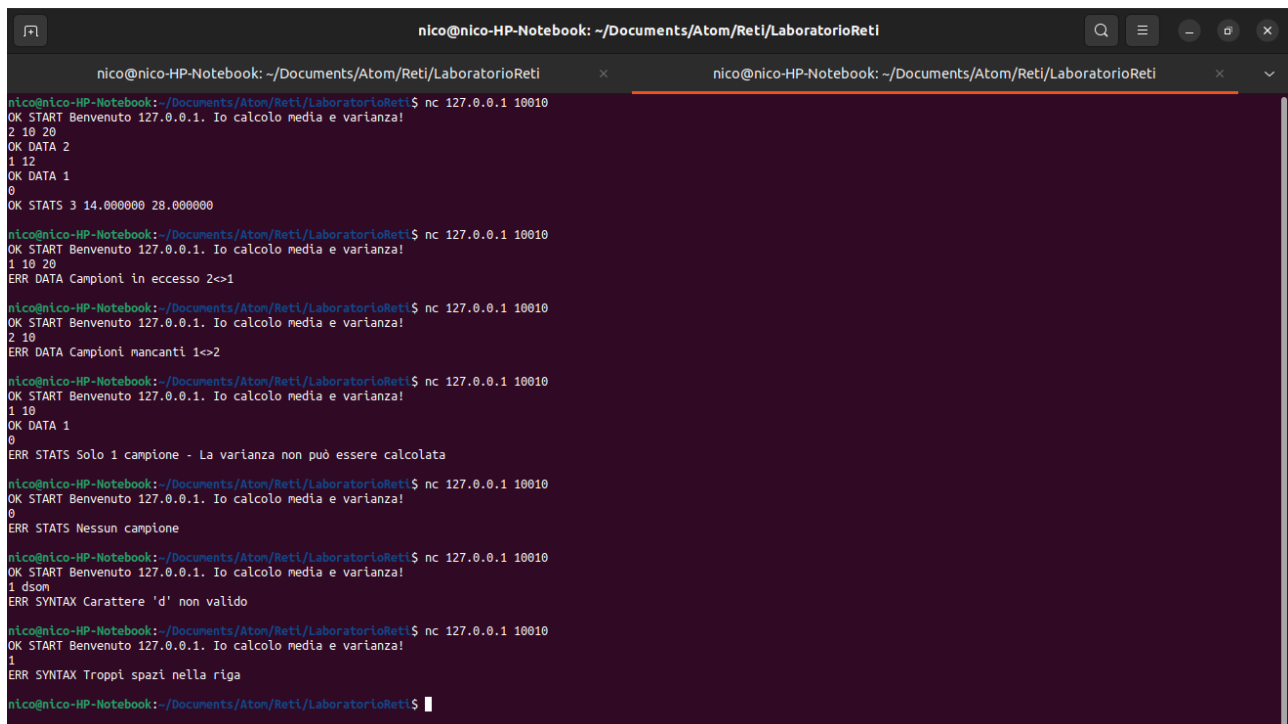


L'applicazione è stata realizzata su un terminale con installato Ubuntu 22.10 e il tutto funziona correttamente come nell'implementazione di riferimento.

Ho iniziato con lo sviluppo del server partendo da quello fornito nelle lezioni di laboratorio e andando man mano a controllare il funzionamento tramite il comando netcat. La logica del server si basa sull'andar ad inserire tutto quello che ricevo sulla porta da parte del client all'interno di un buffer, il quale andrà ad inserire tutti i valori all'interno di un array di double (nel caso ci fosse un input valido) tramite la funzione strtok(). In seguito, verifico che i dati inseriti siano congrui al numero di dati dichiarati e se lo sono, il server farà gli opportuni calcoli e restituirà la risposta al client.

Sono stati fatti i vari controlli sugli errori di tipo SYNTAX, DATA e STATS e la comunicazione in rete sarà come segue:



```
nico@nico-HP-Notebook: ~/Documents/Atom/Reti/LaboratorioReti
nico@nico-HP-Notebook: ~/Documents/Atom/Reti/LaboratorioReti
nico@nico-HP-Notebook:~/Documents/Atom/Reti/LaboratorioReti$ nc 127.0.0.1 10010
OK START Benvenuto 127.0.0.1. Io calcolo media e varianza!
2 10 20
OK DATA 2
1 12
OK DATA 1
0
OK STATS 3 14.000000 28.000000

nico@nico-HP-Notebook:~/Documents/Atom/Reti/LaboratorioReti$ nc 127.0.0.1 10010
OK START Benvenuto 127.0.0.1. Io calcolo media e varianza!
1 10 20
ERR DATA Campioni in eccesso 2<=1

nico@nico-HP-Notebook:~/Documents/Atom/Reti/LaboratorioReti$ nc 127.0.0.1 10010
OK START Benvenuto 127.0.0.1. Io calcolo media e varianza!
2 10
ERR DATA Campioni mancanti 1<=2

nico@nico-HP-Notebook:~/Documents/Atom/Reti/LaboratorioReti$ nc 127.0.0.1 10010
OK START Benvenuto 127.0.0.1. Io calcolo media e varianza!
1 10
OK DATA 1
0
ERR STATS Solo 1 campione - La varianza non può essere calcolata

nico@nico-HP-Notebook:~/Documents/Atom/Reti/LaboratorioReti$ nc 127.0.0.1 10010
OK START Benvenuto 127.0.0.1. Io calcolo media e varianza!
0
ERR STATS Nessun campione

nico@nico-HP-Notebook:~/Documents/Atom/Reti/LaboratorioReti$ nc 127.0.0.1 10010
OK START Benvenuto 127.0.0.1. Io calcolo media e varianza!
1 dson
ERR SYNTAX Carattere 'd' non valido

nico@nico-HP-Notebook:~/Documents/Atom/Reti/LaboratorioReti$ nc 127.0.0.1 10010
OK START Benvenuto 127.0.0.1. Io calcolo media e varianza!
1
ERR SYNTAX Troppi spazi nella riga

nico@nico-HP-Notebook:~/Documents/Atom/Reti/LaboratorioReti$
```

Per il debugging sono andato a compilare il file con gcc server.c -o server -Wall -Wextra -Wconversion e come ho detto prima con il comando nc.

Per quanto riguarda il client sono sempre partito dal programma fornito nelle lezioni di laboratorio. Inizialmente, dopo aver ricevuto il messaggio di benvenuto da parte del server, viene chiesto all'utente di inserire i propri dati nel formato descritto nella traccia e se essi non saranno sintatticamente e semanticamente corretti non verranno inviati al server. Dopo aver inviato i messaggi al server, il client riceve la risposta e vede se è giusta per quanto riguarda i calcoli sui dati ricevuti dal server (OK DATA non la media e la varianza). Essendo che i controlli vengono fatti dal client, se tutto va bene non verranno mai visualizzati da parte dell'utente gli errori del server ma nel caso qualcosa non andasse il programma li farebbe vedere. Per il debugging sono andato a compilare il file con gcc client.c -o client -Wall -Wextra -Wconversion.

Un esempio di comunicazione è come segue:

```
nico@nico-HP-Notebook: ~/Documents/Atom/Reti/LaboratorioReti
nico@nico-HP-Notebook: ~/Documents/Atom/Reti/LaboratorioReti$ ./client 127.0.0.1 10010
Socket Creato!
Connessione Accettata!
Benvenuto 127.0.0.1. Io calcolo media e varianza!
.....
Istruzioni d'uso:
Inserisci i dati di cui desideri calcolare la media e la varianza.
I dati inseriti devono seguire il seguente formato:
<Numero dati> <dato1> <dato2> <daton>
Separare ciascun dato con uno spazio. Una volta inseriti tutti i valori desiderati,
inserisci '0' e il programma calcolerà e restituirà i risultati.

Inserisci i numeri separati da spazio (0 per terminare): 2 10 20
Dati correttamente ricevuti dal server.
Inserisci i numeri separati da spazio (0 per terminare): 1 12
Dati correttamente ricevuti dal server.
Inserisci i numeri separati da spazio (0 per terminare): 0

Media: 14.0
Varianza: 28.0

Connessione chiusa!
.....
nico@nico-HP-Notebook: ~/Documents/Atom/Reti/LaboratorioReti$ ./client 127.0.0.1 10010
Socket Creato!
Connessione Accettata!
Benvenuto 127.0.0.1. Io calcolo media e varianza!
.....
Istruzioni d'uso:
Inserisci i dati di cui desideri calcolare la media e la varianza.
I dati inseriti devono seguire il seguente formato:
<Numero dati> <dato1> <dato2> <daton>
Separare ciascun dato con uno spazio. Una volta inseriti tutti i valori desiderati,
inserisci '0' e il programma calcolerà e restituirà i risultati.

Inserisci i numeri separati da spazio (0 per terminare): 2 10 20
Dati correttamente ricevuti dal server.
Inserisci i numeri separati da spazio (0 per terminare): 2 12

ERRORE I messaggi trasmetti devono essere sintatticamente e semanticamente corretti! (Dati in eccesso)

Inserisci i numeri separati da spazio (0 per terminare): 1 12
Dati correttamente ricevuti dal server.
Inserisci i numeri separati da spazio (0 per terminare): 0

Media: 14.0
Varianza: 28.0
```

Note aggiuntive:

Il server andrà a tenere in sospeso al massimo 5 client e il sesto verrà scartato subito.

Sono andato ad utilizzare una funzione `is_numeric()` per verificare che all'interno dei singoli token ci fossero numeri e non caratteri andando ad utilizzare una libreria che permette l'utilizzo di variabili booleane (sarebbero potute essere tranquillamente interi):

```
1 bool is_numeric(const char *str, char *errorChar)
2 {
3     size_t len = strlen(str);
4
5     for (size_t i = 0; i < len; i++)
6     {
7         if (str[i] == '\n')
8             continue;
9         else if (str[i] < '0' || str[i] > '9')
10        {
11            *errorChar = str[i];
12            return false;
13        }
14    }
15
16    // Prova a convertire la stringa in intero usando sscanf
17    int num;
18    if (sscanf(str, "%d", &num) != 1)
19    {
20        *errorChar = '\0';
21        return false;
22    }
23
24    return true;
25 }
```