Assessment 2 component: Card reader controller

Summary

Each card reader controller is responsible for a single card reader, usually associated with a secure door. Users come up to the card readers and scan their cards/badges - this causes the card reader to scan the 16 character sequence encoded into that user's card, which is then sent to the overseer via TCP. The card reader is not responsible for checking authorisation or for operating doors - these tasks are left to the overseer.

Program name

```
cardreader
```

Command-line arguments

```
{id} {wait time (in microseconds)} {shared memory path} {shared memory offset} {overseer address:port}
```

(Ignore the 'wait time' parameter - it will be supplied to your program but you do not need to use it.)

Shared memory structure

```
struct {
    char scanned[16];
    pthread_mutex_t mutex;
    pthread_cond_t scanned_cond;

    char response; // 'Y' or 'N' (or '\0' at first)
    pthread_cond_t response_cond;
};
```

Initialisation

On startup, this component will send the following initialisation message to the overseer via TCP:

`CARDREADER {id} HELLO#`

Normal operation

After initialisation, the component will perform the following loop:

1. Lock the mutex
2. Look at the scanned code. If the bytes are all `\0` (NUL) nothing has been scanned yet- skip to 7
3. Open a TCP connection to the overseer
4. Send the following data: `CARDREADER {id} SCANNED {scanned}#`
5. If the response was `ALLOWED#`, set the 'response' char to `Y`. Anything else, or if the overseer timed out or the connection failed, set the response char to `N`
6. Signal 'response_cond'
7. Wait on 'scanned_cond'
8. Loop back to 2

Example operation

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

```
./cardreader 101 /shm 100 127.0.0.1:3000
```

The program will shm_open shared memory segment at /shm with an offset of 100 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:

`CARDREADER 101 HELLO#`

It will then immediately close the connection, and then proceed to the 'Normal operation' stage. If, for example, the 'scanned' array contains the following bytes: `0b9adf9c81fb959`, it will send this TCP message to 127.0.0.1:3000:

`CARDREADER 101 SCANNED 0b9adf9c81fb959#`

Then, depending on the response received, it will either write a Y or N into the 'response' char of the shared memory structure.