# ISTRUZIONI PER L'USO

- Compilare il Server o il Client tramite make file (make server; make client) oppure tramite gcc con il flag pthread.
- Non c'è bisogno di fare alcuna azione per i log del server o del client, ci farannp da soli.

## SERVER

- Avviare il server fornendogli in input via argv l'ip numerico formato x.x.x.x da usare (argv[1]) e la porta (argv[2]) su cui ascoltare richieste.
- Per arrestare correttamente il server, chiudendo tutte le connessioni e i file descriptor aperti e avvisando anche i client con un messaggio, bisogna mandare il segnale SIGUSR1 al pid del processo server, il pid verrà stampato sulla shell insieme al comando da chiamare.

## CLIENT

- Avviare il client fornendogli in input via argv l'ip numerico del server a cui connettersi formato x.x.x.x da usare (argv[1]) e la porta (argv[2]) su cui leggere e mandare messaggi.
- Una volta avviato il programma chiederà di inserire un username che deve essere inserito senza spazi, altrimenti viene presa solo la prima parola (non sarà ancora stata fatta richiesta di connessione al server).
- Una volta inserito il nome si sarà acceduti nella chatroom, tutti gli altri utenti verranno avvisati della vostra presenza tramite un messaggio joined, per scrivere basta inserire input da tastiera sul terminale (verrà preso tutto il testo scritto) fino a 256 caratteri, se ecceduti verranno mandati solo i caratteri successivi ai 256.
- Per uscire dalla chatroom in modo corretto bisogna scrivere nella chat "__quit__", così verrà chiusa correttamente la connessione con il server, mandando un messaggio di log-out a tutti gli altri utenti.

# FUNZIONALITÀ IMPLEMENTATE

## SERVER

- Mette a disposizione a ogni client che si connette un nuovo thread che si occupa di estrarre i messaggi entranti e metterli in un buffer in ordine di timestamp in attesa di essere spediti.
- Permette di inserire l'ip del server e la porta su cui ascoltare.
- Vi è un thread consumatore che si occupa di prendere messaggi da un buffer condiviso e spedirli a ogni client fatta eccezione per il client mittente.
- I messaggi vengono inseriti per timestamp all'interno del buffer, il buffer cresce se raggiunge la capienza massima.
- I client che appendono i messaggi al buffer ne segnalano l'arrivo al consumatore che si mette in attesa del lock sul buffer, se un altro produttore appende un messaggio in quel momento viene inserito in ordine di timestamp.
- Il server è in grado di estrarre il timestamp del messaggio grazie al protocollo implementato (vedere giù).
- Il server tiene degli array dinamici per mantenere traccia dei client file descriptor e avere un buffer flessibile in cui appendere messaggi.
- Per il global log Il server salva i messaggi mandati dagli utenti appendendo sotto il messaggio l'indirizzo ip dell'utente. Inoltre, se non è già stato creato, il server al suo avvio crea il suo ambiente per lo storing dei log (directory + file con permessi adatti).
- Permette di chiudere il server tramite il segnale SIGUSR1 chiudendo tutte le connessioni e avvertendo i client.
- Quando un client si collega manda una stringa con all'interno il numero di client attualmente collegati al server.

## CLIENT

- Mette a disposizione 2 thread, uno che ascolta i messaggi in entrata e un altro che si occupa di prendere gli input da tastiera dell'utente e spedirli al server.

- Dà l'opportunità al client di loggare tramite un nome di 16 caratteri preso in input da tastiera, mandando un messaggio "Has joined-->username". Una volta inserito lo username verrà stampata una ASCII art, stampata una stringa che spiega come uscire e il numero di utenti online.
- Permette di inserire l'indirizzo ip del server e la porta.
- Permette di uscire dalla chatroom inserendo come messaggio "__quit__" così mandando un messaggio "Has left the chatroom--> username".
- Per il local log Il server salva i messaggi mandati dall'utente. Inoltre, se non è già stato creato, il client al suo avvio crea il suo ambiente per lo storing del log (directory + file con permessi adatti).
- Il client quando manda un messaggio ha una funzionalità di wrapping, praticamente incapsula il messaggio secondo il protocollo (vedere giù), inserendo timestamp, username, messaggio.
- Il client estrae il timestamp e lo rende human friendly tenendolo con una stringa contenente anno-mese-giorno ore:minuti:secondi.

# PROTOCOLLO

- E' stato implementato un protocollo per standardizzare i messaggi tra client e server e facilitare la reciproca comunicazione. Questo consiste nell' incapsulare il messaggio in:
    1. timestamp anno-mese-giorno ora:minuto:secondo
    2. username
    3. messaggio

  In questo modo il server quando arriva un messaggio è in grado di individuare il timestamp e prima di appendere il messaggio al buffer confrontarlo con tutti gli altri e metterlo alla posizione corretta.

# DYNAMIC ARRAY

- Array dinamico auto riallocante.
- Permette di inserire un unsigned long che rappresenta una generica locazione in memoria o un numero.
- Permette di fare inserimenti in  punti specifici o in coda alla lista.
- Permette di rimuovere un elemento ricercandolo all'interno della lista.
- Permette di inizializzare una mutex da utilizzare, non è pre inizializzata per lasciare più flessibilità.
- Permette di liberare completamente la locazione di memoria allocata dall'array.
- Permette di effettuare un trim sulla lista, liberandola e reinizializzandola a una lunghezza standard di 10 elementi.

# TEST EFFETTUATI

## DYNAMIC ARRAY

- Sono stati effettuati test variando la grandezza e la quantità degli elementi su ogni funzionalità implementata e osservando il comportamento, fino ad arrivare a quello desiderato.

## SERVER

- Sono stati effettuati test sul funzionamento delle lock e delle condition allargando il tempo di possesso di esse da parte dei thread.
- E' stato effettuato il test dell'ordinamento secondo timestamp allargando la finestra temporale di ricezione dei messaggi del consumatore, facendo rilasciare la per riprenderla x secondi dopo, in quel modo è stato possibile passare dei messaggi con timestamp personalizzati (grazie al protocollo utilizzato) che risalivano a tempi inferiori. I messaggi vengono correttamente posizionati nel buffer.
- Sono stati effettuati dei test sul modo in cui il consumer prende e spedisce messaggi con più client, facendo collegare numerosi client allo stesso server e mandando messaggi ripetutamente e rapidamente da questi.
- Sono stati fatti test sul meccanismo di producer consumer aumentando le finestre temporali in cui detengono le lock.
- Sono stati fatti dei test sul funzionamento delle condition semplicemente facendo stampare al consumer quando entrava nella wait e quando usciva monitorando variabili importanti per il suo funzionamento tra le quali l'uso del buffer.
- E' stato testato il funzionamento delle dynamic list applicate alla realtà del server monitorando i valori e i contenuti di esse al log-in e log-out dei client, e monitorando l'allargamento e/o restringimento del buffer dei messaggi.
- Sono stati fatti test sulla creazione delle directory e file semplicemente ripetendo i passaggi e osservando i risultati ottenuti.
- E' stata testata l'intercettazione del signal SIGUSR1 semplicemente usando il comando kill -s 10 pid.

## CLIENT

- Sono stati effettuati test sul funzionamento della ricezione e dell'invio dei messaggi collegandosi al server tramite shell diverse e monitorando i comportamenti.
- Sono stati fatti test sulla creazione delle directory e file semplicemente ripetendo i passaggi e osservando i risultati ottenuti.
- Sono stati effettuati test sull'incapsulamento del messaggio semplicemente calcolando i massimi valori possibili che i 3 elementi potevano assumere (contando anche caratteri aggiuntivi) ed è stato creato un buffer in grado di contenerli per poi essere spedito al server. Sono state fatte svariate prove.