Laboratorio per il corso di Reti

Sviluppare in linguaggio C un'applicazione di rete, costituita da un programma server ed un programma client, in cui il programma server calcola media e varianza di una serie di numeri interi positivi forniti dal client, secondo le specifiche sotto indicate.

SERVER

1. Il programma server viene eseguito indicando sulla riga di comando la porta sulla quale mettersi in ascolto
   `$ programma_server <numero porta>`
2. I messaggi inviati dal server hanno una lunghezza massima di 512 caratteri, devono essere terminati da un carattere di andata a capo ('`\n`'), e hanno il formato generale
   `<Esito> <Tipo> <Contenuto>`
   <Esito> identifica l'esito positivo o negativo del messaggio e può assumere i valori 'OK' ed 'ERR'. *<Tipo>* identifica il comando al quale la risposta fa riferimento, o la categoria di risposta. *<Contenuto>* è il contenuto della risposta. Le tre sezioni sono separate da un singolo spazio (carattere ASCII 32, 0x20).
3. All'apertura della connessione il server manda un messaggio di benvenuto, nel formato
   `OK START <Messaggio>`
   ovvero la stringa '`OK START`', seguita da uno spazio e da una stringa personalizzabile dal server.
4. Il server si pone in attesa di un messaggio da parte del client. Il messaggio deve essere costituito da una stringa di testo nel formato
   `<Numero_dati> <dato1> <dato2> <datoN>`
   ovvero stringa con una serie di valori numerici interi positivi, terminata da un carattere di andata a capo ('`\n`'). Il primo valore indica il numero di dati presenti nella stringa, ed è seguito dai dati. Ad esempio:
   `4 11 21 9 11`
   I messaggi del client sono costituiti da stringhe della lunghezza massima di 512 caratteri, terminate da un carattere di andata a capo ('`\n`').
5. Il server esamina il messaggio ricevuto e ne verifica la correttezza sintattica. Se il messaggio non è sintatticamente corretto risponde con il messaggio
   `ERR SYNTAX <messaggio>`
   ovvero la stringa '`ERR SYNTAX`' seguita da uno spazio e da un messaggio di testo che descrive la natura dell'errore, e chiude la connessione.
6. Se il messaggio è sintatticamente corretto, il server verifica la correttezza semantica del messaggio, ovvero se il contatore è coerente con il contenuto del messaggio stesso.
   a. Se il numero di dati letti è corretto e maggiore di zero, memorizza i dati e risponde con il messaggio
      `OK DATA <numero_dati_letti>`
      ovvero la stringa '`OK DATA`' seguita da uno spazio e da una stringa numerica che rappresenta il numero di dati estratti dal messaggio ricevuto.
   b. Se il numero di dati letti non è corretto, ovvero il valore dichiarato non è coerente con il contenuto del messaggio, il server risponde con il messaggio
      `ERR DATA <messaggio>`

ovvero la stringa ‘ERR DATA’ seguita da uno spazio e da un messaggio di testo che descrive la natura dell’errore, e chiude la connessione.

c. Se il numero di dati letti è uguale a 0, ovvero il client aveva mandato il messaggio
   0
   il server elabora tutti i dati ricevuti e ne calcola la media e la varianza dei campioni.
   i. Se media e varianza dei campioni possono essere calcolati correttamente, il server trasmette il messaggio
      OK STATS <numero totale campioni> <media> <varianza>
      dove il numero dei campioni è un intero, mentre media e varianza sono dei valori *floating point*. Ad esempio
      OK STATS 15 11.23 2.28
   ii. Se media o varianza dei campioni non possono essere calcolati correttamente, il server trasmette il messaggio
      ERR STATS <Messaggio>
      ovvero la stringa ‘ERR STATS’, seguita da uno spazio e da una stringa personalizzabile dal server che descrive la natura dell’errore.

7. Dopo aver ricevuto i dati (caso 6.a), il server torna al punto 4 in attesa di un nuovo messaggio del client.
8. Dopo avere mandato il messaggio finale o aver mandato i messaggi di errore (casi 6.b e 6.c), il server chiude la connessione e si pone in attesa della richiesta di un nuovo client.

CLIENT

1. Il programma client viene eseguito indicando sulla riga di comando l’indirizzo IPv4 del server da contattare e la porta sulla quale contattarlo
   $ programma_client <indirizzo_server> <numero_porta>
2. **Il client elabora sempre i messaggi che riceve dal server, ovvero non ne presenta il contenuto direttamente all’utente, ma rimuove qualsiasi delimitatore del protocollo e mostra all’utente l’informazione ricevuta in maniera chiara e contestualizzata**
3. Dopo l’apertura della connessione il client si aspetta di ricevere dal server il messaggio di benvenuto, nel formato
   OK START <Messaggio>
   ovvero la stringa ‘OK START’, seguita da uno spazio e da una stringa personalizzata dal server. Il messaggio deve avere una lunghezza massima di 512 caratteri ed è terminato da un carattere di andata a capo (‘\n’)
4. Il client presenta all’utente il messaggio del server, senza il delimitatore “OK START“ che costituisce una parola chiave del protocollo di scambio e non fa parte del messaggio del server
5. Il client spiega chiaramente all’utente lo scopo del programma, le opzioni a disposizione, il formato atteso e le modalità di funzionamento previsto.
6. Il client sollecita l’utente ad inserire i numeri di cui deve essere calcolata media e varianza, secondo il criterio che preferisce.
   a. Una volta verificata la congruità dei dati inseriti, in relazione allo scopo del programma, il client trasmette i dati al server all’interno di uno o più messaggi, secondo il formato descritto al punto 4 del server. I messaggi trasmessi **devono** essere sintatticamente e semanticamente corretti.

    b. A seconda della modalità di inserimento dei dati, quando il client non ha più dati da trasmettere manda un messaggio con 0 (zero) dati, segnalando implicitamente l'attesa dei risultati da parte del server.
    c. Dopo aver trasmesso ogni messaggio, il client si pone in attesa della risposta del server.
        i. In caso di risposta positiva (OK), il client prosegue le sue operazioni, eventualmente dando riscontro del successo all'utente
        ii. In caso di errore (ERR), il client riporta all'utente l'eventuale messaggio del server (senza i delimitatori del protocollo), e chiude la connessione.
7. Dopo aver trasmesso ogni messaggio, il client si pone in attesa di una risposta da parte del server. Le risposte possibili sono:
    a. `OK DATA <numero>`
    b. `OK STATS <numero> <media> <varianza>`
    c. `ERR DATA <messaggio>`
    d. `ERR STATS <messaggio>`
8. Se la risposta indica la corretta ricezione di dati (caso 7a), e il numero di dati ricevuti corrisponde al numero di dati trasmessi, il client torna al punto 5 (o al punto 6a se i dati dell'utente sono già stati raccolti) e manda un nuovo insieme di dati al server
9. Se la risposta indica la corretta ricezione di dati (caso 7a), ma il numero di dati ricevuti è diverso dal numero di dati trasmessi, il client segnala l'errore all'utente e torna al punto 6b per mandare al server il messaggio di fine dati "`0`"
10. Se il client ha spedito il messaggio di fine dati "0" e riceve come risposta l'elaborazione con successo (caso 7b), il client ne estrae le informazioni e le riporta all'utente **in maniera opportunamente informativa**, dopo di che chiude la connessione e termina l'esecuzione
11. Se il client ha spedito il messaggio di fine dati "0" e la risposta indica un errore (caso 7d), il client lo comunica all'utente riportando il messaggio del server (senza i delimitatori del protocollo), insieme ad un eventuale messaggio da parte del client, dopo di che chiude la connessione e termina l'esecuzione
12. In ogni altro caso di errore (caso 7c e possibili casi non previsti), il client lo comunica all'utente riportando l'eventuale messaggio del server (senza i delimitatori del protocollo), insieme ad un eventuale messaggio da parte del client, dopo di che chiude la connessione e termina l'esecuzione

Entrambi i programmi devono gestire opportunamente tutti i possibili errori che si possono verificare in fase di apertura e chiusura delle connessioni. Entrambi i programmi devono operare in maniera tale da rispettare il protocollo previsto e reagire opportunamente a violazioni di tale protocollo.
Tutti i messaggi scambiati sono costituiti da stringhe terminate da un carattere di andata a capo ('`\n`'). Tutte le parole chiave sono rappresentate da caratteri maiuscoli.
Entrambi i programmi client e server devono essere in grado in inter-operare con programmi analoghi che rispettino i requisiti elencati: per questo motivo i messaggi di rete devono rispettare rigorosamente i formati definiti. L'interoperabilità è un requisito fondamentale che verrà verificato in fase di valutazione. Ogni altro aspetto di interazione tra il client, il server e l'utente, può essere personalizzato a piacimento, ad esempio i messaggi a schermo di notifica o di errore, nei limiti dell'utilizzabilità del programma stesso.
La definizione della modalità di acquisizione dati da parte del client viene lasciata alla scelta dello studente. **La conoscenza di queste specifiche non deve essere un requisito per l'usabilità dei programmi.**

Esempi di comunicazione sulla rete

```
S: OK START Benvenuto, mandami i tuoi dati
C: 2 10 20
S: OK DATA 2
C: 1 12
S: OK DATA 1
C: 0
S: OK STATS 3 14.0 28.0
```

```
S: OK START Benvenuto, mandami i tuoi dati
C: 1 10
S: OK DATA 1
C: 0
S: ERR STATS Non posso calcolare la varianza di 1 campione
```