Assessment 2 component: Door controller

Summary

Each door controller is responsible for a single secured door. The doors are operated remotely and may receive TCP messages requesting that they open or close - typically from the overseer, but in an emergency, from the fire alarm unit. Doors have four states: open, closed, and in the process of opening or closing.

Doors can come in two security configurations: fail-safe and fail-secure. Fail-safe doors will be instructed to open by the fire alarm unit during an emergency situation. During a security breach, fail-secure doors will be instructed to permanently close (until the building's systems are reset--this event is outside the scope of this assessment.)

Safety-critical component

As the doors may need to be opened in an emergency by the fire alarm unit, the door controller is considered a **safety-critical component** and therefore must be implemented following appropriate safety-critical software standards. Make sure to include a **block comment** near the top of your program's source code documenting and justifying any deviations or exceptions.

Program name

```
door
```

Command-line arguments

```
{id} {address:port} {FAIL_SAFE | FAIL_SECURE} {shared memory path} {shared memory offset} {overseer ad
dress:port}
```

Shared memory structure

```
struct {
    char status; // 'O' for open, 'C' for closed, 'o' for opening, 'c' for closing
    pthread_mutex_t mutex;
    pthread_cond_t cond_start;
    pthread_cond_t cond_end;
};
```

Initialisation

On startup, this component will bind the TCP port it was supplied with and start listening on it. It will then send the following initialisation message to the overseer via TCP if the door is in a fail-safe configuration:

`DOOR {id} {address:port} FAIL_SAFE#`

And the following message if the door is fail-secure:

`DOOR {id} {address:port} FAIL_SECURE#`

The default door status on initialisation will be C.

Normal operation

After initialisation, the component will perform the following loop:

1. Lock the mutex, retrieve the current door status, then unlock it
2. Accept the next TCP connection
3. If the message received is OPEN# and the door status is O, respond with ALREADY#, close the connection and go back to 2
4. If the message received is CLOSE# and the door status is C, respond with ALREADY#, close the connection and go back to 2
5. If the message received is OPEN# and the door status is C
   - A. Respond with OPENING#
   - B. Lock the mutex
   - C. Set the door status to o
   - D. Signal cond_start
   - E. Wait on cond_end

   F. Unlock the mutex
   G. Respond with OPENED#, close the connection
6. If the message received is CLOSE# and the door status is O
   A. Respond with CLOSING#
   B. Lock the mutex
   C. Set the door status to c
   D. Signal cond_start
   E. Wait on cond_end
   F. Unlock the mutex
   G. Respond with CLOSED#, close the connection
7. If the message received is OPEN_EMERG# and the door status is C, complete the sub-steps in 4 to open the door (except for sending responses to the fire alarm unit), but from then on do not close the door again; respond to all further CLOSE# requests with EMERGENCY_MODE# and close the connection. If the door status is O, do the same but skip the sub-steps in 4 as the door is already open.
8. If the message received is CLOSE_SECURE# and the door status is O, complete the sub-steps in 5 to close the door (except for sending responses to the overseer), but from then on do not open the door again; respond to all further OPEN# with SECURE_MODE# and close the connection. If the door status is C, do the same but skip the sub-steps in 5 as the door is already closed.
9. Loop back to 1

Example operation

The program might be executed from the command-line with the following:

```
./door 101 127.0.0.1:4000 FAIL_SAFE /shm 200 127.0.0.1:3000
```

The program will shm_open shared memory segment at /shm with an offset of 200 and the size of the struct defined above, and mmap it into memory.

It will then open up a TCP connection to 127.0.0.1 on port 3000 and send the following text:

```
DOOR 101 127.0.0.1:4000 FAIL_SAFE#
```

It will then immediately close the connection, and then proceed to the 'Normal operation' stage. As this is a fail-safe door, it may receive OPEN# or CLOSE# messages from the overseer, and it may receive OPEN_EMERG# messages from the fire alarm unit, but it will not receive CLOSE_SECURE# messages.