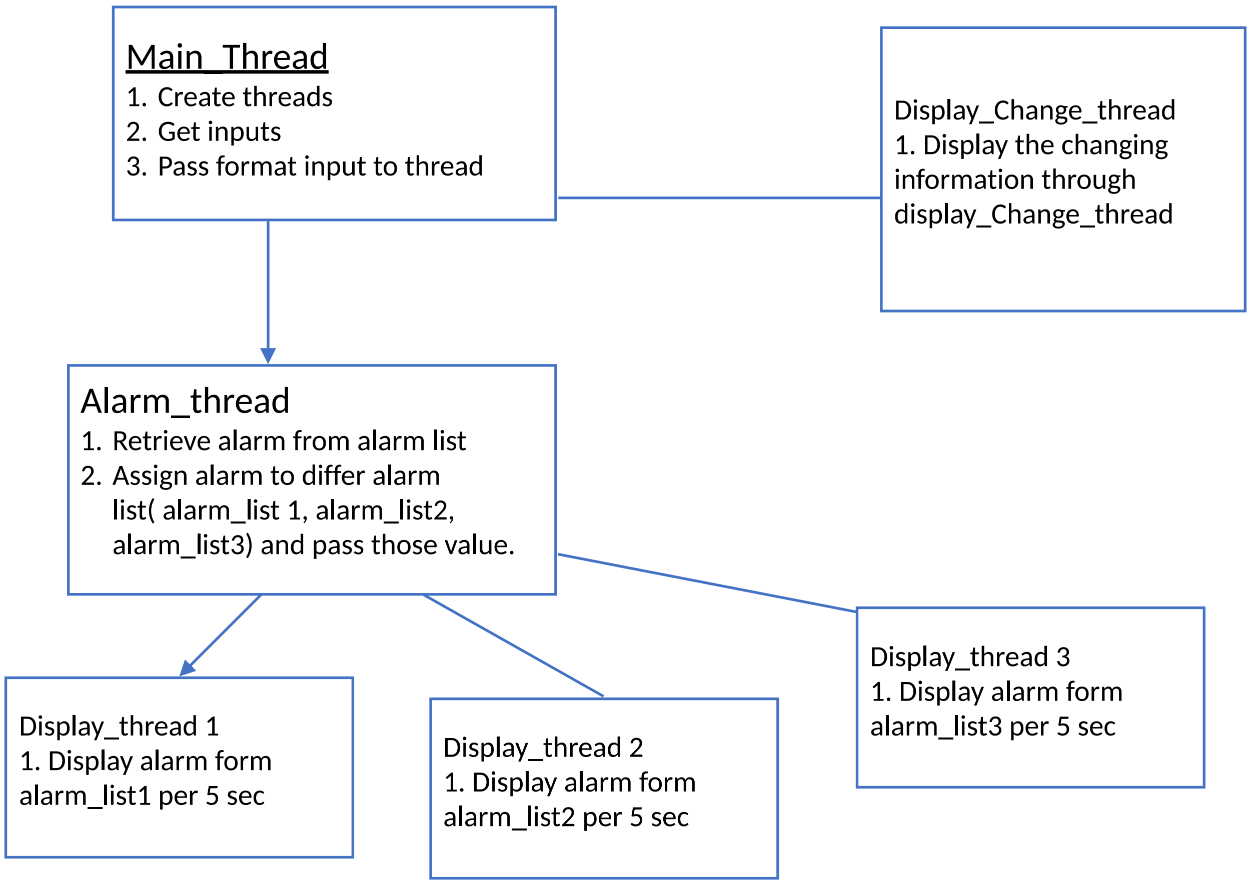
Design decision

In the design, the most significant change we made is the implementation of conditional variable, so that we can make our display_thread comes true, the conditional variable can track the status of the current alarm so that we can use the information to print the status of alarms in the command line.

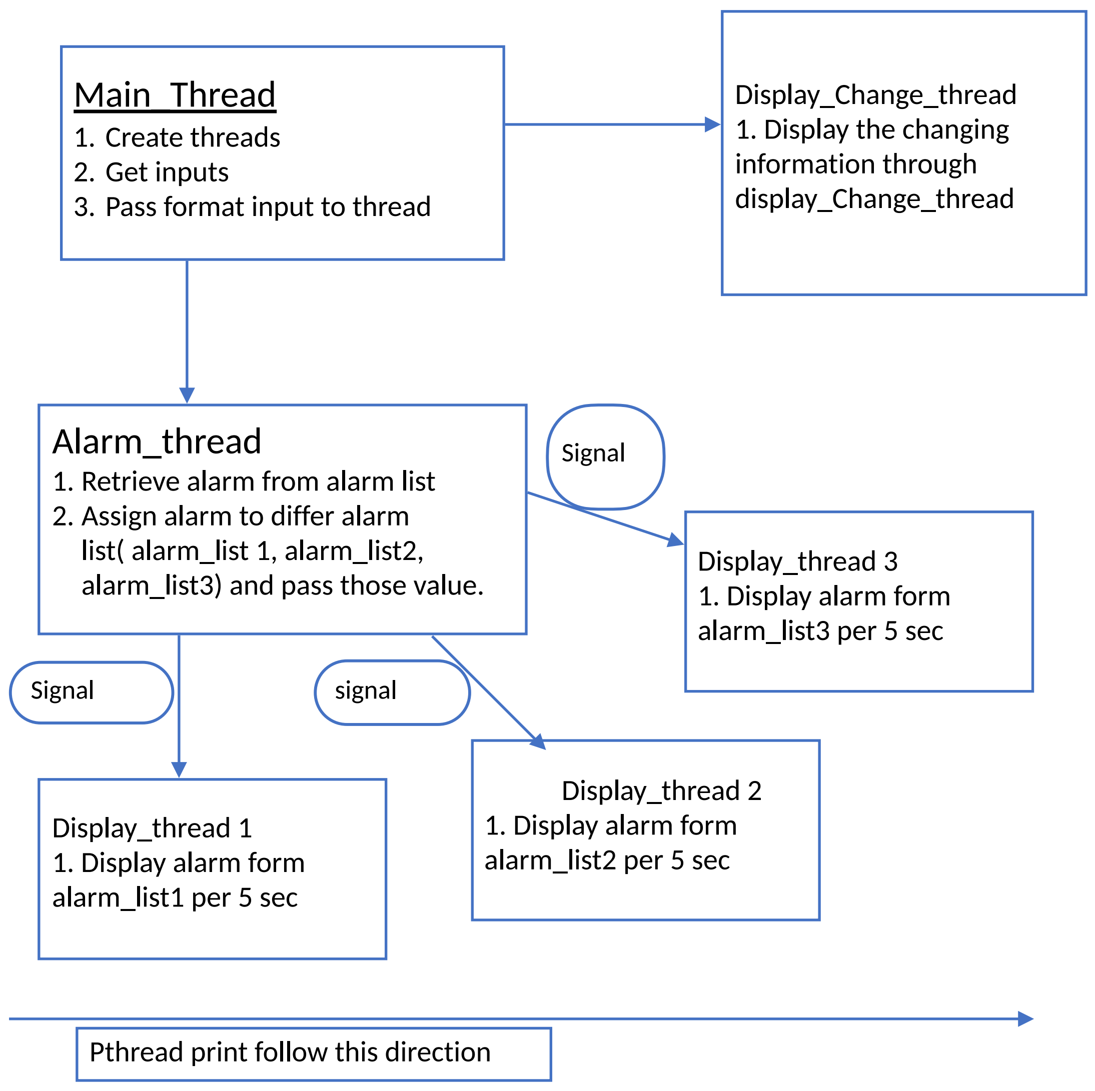
When the first alarm is inserted into the alarm list, it is associated with an alarm_display_thread immediately, then the alarm will sleep, so we don’t get the chance to modify that alarm.

Design Diagram:

To begin with, we try to design like this, however, because we cannot solve segmental fail. We change to plan two.


Main_Thread
1. Create threads
2. Get inputs
3. Pass format input to thread
Display_Change_thread
1. Display the changing information through display_Change_thread
Alarm_thread
1. Retrieve alarm from alarm list
2. Assign alarm to differ alarm list(alarm_list 1, alarm_list2, alarm_list3) and pass those value.
Display_thread 1
1. Display alarm form alarm_list1 per 5 sec
Display_thread 2
1. Display alarm form alarm_list2 per 5 sec
Display_thread 3
1. Display alarm form alarm_list3 per 5 sec

Because we cannot solve the Synchronization problem by mutex lock. We decided to use condition signal to control thread.


Main_Thread
1. Create threads
2. Get inputs
3. Pass format input to thread
Display_Change_thread
1. Display the changing information through display_Change_thread
Alarm_thread
1. Retrieve alarm from alarm list
2. Assign alarm to differ alarm list(alarm_list 1, alarm_list2, alarm_list3) and pass those value.
Signal
Display_thread 3
1. Display alarm form alarm_list3 per 5 sec
Signal
signal
Display_thread 1
1. Display alarm form alarm_list1 per 5 sec
Display_thread 2
1. Display alarm form alarm_list2 per 5 sec
Pthread print follow this direction

Design issues

First issues is to get the format input. We try to assign first part including command and left bracket, second part including Alarm_id, third part is right bracket, fourth part is time, and fifth part is message. However, we didn't make it. After we print the input after formatting, it didn't print as we designed. We found sscanf() method can detect bracket. Therefore, we designed stander input like Command + (integer) + integer + String

Second issues is that we don’t know how to pass the value from alarm_thread to three display_ thread. We set a three condition signal to awake three display thread. When alarm_thread assign a new alarm to a display_thread, alarm_thread will use those thread condition signal to awake that thread, and then print information per 5 sec.

Third issues is that when we change the information. We cannot use alarm_thread to pass value to display_thread to print. Then, we construct a new struct display_info to store display information including each display_thread capacity, number of wake_up_thread, and number of alarm in each display_thread.

Fourth issues: We search the man pthread_create(), we found we can pass value through pthread_create method like pthread_create(&thread, NULL, alarm_thread, (void *) alarm_thread), however, we got trouble on it. Sometimes mute lock cannot lock the thread properly as we designed. To avoid segmental fault, we add condition signal to control display_thread working.