Laboratorio per il corso di Reti I (A.A. 2018/19) – Applicazione A

Sviluppare in linguaggio C un'applicazione di rete, costituita da un programma server ed un programma client, in cui il programma server calcola la distribuzione dei caratteri (diversi da spazi) che compongono un testo fornito dal client, secondo le specifiche sotto indicate.
In particolare, sono da tenere in considerazione tutti i caratteri ASCII a 7-bit alfanumerici (classe *alnum*), ovvero i caratteri alfabetici (classe *alpha*) e le cifre numeriche (classe *digit*). Le classi di caratteri sono quelle definite dall'ANSI C Standard Library e incluse in *ctype.h*. Questo esclude spazi, punteggiature, caratteri di controllo, e caratteri accentati. Per semplicità e uniformità si suppone che il server operi secondo la *locale* di default "C" o "POSIX".

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 di *comando* da parte del client.
5. I possibili comandi sono:

Comando	**Descrizione**
TEXT	Testo da analizzare
HIST	Richiede la distribuzione dei caratteri
EXIT	Termina la comunicazione, richiedendo i dati
QUIT	Termina la comunicazione, senza richiedere i dati

6. I comandi sono costituiti da stringhe della lunghezza massima di 512 caratteri, terminate da un carattere di andata a capo ('\n').
7. La semantica e il formato dei comandi sono i seguenti:
   a. Il comando TEXT permette al client di indicare del testo da analizzare, e ha il formato
      `TEXT <testo> <contatore>`
      ovvero la stringa '`TEXT`', seguita da uno spazio, dal testo da analizzare, da uno spazio, da un contatore di controllo, e terminata da un carattere di andata a capo ('\n'). Il contatore di controllo `<contatore>` indica il numero di caratteri validi (ovvero alfanumerici, come descritto all'inizio della traccia) presenti nel testo `<testo>` ed è una stringa che rappresenta un valore numerico intero positivo. Ad esempio:

`TEXT Era una notte buia e tempestosa 26`
Se il testo da analizzare supera la lunghezza permessa dai messaggi, il client può utilizzare più comandi TEXT per trasmettere il testo.

b. Il comando HIST richiede al server la distribuzione (istogramma) dei caratteri non spazio contenuti nel testo fornito dall'utente (tramite uno o più comandi TEXT) e ha il formato
`HIST`
ovvero la stringa '`HIST`' seguita da un carattere di andata a capo ('`\n`').

c. Il comando EXIT premette al client di richiedere al server la distribuzione (istogramma) dei caratteri non spazio contenuti nel testo fornito dall'utente (tramite uno o più comandi TEXT), chiudendo al tempo stesso la connessione, e ha il formato
`EXIT`
ovvero la stringa '`EXIT`' seguita da un carattere di andata a capo ('`\n`'). L'effetto del comando EXIT è equivalente al comando HIST seguito da un comando QUIT.

d. Il comando QUIT permette al client di chiudere la connessione senza chiedere al server l'esito dei suoi calcoli e ha il formato
`QUIT`
ovvero la stringa '`QUIT`' seguita da un carattere di andata a capo ('`\n`').

8. In tutti i comandi, il carattere di separazione tra i campi, se presente, è sempre costituito da un singolo spazio (carattere ASCII 32, 0x20). All'interno del testo da analizzare, sono da considerarsi solo i caratteri alfanumerici, riconosciuti come tali dal sistema locale (classe *alnum*), e qualsiasi altro carattere non va incluso nel contatore di controllo.

9. 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.

10. Se il messaggio è sintatticamente corretto, il server risponde in base al comando ricevuto:

a. Alla ricezione del comando TEXT, il server verifica la correttezza semantica del messaggio, ovvero se il contatore è coerente con il contenuto del messaggio stesso.

i. Se il messaggio è semanticamente corretto, il server elabora la stringa ricevuta e risponde con un messaggio di conferma, nel formato
`OK TEXT <contatore>`
ovvero la stringa '`OK TEXT`' seguita da uno spazio e da una stringa numerica che rappresenta il valore corretto e verificato del contatore di controllo.

ii. Se il messaggio non è semanticamente corretto, il server ignora la stringa ricevuta, risponde con un messaggio di errore e chiude la connessione. Il messaggio di errore ha formato
`ERR TEXT <messaggio>`
ovvero la stringa '`ERR TEXT`' seguita da uno spazio e da un messaggio di testo che descrive la natura dell'errore, e chiude la connessione.

b. Alla ricezione del comando HIST, il server risponde con uno o più messaggi nel formato
`OK HIST <risposta>`
ovvero la stringa '`OK HIST`' seguita da uno spazio e da una sequenza di coppie `<carattere>:<contatore>` separate da spazi, e terminata da un carattere di

andata a capo (‘\n’). Solo i caratteri presenti nel testo analizzato devono essere inclusi nella risposta.
Poiché i messaggi hanno una lunghezza massima, il server può utilizzare più messaggi di risposta per spedire tutte le informazioni al client. La sequenza di messaggi è quindi sempre terminata da un messaggio in cui la risposta è costituita dalla stringa ‘END’, ovvero
`OK HIST END`
Un esempio di risposta al comando HIST può quindi essere la seguente sequenza di messaggi
```
OK HIST E:1 a:4 b:1 e:4 i:1 m:1 n:2
OK HIST o:2 p:1 r:1 s:2 t:4 u:2
OK HIST END
```

c. Alla ricezione del comando EXIT, il server risponde come se fosse stato ricevuto il comando HIST, ovvero con la sequenza di messaggi prevista dal comando HIST, seguita da un messaggio di chiusura nel formato
`OK EXIT <messaggio>`
ovvero la stringa ‘OK EXIT’ seguita da uno spazio e da un messaggio di commiato, e chiude la connessione.

d. Alla ricezione del comando QUIT, il server risponde con un messaggio di chiusura nel formato
`OK QUIT <messaggio>`
ovvero la stringa ‘OK QUIT’ seguita da uno spazio e da un messaggio di commiato, e chiude la connessione.

11. Dopo aver elaborato i dati e mandato il messaggio finale (casi 10.c e 10.d), o dopo aver mandato qualsiasi messaggio di errore, 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. Le opzioni che il client offre all’utente sono:
   a. Inserimento del testo

   b. Analisi del testo
   c. Uscita dal programma (con analisi del testo)
   d. Abbandono del programma (senza analisi del testo)

   e corrispondono ai comandi del protocollo descritti al punto 5 del server.
7. Inserimento del testo
   a. Il client sollecita l'utente ad inserire il testo che deve essere analizzato, secondo il criterio che preferisce (da tastiera, da file, una parola alla volta, ecc.). Una volta verificata la congruità del testo inserito, in relazione all'insieme di caratteri per il quale è previsto il calcolo della distribuzione, il client lo trasmette al server all'interno di uno o più comandi TEXT, secondo il formato descritto al punto 7.a del server. I messaggi trasmessi devono essere sintatticamente e semanticamente corretti.
   b. Dopo aver trasmesso ogni comando, 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
8. Analisi del testo
   a. Su richiesta dell'utente, il client richiede la distribuzione dei caratteri del testo finora inserito al server, trasmettendo al server un comando HIST, secondo il formato descritto al punto 7.b del server.
   b. Dopo aver trasmesso il comando, il client si pone in attesa della risposta del server.
      i. In caso di risposta positiva (OK), il client analizza la sequenza di messaggi descritta al punto 10.b del server, ne estrae le informazioni e le riporta all'utente **in maniera opportunamente informativa**. Questo significa che non è corretto proporre all'utente il contenuto dei messaggi ricevuti dal server in maniera acritica e senza alcuna elaborazione.
      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
9. Uscita dal programma (con analisi del testo)
   a. Il client trasmette al server il comando EXIT, secondo il formato descritto al punto 7.c del server
   b. Dopo aver trasmesso il comando, il client si pone in attesa della risposta del server.
      i. In caso di risposta positiva (OK), il client analizza la sequenza di messaggi descritta al punto 10.b del server, ne estrae le informazioni e le riporta all'utente **in maniera opportunamente informativa**. Poi si aspetta di ricevere il messaggio descritto al punto 10.c del server, ne riporta il contenuto (senza i delimitatori del protocollo) all'utente, e chiude la connessione.
      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
10. Abbandono del programma (senza analisi del testo)
   a. Il client trasmette al server il comando QUIT, secondo il formato descritto al punto 7.d del server
   b. Dopo aver trasmesso il comando, il client si pone in attesa della risposta del server.

   i. In caso di risposta positiva (OK) secondo il formato descritto al punto 10.d del server, il client ne riporta il contenuto (senza i delimitatori del protocollo) all'utente, e chiude la connessione.
   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
11. In ogni caso di errore e dopo la chiusura della connessione, il client 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.

Esempi di comunicazione sulla rete

```
S: OK START Benvenuto, mandami i tuoi dati
C: TEXT questo e' il testo 14
S: OK TEXT 14
C: HIST
S: OK HIST e:3 i:1 l:1 q:1 o:2
S: OK HIST u:1 t:3 s:2
S: OK HIST END
C: QUIT
S: OK QUIT Arrivederci
```

```
S: OK Benvenuto, mandami i tuoi dati
C: TEXT questo testo 12
S: ERR TEXT Numero di caratteri non coerente
```