INFORMAZIONI GENERALI

L'applicazione cerca di essere il più **user-friendly** possibile, automatizzando quanti più meccanismi possibili e stampando a terminale dei messaggi chiari e puliti.

Con il comando '**signup**' si registra un nuovo utente ma non si esegue effettivamente il login: per questo va usato il comando '**in**'. Quest'ultimo comando è stato modificato in modo che prenda in input solo username e password: la porta del server può essere specificata da terminale come secondo parametro: './dev <client port> [server port]'.
Sono consentite **passphrase** al posto delle semplici password.
Finché non si esegue il login non sarà permesso compiere nessun'altra operazione.
L'autenticazione è possibile solo se il server è online.

Dopo l'autenticazione è possibile ricevere notifiche di messaggi e inviti a chat di gruppo, oltre ad eseguire tutti i comandi delle specifiche. Nel caso si esegua '**chat <username>**':

- Se *username* è offline, i messaggi vengono inviati al server che li memorizzerà
- Se *username* è online, i messaggi vengono inviati direttamente al device (peer-to-peer) senza l'intermediazione del server

Se *username* va online in un secondo momento, il device si aggancerà automaticamente a lui stabilendo una connessione peer-to-peer e smettendo di comunicare con il server.
Durante una chat si possono invitare altri membri (online) digitando '**\u**'.
I messaggi/file inviati nella chat sono automaticamente inviati a tutti i membri.
Se un membro esce dalla chat ('**\q**') continuerà a ricevere le notifiche di messaggi ricevuti.
Se un membro si disconnette ('**out**'), i messaggi verranno automaticamente rediretti al server che li memorizzerà per l'invio futuro.
Un utente può partecipare solo ad una chat di gruppo per volta: altri inviti verranno automaticamente rifiutati.
Quando si entra in modalità chat il terminale viene pulito e saranno visibili solo i messaggi della chat.
Il **file** che si vuole **condividere** deve trovarsi nella cartella './shared/<proprietario>/'. Un file ricevuto si troverà nella cartella './shared/<ricevente>/'.

In caso di errore di una ***send(…)*** o una ***recv(…)*** viene sempre stampato un messaggio di errore, in modo da informare l'utente di ogni possibile guasto.

Per semplicità, si è supposto che i log delle chat e il file di log di login/logout siano condivisi tra client e server. Così facendo, se il server va offline, il client può scrivere direttamente il suo login/logout nel file di log. Così come il server può scrivere direttamente nei log di una chat (inserendo il segno '**' che indica che un messaggio è stato letto) quando recapita dei messaggi pendenti ad un utente *diventato online*.

AVVIO APPLICAZIONE

Il server e 3 device possono essere avviati in 2 modi:

- Modalità tradizionale → './exec2022.sh'
- Modalità debug → './exec2022.sh debug'

La *debug-mode* fornisce molte più stampe e dà accesso a 2 comandi aggiuntivi:

- 'registro' (server) → Stampa a video il registro del server
- 'rubrica <username>' (client) → Aggiunge *username* alla rubrica dell'utente corrente

FILES

Tutti i file qui riportati sono contenuti dentro la cartella del progetto.
I **file condivisi** di ogni utente si trovano nella cartella ‘shared/’.
Le **rubriche** di ogni utente si trovano nella cartella ‘rubriche/’.
I **log delle chat** di ogni coppia di utenti si trovano nella cartella ‘chat/’.
I **log di login/logout** di tutti gli utenti si trovano nel file ‘activity.txt’.
Le **credenziali** di tutti gli utenti si trovano nel file ‘users.txt’ .
Le **show** da notificare si trovano nel file ‘show_log.txt’ con il seguente formato: ‘<mittente messaggi>:<destinatario>’ (mittente è colui che deve ricevere la notifica).
I **messaggi pendenti** si trovano nel file ‘messaggi_offline.txt’ con il seguente formato: ‘<ricevente> list:<mittente>:<num. messaggi pendenti>:<timestamp messaggio recente>:’.

STRUTTURE DATI

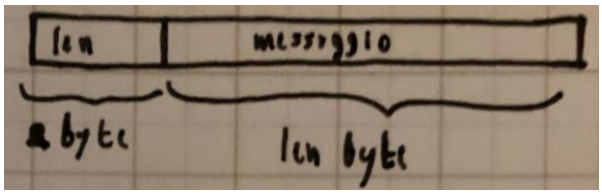
C’è solo una struttura dati utilizzata dal server: ‘record_registro’. Essa rappresenta le entry (username, password, timestamp di login/logout, porta del client) del registro del server.

MESSAGGI

Tutti i socket utilizzati sono **TCP**, perché in un’applicazione di messaggistica ritengo sia più importante l’affidabilità rispetto alla velocità.

Tutti i messaggi/comandi scambiati tra peer o tra client e server possono essere comodamente consultati nel file ‘costanti.h’.

Tutti i **messaggi** hanno uno stesso **formato**: i primi 2 byte contengono la lunghezza e i rimanenti il messaggio stesso.
Unica eccezione si ha con i messaggi che inviano interi: questi contengono solo il numero stesso.



Schema dei protocolli: