# Progetto Simple Public Ledger

## Descrizione sintetica:

Realizzare una applicazione che implementi un servizio distribuito di memorizzazione di coppie (chiave, valore). L'applicazione server viene eseguita contemporaneamente su più calcolatori. Il client può inviare richieste di "store" di una coppia e di "search" di un chiave ad uno qualsiasi dei server. Quando un server riceve una richiesta di memorizzazione, la inoltra a tutti i server attivi. Ogni server memorizza localmente l'informazione ricevuta. Alla ricezione di una richiesta di "search(chiave)", il server contatta gli altri server attivi, riceve le coppie (chiave, valore) da tutti i server contattati, invia al client il "valore" solo se tutte le coppie ricevute contengono lo stesso "valore". In caso contrario, invia al client il messaggio "Ledger corrotto"

L'applicazione deve essere scritta utilizzando il linguaggio C. I processi comunicano attraverso socket TCP. L'elaborato deve essere corredato da una opportuna documentazione.

## Descrizione dettagliata:

**Processo server**:

Il processo server riceve su linea di comando, il nome di un file di configurazione il numero di porta su cui mettersi in ascolto. Il file di configurazione contiene

- L'elenco degli indirizzi IP (che includono il numero di porta) che identificano gli altri server. Ogni riga contiene un solo indirizzo IP ed ha il seguente formato:
  - xxx.yyy.zzz.www:pppp

Ogni server associa un thread per la gestione delle comunicazioni con ogni altro server nel sistema. In fase di startup, il server crea una connessione con ogni altro server presente nel file di configurazione. Il sistema rimane bloccato fin quando tutte le connessioni richieste non siano operative. Si assuma che tutti i file di configurazione siano tra loro consistenti.
Terminata la fase di startup, il sistema transita in fase operativa.
Quando il server riceve una richiesta di connessione da parte di un client, viene creato un nuovo thread che ne gestisce la sessione. Il server accetta da ogni client **esattamente un comando.**

- Se il comando e' ***store(x,y)***:
  - Memorizza in una lista dinamica la coppia ***(x,y)*** solo se la chiave x non è già presente nella lista
  - Inoltra il comando a tutti i server attivi (ognuno dei quali memorizza la coppia ***(x,y)*** **senza** inoltrare ulteriormente il comando)
- Se il comando e' ***corrupt(x,z)***
  - Sostituisce la coppia ***(x,y)*** presente nella lista locale con la coppia ***(x,z).*** Se la lista non contiene una coppia ***(x,y)***, il client riceve un messaggio d'errore. Questo comando ***NON*** viene inoltrato.
- Se il comando e' ***search(x)***:
  - Ricerca nella lista locale una coppia con prima componente pari ad ***x***
    - Se ***(x,y)*** NON esiste nella lista locale, ritorna "Chiave Assente" al client
    - Se è presente la coppia ***(x,y)*** nella lista locale
      - Invia il comando ***search(x)*** a tutti i server
      - Riceve i messaggi $(x,y_1)\ldots(x,y_N)$ di ritorno dai server
      - Se $y= y_1=\ldots= y_N$, invia al client la coppia ***(x,y)***, altrimenti invia il messaggio "Ledger Corrotto"
- Se il comando è ***list***:

- Invia al client tutte le coppie presenti nella lista locale.

Si consideri che:
- Possono esistere più istanze del server in esecuzione sulla stessa macchina
- Il server può interagire concorrentemente con più client.

**Processo client**:
Il processo client **riceve comandi esclusivamente da linea di comando**.
La sintassi del client e' la seguente:

***client IPServer portaServer comando [par1] [par2]***

Il parametro comando corrisponde ai comandi interpretabili dal server e puo' assumere i valori "***store***", "***corrupt***", "***search***" e "***list***".

Il client, verifica la sintassi dei comandi. Se questa e' corretta, invia il comando al server indicato, visualizza la risposta ricevuta e termina.

## Regole generali.
Il server ed il client vanno realizzati in linguaggio C su piattaforma UNIX/Linux. Le comunicazioni tra client e server si svolgono tramite socket TCP. Oltre alle system call UNIX, i programmi possono utilizzare solo la libreria standard del C. Sarà valutato negativamente l'uso di primitive non coperte dal corso (ad es., code di messaggi) al posto di quelle studiate.

## Relazione

Il progetto va accompagnato da una relazione che contenga almeno le seguenti sezioni:

1. Una guida d'uso per il server e per il client, che illustri le modalità di compilazione e d'uso dei due programmi.
2. Una sezione che illustri il protocollo al livello di applicazione utilizzato nelle comunicazioni tra client e server (non il protocollo TCP/IP!).
3. Una sezione che descriva i dettagli implementativi giudicati più interessanti (con particolare riferimento alle system call oggetto del corso), eventualmente corredati dai corrispondenti frammenti di codice.
4. In appendice, la relazione deve riportare il codice sorgente integrale del progetto.

## Consegna del progetto
Entro due giorni dalla data prescelta per lo scritto finale, vanno consegnati al docente il progetto e la relazione. Il progetto va inviato all'indirizzo clemente.galdi@unina.it in un archivio in formato ***zip***. Durante la discussione del progetto, il client ed il server verranno testati, eseguendoli su due o più macchine diverse.