INTRODUZIONE

Qualsiasi comunicazione all'interno avviene tramite socket TCP in quanto l'applicazione sviluppata è loss sensitive (non si vuole per esempio ricevere comandi errati), il formato dei messaggi dipende strettamente dal comando che si sta eseguendo, sia esso da parte della lavagna o da parte di un utente, non c'è nessun invio della lunghezza precedente alla comunicazione, questo per il motivo di mantenere minima l'interazione con la rete ai fini di efficienza, Si va a definire quindi il formato dei byte da inviare prima di ciascuna comunicazione, questo è vero tranne in una situazione ovvero l'invio della lista delle porte, in questo caso si procede prima con l'invio della lunghezza causa l'alta variabilità di quest'ultima.
Le strutture dati sono definite nel file strutture.h esso contiene, oltre alle strutture descrittive di lavagna utenti e card, la definizione di costanti rappresentative di comandi e dimensione dei buffer. Per la compilazione si può utilizzare il makefile all'interno del progetto.

LAVAGNA

I file relativi alla lavagna sono contenuti interamente nella cartella lavagna a sua volta contenuta nella cartella src. All'avvio oltre a inizializzare le strutture dati si procede con la creazione un thread che servirà ad accettare da linea di comando i comandi di: SHOW_LAVAGNA per stampare la lavagna a video e QUIT per terminare correttamente il processo. La lavagna memorizza utenti e le varie colonne come linked list poiché entrambi sono di estrema variabilità e prevedere un numero massimo per ciascuno di essi porterebbe inevitabilmente a uno spreco di memoria o una limitata scalabilità in caso il numero di utenti e conseguentemente delle card sia particolarmente elevato.
Per la comunicazione la lavagna utilizza unicamente multithreading per motivi di estrema variabilità dei comandi possibili e di un conseguente aumento di complessità nella gestione di ogni stato possibile della comunicazione se si usasse IO multiplexing, la gestione della concorrenza tramite semafori non costituisce un grande intralcio in quanto le operazioni eseguite sulla lavagna sono di relativa semplicità (trattasi perlopiù di ricerche/ rimozioni da liste), inoltre ai fini di ridurre al minimo l'overhead dovuto alla creazione di thread, alla registrazione ad un utente vengono associati 2 thread che rimarranno attivi fino alla disconnessione dello stesso. Il primo dei due thread viene creato immediatamente dopo l'accettazione della connessione da parte della lavagna, esso serve unicamente ad accettare e gestire i comandi inviati da parte dell'utente trai quali: HELLO, CREATE_CARD, QUIT, ACK_CARD e CARD_DONE; il secondo thread che viene creato durante la HELLO avrà il compito duale di inviare i messaggi all'utente, in particolare SEND_USER_LIST, AVAILABLE_CARD e PING_USER. Esistono quindi per ogni utente 2 descrittori di socket corrispondenti rispettivamente alla comunicazione di comandi lavagna => utente e vice versa.
Per quanto riguarda i comandi gestiti dalla lavagna innanzitutto, qualunque esso sia, si va a ricevere un byte rappresentativo del comando che l'utente vuole eseguire (e.g. HELLO = 0), si procede poi con lo scambio dei dati specifico del comando descritti in seguito:

- HELLO: si ricevono 2 byte di porta, si invia poi un byte per comunicare all'utente se la porta è disponibile o meno, in caso lo fosse si aspetta la ricezione di un byte qualunque che ha il significato di "via libera" per la connessione del socket destinato all'invio dei comandi verso l'utente.

- CREATE_CARD: si ricevono 106 byte: 4 byte di ID della card, 101 byte di testo attività e 1 byte di colonna, si invia poi un byte che ha lo scopo di comunicare all'utente se l'ID è disponibile.
- QUIT: non richiede comunicazione ulteriore, si procede semplicemente con la rimozione dell'utente.
- ACK_CARD: si ricevono 4 byte di ID della card, si invia poi un byte che ha lo scopo di comunicare all'utente se la card è già stata ackata (situazione anomala dovuta a errori della comunicazione).
- CARD_DONE: non richiede comunicazione ulteriore si procede con lo spostamento della card nella colonna DOING nella colonna DONE.

Ogni errore dovuto alla comunicazione nei comandi appena descritti viene gestito tramite disconnessione dell'utente che lo ha causato. Le rimozioni degli utenti sono gestite interamente dal thread che riceve comandi, sarà quindi lui stesso a aspettare che l'altro termini per procedere con la distruzione dell'utente.

UTENTE

I file relativi all'utente sono contenuti interamente nella cartella utente a sua volta contenuta nella cartella src. All'avvio l'utente procede con la connessione verso la lavagna con la porta inserita da linea di comando, una volta conclusa questa fase si procede con la HELLO che nel caso termini con successo (comunicazione avvenuta e porta disponibile) porta alla creazione di 2 thread distinti, uno per la ricezione di comandi da linea testuale trai quali: CREATE_CARD e QUIT e l'altro per ricevere i messaggi di: SEND_USER_LIST, AVAILABLE_CARD e PING_USER da parte della lavagna.
La comunicazione con la lavagna infatti avviene sempre utilizzando un approccio con socket bloccanti e thread associato, cosa diversa per la fase di scambio dei messaggi di CHOOSE_USER, infatti quest'ultima è stata implementata attraverso IO multiplexing principalmente per 2 ragioni: semplicità della gestione dello stato della comunicazione per ogni peer in quanto il messaggio è standard e uguale per tutti, trattasi di 2 byte di porta seguiti da 4 byte di intero rappresentativo del costo; e limitare il più possibile l'overhead spaziale e temporale dovuto alla creazione di un potenziale numero molto elevato di thread, in quanto si suppone che la macchina in cui gira il processo utente non sia dotata di risorse tali da potersi permettere un utilizzo così potenzialmente ampio di thread, a differenza della macchina dove verosimilmente sarà in esecuzione la lavagna che si suppone essere un server a tutti gli effetti.
Per quanto riguarda il formato dei messaggi inviati dalla lavagna inizialmente si procede come già descritto per l'invio del comando, successivamente si passa allo scambio dei dati con il formato descritto in seguito:

- SEND_USER_LIST: l'unico caso dove viene ricevuta inizialmente la lunghezza del messaggio successivo, sia len, si procede poi con la ricezione di len byte rappresentativi a 2 a 2 delle porte degli utenti.
- AVAILABLE_CARD: si ricevono 105 byte, 4 di id della card e 101 di testo attività.
- PING_USER: si invia un byte qualunque rappresentativo del messaggio PONG_LAVAGNA.