BIBLIO

Progetto di Laboratorio 2

AA 2022-2023

Corso A

Introduzione

Lo studente dovrà realizzare BIBLIO, un sistema bibliotecario distribuito, che permette agli utenti di ricercare e prendere in prestito dei libri.

Il sistema è costituito da più istanze dei seguenti processi:

- *bibclient*: il processo client (uno per ogni interrogazione) che si occupa di interagire con i server per ottenere informazioni sui volumi e richiederne il prestito;
- *bibserver*: il processo server che gestisce il catalogo di una sigola biblioteca, ovvero: verifica se un volume è presente nel catalogo, fornisce le informazioni relative al volume e (se possibile) ne registra il prestito.

I processi comunicano via socket. Lo studente può scegliere se usare socket UNIX o INET.

Inoltre, deve essere realizzato uno script bash (*bibaccess*) che analizza il file di log in cui ogni server registra le richieste processate.

I Server

Un bibserver viene attivato da shell con il comando

```
$ bibserver name_bib file_record W
```

dove

- `file_record` è il file *di testo* che contiene la registrazione delle schede relative a tutti i volumi della biblioteca `name_bib`. Un esempio di formato di tali record è il seguente

autore: Kernighan, Brian W.; autore: Ritchie, Dennis M.; titolo: Il linguaggio C (seconda Edizione); editore: Jackson Libri; anno: 1989; nota: Edizione italiana; collocazione: Z.22.56; luogo_pubblicazione: Milano; descrizione_fisica: 359 p., softcov, 13 cm;

I record sono separati da un'interlinea, e ogni record è costituito da un numero variabile di campi che descrivono autore/i, titolo, editore, anno, etc. Ogni campo ha il formato *nomecampo: valore;* in cui *nomecampo* è l'etichetta del campo (deve appartenere ad un insieme fissato) e *valore* è il valore associato a quell'etichetta. Un'etichetta può comparire più volte, nel caso in cui ci sono più valori da associare allo stesso *nomecampo* (e.g. libri con autori multipli). Le informazioni di un campo sono terminate da punto e virgola (;). I valori sono sempre stringhe di lunghezza limitata. Il campo *prestito* è presente solo nel caso in cui un volume è già in prestito, e il valore indica la data di scadenza del prestito. Nell'esempio sopra il libro è disponibile (campo *prestito* mancante), invece un esempio di volume in prestito conterrebbe anche il campo *prestito: 20-03-2023;.* Se la scadenza del prestito è passata, il libro si considera disponibile.

All'avvio del server vengono caricate le informazioni dal file con i record. Per ogni record esiste una sola copia in biblioteca.

- `name_bib` è il nome della biblioteca (a scelta dello studente)
- W è il numero di thread worker (vedi sotto)

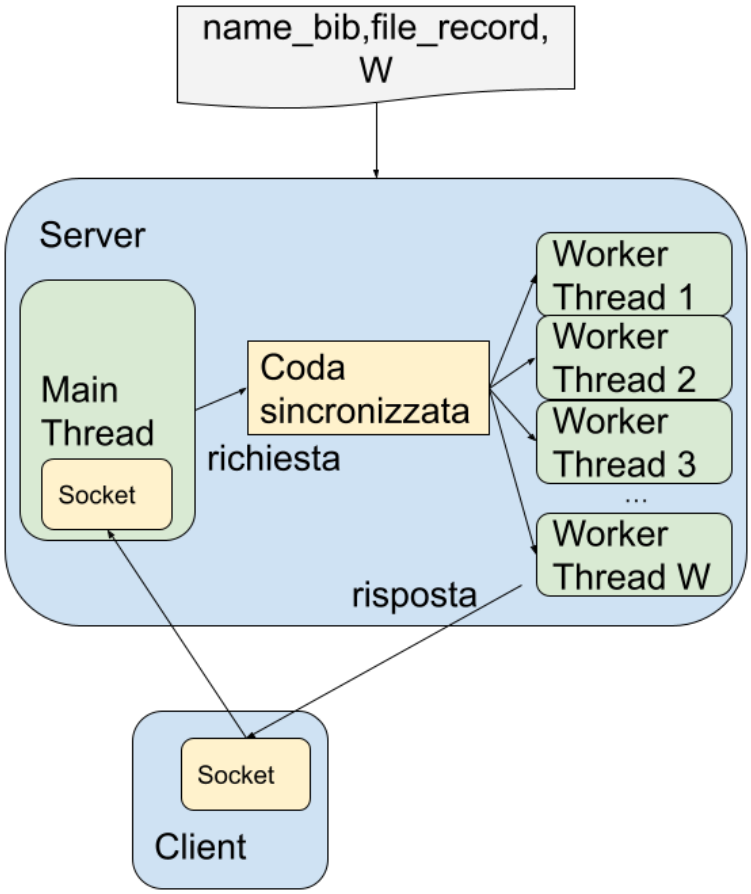
Tutti e tre parametri alla linea di comando sono obbligatori.

Il server registra un file di log *(./name_bib.log*) in cui per ogni richiesta effettuata da un client si registrano il numero dei record inviati, dei prestiti effettuati e le informazioni relative a ciascun record.

Il funzionamento del server è il seguente.

Se il file passato come argomento esiste viene aperto in lettura e ne viene verificato il formato. Se il formato è corretto viene creato il file di log (se il file esiste già viene troncato) e viene creata una socket. Su questa socket i client apriranno le connessioni con i server delle varie biblioteche per inviare le richieste di informazioni/prestito.

Il server è multithreaded e usa un numero fisso di thread worker (W), come decritto nello schema:



Ogni volta che arriva una richiesta sui socket, viene messa a disposizione dei worker usando una coda condivisa (produttore-consumatore). I worker prendono le richieste dalla coda e lavorano su una struttura condivisa che contiene le informazioni su tutte le schede contenute in `file_record`. Se il client richiede un prestito, questo viene concesso soltanto se il volume non è già stato prestato o se il prestito precedente è già scaduto. Ogni prestito dura 30 giorni (potete usare 30 secondi per testare il programma). I prestiti vengono registrati in RAM nella struttura condivisa fra i thread, aggiornando il campo "*prestito*". Alla chiusura, il server deve aggiornare `file_record` sovrascrivendolo con i record aggiornati della struttura condivisa. I libri per cui la scadenza del prestito è passata sono considerati disponibili e quindi il campo *prestito* non viene scritto nel file.

Ogni client invia una sola richiesta. Il thread main del server, per ogni richiesta, inserisce i dati nella coda condivisa. Il worker scandisce la struttura condivisa alla ricerca di uno o più record che verificano la richiesta effettuata ed eventualmente registrare il prestito. Le risposte alle richieste fatte verranno inviate al client sulla stessa socket di connessione.

Le richieste del client possono essere di due soli tipi. Una *query*, in cui si chiedono i record che contengono una specifica stringa in uno o più campi e un *loan* (prestito) in cui si richiede anche il prestito di tutti i volumi relativi a record che verificano la query.

Il server termina quando riceve un segnale di SIGINT o SIGTERM, in questo caso si attende la terminazione dei thread worker, in modo che vengano elaborate le richieste pendenti, si termina la scrittura del file di log, si registra il nuovo `file_record` (con le nuove date di prestito) e si elimina la socket del server. Infine il server viene terminato. Il protocollo di interazione fra client e server è descritto sotto.

La lista di tutti i server disponibili è inclusa in un file di configurazione *bib.conf,* che include tutti i nomi dei vari server attivi, e i dati necessari per la connessione ai rispettivi socket.

I Client

Il client *bibclient* viene attivato da linea di comando per avviare la ricerca delle informazioni relative ad un record che contiene particolari stringhe. Il formato di una invocazione è

```
$ ./bibclient --author="ciccio" -p

$ ./bibclient --author="ciccio" --title="pippo"
```

dove ogni opzione con doppio meno corrisponde al nome di un possibile campo del record, mentre l'opzione `-p` serve a richiedere il prestito. Per ogni invocazione è possibile specificare solo una opzione lunga (con doppio meno) per ogni campo. Il valore specificato dopo il simbolo = `e una stringa che deve essere contenuta nel campo specificato. E' obbligatorio inserire almeno un'opzione lunga, in caso contrario il programma stampa un messaggio di errore.

La risposta è negativa, se non esiste alcun record che contiene le stringhe specificate nei corrispondenti campi e positiva altrimenti. In caso di risposta positiva il client riceve dal server tutti i record che verificano le condizioni secondo il formato visto sopra nella descrizione del file di input per il server. Ogni record ricevuto è stampato su stdout dal client.

Se l'opzione `-p` è specificata viene richiesto il prestito di tutti i volumi che verificano le condizioni, a patto che non siano già in prestito presso altri utenti della biblioteca (campo prestito mancante o co una scadenza già passata). Il prestito viene registrato con durata pari a 30 giorni a partire dall'istante della interrogazione.

All'avvio, *bibclient* effettua il parsing delle opzioni e, se l'invocazione è corretta, legge il file *bib.conf* che contiene tutti i dati delle biblioteche disponibili per l'interrogazione (bib_name più dati relativi ai socket per ogni server). Poi interroga tutte le biblioteche connettendosi sulla socket e mandando una richiesta come descritto sotto.

Protocollo di comunicazione

I server ed i client interagiscono tramite un socket. I client si connettono al server di ogni biblioteca e mandano una richiesta. I server rispondono sulla stessa connessione.

Sia la richiesta che la risposta includono 3 campi: `type, length, data`. Il campo `type` è un char (8 bit) che contiene il tipo del messaggio spedito, e può assumere i seguenti valori:

#define MSG_QUERY 'Q'

#define MSG_LOAN 'L'

#define MSG_RECORD 'R'

#define MSG_NO 'N'

#define MSG_ERROR ’E’

Il campo `length` è un unsigned int che indica il numero dei dati significativi all’interno del campo `data`. Il campo `length` vale 0 nel caso in cui il campo `data` non sia significativo.

Il campo `data` è una stringa contenete il messaggio di risposta effettiva. Per ogni send/receive si richiede di inviare solo i byte significativi del messaggio e non un buffer di lunghezza fissa.

Messaggi da Client a Server

Nei messaggi spediti dal client al server, il campo `type` può assumere i seguenti valori:

- MSG_QUERY - Messaggio per richiedere i record che contengono alcune parole specifiche in alcuni campi. Il campo `data` contiene le condizioni da verificare secondo il formato campo1: valore1;... campoN: valoreN; Il server risponderà con un MSG RECORD se esiste almeno un record che soddisfa la richiesta e invierà i record completi in una serie successiva di messaggi di tipo MSG RECORD. Se nessun record soddisfa la query risponderà con MSG NO, mentre se si è verificato un errore risponderà con MSG ERROR ed una eventuale stringa che descrive l’errore.
- MSG_LOAN - Messaggio per richiedere il prestito di tutti i record che contengono alcune parole specifiche in alcuni campi. Funziona esattamente come MSG QUERY ma seleziona il prestito de volumi disponibili che verificano la query. In questo caso vengono inviati al client solamente i record per cui è stato accordato il prestito.

Messaggi da Server a Client

Nei messaggi spediti dal Server a Client, il campo type può assumere i seguenti valori:

- MSG ERROR Messaggio di errore. Questo tipo di messaggio viene spedito quando si è verificato un errore nel processare la richiesta del client. Il campo data può contenere una stringa che spiega l’errore verificatosi.
- MSG NO Messaggio di risposta negativa. Con questo messaggio il server segnala che non ci sono record che verificano la query.
- MSG RECORD Messaggio di invio record. Il campo buffer contiene il record completo come stringa correttamente terminata da ‘\0’.

File di log

Ogni richiesta processata da ogni server viene registrata all’interno di un file di log, con nome *name_bib.log*. All’inizio dell’elaborazione il server tronca il file (ne elimina eventuali contenuti precedenti) poi per ogni richiesta registra delle linee.

Per una richiesta di tipo MSG_QUERY registra delle linee del tipo

QUERY 0

se nessun record verifica la query oppure

autore: Luccio, Fabrizio; autore: Pagli, Linda; autore: Steel, Graham; titolo: M athematical and Algorithmic Foundations of the Internet; editore: CRC Press, Tay lor and Francis Group; luogo_pubblicazione: New York; anno: 2011; collocazione: Z.DDf.56;descrizione_fisica: 434 p., softcov, 22 cm;nota: Chapman & Hall/CRC Ap plied Algorithms and Data Structures series;

QUERY 1

ovvero tutti i record inviati, nel formato specificato, seguiti da una linea che contiene solo QUERY ed il numero totale di record inviati (nell'esempio sopra 1 risultato). Non è richiesto che tutte le linee relative ai record che corrispondono ad una singola query siano contigue nel file.

Il formato per una richiesta MSG_LOAN è analogo

autore: Luccio, Fabrizio; autore: Pagli, Linda; autore: Steel, Graham; titolo: Mathematical and Algorithmic Foundations of the Internet; editore: CRC Press, Tay lor and Francis Group; luogo_pubblicazione: New York; anno: 2011; prestito: 20-03-2023; collocazione: Z.DDf.56;descrizione_fisica: 434 p., softcov, 22 cm;nota: Chapman & Hall/CRC Applied Algorithms and Data Structures series;

LOAN 1

se è stato possibile accordare il prestito (notare la presenza della data di scadenza), oppure

LOAN 0

se non c'è stato prestito.

Script di controllo

bibaccess è uno script bash che elabora off-line i file di log generati dai bibserver. Lo script può essere invocato con due opzioni:

```
$ ./bibaccess --query log1 ... logN
```

```
$ ./bibaccess --loan log1 ... logN
```

dove `log1...logN` sono file di testo (ASCII) che contengono i log dei record prestati dai vari server, mentre con le opzioni precedute da doppio trattino (--query e –loan) si può richiedere il numero complessivo di richieste e di prestiti presenti nei file specificati. L'ordine di opzioni e parametri può essere qualsiasi.

Lo script deve controllare la validità dei suoi argomenti, scorrere i file di log ed individuare le informazioni richieste. Lo script stampa una riga per file di log ricevuto in input. Alla fine dell'elaborazione stampa le parole QUERY o LOAN e il loro valore complessivo (somma dei valori nel file di log)., e.g:

```
$ ./bibaccess --query bib1.log bib2.log bib3.log
```

bib1.log 5

bib2.log 7

bib3.log 0

QUERY 12

Makefile

Il progetto dovrà includere un makefile avente, fra gli altri, i target *all* (per generare tutti gli eseguibili), *clean* (per ripulire la directory di lavoro dai file generati), e *test.* Quest'ultimo deve eseguire un ciclo completo di test, lanciando 5 server con delle database diverse (eventualmente usando i file bib1.txt-bib5.txt messi a disposizione), e con un numero variabile di thread worker (da 1 a 5). Dopo un'attesa di 1 secondo dovrà lanciare un totale di 40 client, con richieste di tipo query e loan. I client andranno lanciati 5 alla volta, e con un'attesa di 1 secondo fra ogni gruppo.

Dopo aver lanciato l'ultimo gruppo, invierà un SIGINT al server. Dopo 10 secondi potrà lanciare lo script *bibaccess* sui cinque file di log.

Relazione

La documentazione del progetto consiste nei commenti al codice e in una breve relazione (massimo 5 pagine) il cui scopo è quello di descrivere la struttura complessiva del lavoro svolto. La relazione deve rendere comprensibile il lavoro svolto ad un estraneo, senza bisogno di leggere il codice se non per chiarire dettagli implementativi. In particolare, NON devono essere ripetute le specifiche contenute in questo documento. In pratica la relazione deve contenere:

- le principali scelte di progetto (strutture dati principali, algoritmi fondamentali e loro motivazioni)
- la strutturazione del codice (logica della divisione su più file, librerie etc.)
- la struttura dei programmi sviluppati
- la struttura dei programmi di test (se ce ne sono)
- le difficoltà incontrate e le soluzioni adottate
- quanto altro si ritiene essenziale alla comprensione del lavoro svolto
- README di istruzioni su come compilare/eseguire ed utilizzare il codice

La relazione deve essere in formato PDF.

Note finali

Nel realizzare il sistema oggetto del progetto, potete costruirvi le vostre strutture di dati o utilizzare alcune viste a lezione. Il protocollo di comunicazione può essere modificato se necessario, in quel caso va motivato nella relazione,

Ci si attende che tutto il codice sviluppato sia conforme POSIX e funzioni sulla macchina di laboratorio laboratorio2.di.unipi.it. Il codice che non compila o non esegue sulla macchina di laboratorio non verrà valutato.

La consegna dovrà avvenire, entro la data e ora dell'appello, attraverso Teams. Per ogni appello ci sarà un assignment nuovo aperto dove potete fare la consegna. Va consegnato un archivio con nome *nome_cognome*.tar.gz contenente tutto il necessario. Scompattando l'archivio in una directory vuota, dovrà essere possibile eseguire i comandi *make* (con target di default) per costruire tutti gli eseguibili, e *make test* per eseguire il test e vederne i risultati. L'archivio dovrà anche contenere la relazione.