CS6868: Concurrent Programming
Spring 2014
Assignment 1: Due 22 March 2014, 11:55 pm

Abil N George (CS13M002)
Dennis Antony Varkey (CS14S008)

Problem 2:Zombie Invasion

Problem Statement

You and 4 of your friends are cleaning out a zombie invasion. You've secured a warehouse consisting of just one very large room and 4 doors to the street. Zombies move slowly, so it's easy to control in general. You only have one weapon though, so you've set up the following protocol.Each of your 4 friends controls each of the 4 doors; they let in individual zombies, keeping count of how many have entered. You stand in the center, and eliminate the zombies that have entered as fast as you can, keeping track of how many you have removed. You don't want too many zombies in the room for obvious reasons, and so must periodically check on how many zombies are in the building in total. If there are too many you need to ensure no new zombies enter until you've had opportunity to reduce their numbers. For this you can only radio each of your friends individually and find out how many they have let in and/or ask them to close the door. Only once the total number is below a reasonable threshold should you allow zombies back in, again only by radioing each friend individually.

Simulate this as a multi-threaded program. You should have 5 threads, one representing you and one for each friend/door. Each friend thread should let in a zombie with a 10% chance every 10ms, keeping track of the number she admitted. The thread representing you has a 40% probability of removing a zombie once every 10ms. You should check the total every 2s, and if it is below n then everything is ok, otherwise no new zombies should be admitted until you've reduced the number to below n/2. n is a command-line parameter. You should use pthreads to solve this problem. Your solution should allow for maximal concurrency—operations should not be serialized unless necessary. Run your program for a few minutes with n = 5, n = 10, n = 100. What is your throughput (zombies eliminated/second)?

Approach

We can use global count for keep track of total zombies in room.Access to count should be protected by mutex.There should be 4 door boolean flag representing state of each door. Each Door thread check door flag if true (Door locked) it wait for signal to open door.Else it increment count of zombies in room by 1 (with probability 10%)

The armed thread will decrement count and in every 2s it check count against threshold (provided door is true;open) if above set door flags to false (which cause door to wait).If door is already closed and count is below threshold/2 thread send signal to door threads.

Shared Variables:

int ***count*** - protected by mutex ***count_lock***
boolean ***Door[4]*** - protected by mutex ***door_lock[4]***
conditionVariable ***doorOpen[4]***

Door Thread

```
while(true)
        if Door[threadId] is false
                wait doorOpen[threadId] signal
        else
                increment count with probability 10%
        sleep for 10ms
end
```

Armed Thread

```
while(true)
        decrement count with probability 40%
        if Door [i] is false for all i and count < threshold/2
                Door[i] = true for all i
                send doorOpen[i] signal for all i
         if Door [i] is true for all i and last check is before 2s
                if count > threshold
                        Door[i] = false for all i

        sleep for 10ms
end
```

Throughput

Threshold	*No of Zombies Eliminated (1 minute)*	*Throughput* $\left(\frac{\text{No of Zombies Eliminated}}{\text{Time in sec}}\right)$
5	2284	38.066666
10	2359	39.316666
25	2372	39.533333
50	2341	39.016666
75	2385	39.750000
100	2363	39.383335

Sample Output

Output of Program with threshold 100:

[Main] : Armed Thread created successfully
[Main] : Door(0) Thread created successfully
[Main] : Door(1) Thread created successfully
[Main] : Door(2) Thread created successfully
[Main] : Door(3) Thread created successfully
[10]: No zombie to kill
[30]: No zombie to kill
[50]: No zombie to kill
[100]: Killing zombie (1 -> 0)
[120]: Killing zombie (1 -> 0)
[200]: Killing zombie (3 -> 2)
….
….
[420]: Killing zombie (1 -> 0)
[440]: No zombie to kill
[470]: Killing zombie (2 -> 1)
….
….
[1990]: Killing zombie (4 -> 3)
[2000]: Killing zombie (3 -> 2)
Checking is count above threshold ?..
[70]: Killing zombie (3 -> 2)
[90]: Killing zombie (2 -> 1)
…

…
[1830]: Killing zombie (123 -> 122)
[1840]: Killing zombie (122 -> 121)
[1890]: Killing zombie (124 -> 123)
[1970]: Killing zombie (129 -> 128)
[1980]: Killing zombie (128 -> 127)
Checking is count above threshold ?..
Sending signal to close Door 0
Sending signal to close Door 1
Sending signal to close Door 2
Sending signal to close Door 3
Closing Door 1..
Waiting For signal to open Door 1..
Closing Door 0..
Waiting For signal to open Door 0..
Closing Door 2..
Waiting For signal to open Door 2..
Closing Door 3..
Waiting For signal to open Door 3..
[20]: Killing zombie (128 -> 127)
[30]: Killing zombie (127 -> 126)
[40]: Killing zombie (126 -> 125)
[50]: Killing zombie (125 -> 124)
[70]: Killing zombie (124 -> 123)
[90]: Killing zombie (123 -> 122)
[150]: Killing zombie (122 -> 121)
….
….
[1990]: Killing zombie (52 -> 51)
[2000]: Killing zombie (51 -> 50)
count below threshold/2
Sending signal to open Door 0
Sending signal to open Door 1
Sending signal to open Door 2
Sending signal to open Door 3
Opening Door 2...
Opening Door 0...
Opening Door 3...
Opening Door 1...
[50]: Killing zombie (53 -> 52)
[90]: Killing zombie (54 -> 53)
[130]: Killing zombie (58 -> 57)
….
….
[MAIN]:Sending Cancel Signal...
[Armed Thread] : Received Cancel Signal.
[Armed Thread] : Canceling Thread..
[Door 3] : Received Cancel Signal.
[Door 3] : Canceling Thread..
[Door 1] : Received Cancel Signal.
[Door 1] : Canceling Thread..

[Door 0] : Received Cancel Signal.
[Door 0] : Canceling Thread..
[Door 2] : Received Cancel Signal.
[Door 2] : Canceling Thread..
Total Zombies Eliminated = 2363
Zombies Eliminated per sec = 39.383335