PROTOCOLLO DI COMUNICAZIONE

Nello scambio di informazioni tra C-S vengono applicati protocolli di tipo diverso in base alle esigenze di ciascun comando ma, in generale, i messaggi di tipo verboso attraversano la rete come coppia [len, msg], dove “len” rappresenta la lunghezza di “msg”, campo contenente i dati veri e propri. Un altro scambio di messaggi è invece relativo agli “acknowledgement”: si tratta di messaggi di dimensione limitata, come singoli *int* o *char* che, nella comunicazione da server a client, comunicano al secondo l’esito delle operazioni sul server. Ovviamente, sia il campo “len” che gli acknowledgement interi attraversano la rete secondo protocollo “binary”.

I messaggi del client sono formattati secondo funzione *format_by_protocol()*, la quale si limita a generare una stringa contente fino a tre campi inframezzati da spazio:

1. Un comando codificato come un singolo carattere, per risparmiare spazio.
2. Un numero variabile di argomenti (da zero a due). Il loro numero viene ottenuto direttamente sul server come valore di ritorno di *sscanf()*.

Quando un client risolve un enigma è il server a comunicare al client le ricompense ottenute, quali tempo e token; tali premi vengono inoltrati attraverso l’apposita funzione *prize_protocol()* la quale concatena:

1. Gli interi [token] e [time] che rappresentano rispettivamente token e tempo ottenuti immettendoli in rete come dati “binary”.
2. La stringa [granted], la quale racchiude il nome di un eventuale oggetto sbloccato secondo comunicazione “text”. Al suo termine segue il carattere ‘\0’, necessario per separarla dal campo successivo.
3. La stringa [message], riservata ad un eventuale messaggio da trasmettere, attraverso una comunicazione di tipo “text”.

SERVER

Il server consiste in un processo basato su I/O multiplexing. Questa scelta è stata presa tenendo in considerazione che:

1. Un server iterativo sarebbe risultato troppo rudimentale: l’utilizzo della *select()* riesce a facilitare di molto la gestione di socket differenti.
2. I diversi processi generati in un server di tipo multiprocesso avrebbero prodotto un overhead eccessivo e non avrebbero portato a grandi guadagni in performance vista le semplicità delle operazioni svolte.

Tutti i descrittori di socket vengono memorizzati in una lista di testa “sdlist”, elemento contenente il descrittore dello stdin, ordinata dal secondo elemento in poi per descrittori di socket decrescenti, in modo da trovare il maggiore in “sdlist.next->sd”. Ognuno di questi elementi contiene poi un campo “match”, il quale mantiene le informazioni relative alla partita intrapresa su quella connessione.

CLIENT

Il client prova il più possibile a farsi carico della complessità della comunicazione, ove possibile. Il suo ruolo più importante è sicuramente quello di validare gli input da tastiera prima di inoltrarli al server per evitare che a doverlo fare sia il server stesso. Per alleggerire ulteriormente il suo carico ho ritenuto opportuno incaricare il client di tenere traccia del tempo rimanente, dei token ottenuti e del numero di oggetti in mano al giocatore: sarà infatti quest'ultimo a chiudere la connessione qualora la partita termini. Ultimo ruolo del client è ovviamente quello di ricevere gli "acknowledgement" dal server per poi estrapolarne un significato e comunicarlo all'utente.

Questa scelta implementativa comporta però problemi relativi alla sicurezza della connessione: un client potrebbe infatti manomettere il codice client e di conseguenza anche i messaggi scambiati, di fatto "mentendo" al server. Questo aspetto è stato però messo da me in secondo piano perchè a mio parere non necessario al funzionamento dell'applicazione.

FUNZIONE "GRAFFITO"

Funzione per l'interazione tra client. Un graffito non è che un messaggio che può essere lasciato nei pressi di una qualsiasi locazione. Un utente può interagire con i graffiti in due modi:

1. *graffito locazione*: l'utente prova a visualizzare il graffito lasciato presso "locazione".
2. *graffito locazione text*: l'utente scrive "text" come graffito presso "locazione". Eventuali graffiti lasciati in precedenza sono sovrascritti.

I graffiti sono pensati per "scolorirsi" con il tempo; infatti, ogni qual volta un utente lascia la room che li contiene, parte dei loro da caratteri viene rimpiazzata da "#". Una volta scritto un graffito, questo è subito visibile a tutti gli utenti impegnati all'interno della stessa room fino allo stop del server.