Documentazione Progetto Reti Informatiche 2023/24 – Lorenzo Melani

Il progetto si compone di un file server e di un solo client in quanto ho deciso di implementare la funzione a piacere in forma di comunicazione tra due host che stanno giocando sul server.

L'invio e la ricezione dei messaggi sono gestiti da due funzioni, send_msg e recv_msg, che si occupano prima dello scambio della lunghezza del messaggio e successivamente del messaggio vero e proprio. Questa scelta comporta un maggiore traffico sulla rete ma dall'altra parte permette di non definire un protocollo unico per uno scambio di messaggi diversi tra di loro. Per lo scambio della lunghezza si utilizza un binary protocol, il buffer del messaggio invece viene scritto e riempito secondo un text protocol.

Per quanto riguarda il gioco, un utente può ottenere informazioni su oggetti e location con il comando look, può raccogliere fino a 5 oggetti con il comando take, due dei quali sono bloccati da enigmi (uno richiede l'inserimento di un codice numerico e l'altro la lettera corrispondente alla risposta corretta tra quelle proposte), e usarli una volta raccolti. Il comando use abbinato ai giusti oggetti permette di ottenere 2 token su 3, il terzo si ottiene grazie alla funzionalità a piacere. Ho aggiunto inoltre il comando drop per rilasciare un oggetto raccolto.

SERVER

Il server mantiene le informazioni di tutte le partite in corso in memoria mentre sul file utenti.txt scrive e legge le credenziali di accesso dei giocatori; si basa sull'I/O multiplexing che permette di gestire con semplicità molte richieste. Il server viene messo in ascolto al momento dell'esecuzione ma non risponde alle richieste finché non si inserisce il comando start da stdin. Il server riceve dal client solo comandi che può gestire e comunica al client l'esito del comando ricevuto ignorando eventuali parametri in eccesso (per esempio se il client inviasse "end oggetto" il server gestirebbe il comando end senza curarsi del parametro). Nel caso della funzionalità a piacere e in quello della gestione degli enigmi vi è uno scambio di messaggi ("ready", "ack", "nak") dal client al server non innescato dall'utente ma necessario al prelievo di una risposta pronta sul client (ready) e all'eventuale aggiunta di un token alla partita dell'utente dal cui host è arrivata la comunicazione ack/nak.

Il server gestisce i comandi ricevuti dall'utente inviando delle comunicazioni brevi, un po' impropriamente definite ack, che possono essere seguite dall'invio di un altro messaggio contenente informazioni richieste dall'utente. Prima di gestire il comando proveniente da un utente viene invocata la funzione gestione_timer che controlla se l'utente che ha inviato il comando non abbia esaurito il tempo a disposizione. Se il tempo è scaduto viene inviata la comunicazione EXTIM, viene rimossa la partita e chiuso il socket. Si noti che l'esaurimento del tempo è l'unica via che porta alla sconfitta. Nei casi in cui un comando ricevuto generi l'ottenimento di un token si verifica se esso è il terzo: se non lo è si invia NOWIN che verrà ignorato dal client, altrimenti si invia YUWIN e si chiude la partita. Il terzo e ultimo caso di chiusura di una connessione è il comando end, una volta ricevuto quello, il server invia OKEND e chiude il socket.

Le partite in corso sono tenute in una lista, ogni struct partita ha il proprio vettore di struct obj in quanto partite diverse possono avere gli stessi oggetti in stati diversi (magari bloccati per uno ma già sbloccati per l'altro). Le location e gli enigmi sono invece memorizzati in strutture dati statiche e comuni a tutte le partite visto che gli utenti non possono modificarli.

CLIENT

Il codice del client presenta due funzioni principali: gestione_comandi_partita e gestione_ack. Con la prima si esegue semplicemente l'inoltro del comando inserito da tastiera al server e ci si mette in attesa della risposta, se la keyword del comando non è riconosciuta il comando non viene inviato. La funzione gestione_ack, invece, gestisce tutte le possibili comunicazioni da parte del server (ad eccezione di NOWIN che viene ignorata). Nei casi in cui il comando ha avuto esito negativo si limita a stampare un messaggio a video, negli altri casi spesso sono necessarie altre azioni: per esempio se si riceve il messaggio DESCR significa che il comando look è andato a buon fine quindi ci si mette in attesa di un ulteriore messaggio contenente la descrizione richiesta che viene poi stampata.

Vista la necessità di mantenere più connessioni anche il client utilizza l'I/O multiplexing: nel set di descrittori da ascoltare sono inseriti solamente lo standard input e il listener, il socket di comunicazione con il server non è necessario in quanto gli scambi di messaggi con il server vengono gestiti conseguentemente a comandi inviati da standard input.

FUNZIONE A PIACERE: GETN

La funzione a piacere implementata serve al giocatore per ottenere un token indispensabile per la vittoria: all'utente viene assegnato randomicamente un numero tra 100 e 900 al momento dell'avvio, con il comando "getn username" il client ottiene il numero relativo alla stanza dell'utente di cui ha inserito l'username, successivamente viene richiesto di inserire il resto della divisione intera tra il proprio numero e quello ricevuto, se la risposta è corretta si ottiene un token. Al momento della connessione con il server, il client invia le informazioni relative al proprio listener socket,, quando viene effettuato il login/signup si memorizza il nome, allo start viene assegnato il numero e si viene considerati raggiungibili da altri utenti che invocano la getn. Un client che invoca la getn richiede al server l'indirizzo a cui contattare l'utente di cui ha inserito l'username nel comando, in caso l'utente sia online il server risponde con l'indirizzo, a questo punto il client stabilisce una connessione con l'altro host client da cui riceve il numero in modo trasparente per l'utente che non si accorge di nulla. Non è stata considerata la possibilità di stampare una lista di possibili username validi per il comando in caso un utente non conosca altri giocatori. Ricevuto il numero viene posto l'indovinello, la cui verifica della risposta avviene a livello client: se la risposta è corretta si invia "ack" al server che assegna un token, altrimenti si invia "nak". Ovviamente si può ottenere un solo token dall'utilizzo del comando getn.