Univerisita' degli Studi di Trento

Sistemi Operativi

AA 2014

CODEC

Autori:
Andrea Ghizzoni
Federica Lago

1 Analisi del Progetto

Il progetto scelto deve essere sviluppato solo per sistemi operativi Linux e consiste nello scrivere un programma client che può richiedere ad un programma server di elencare la lista dei messaggi da lui codificati oppure di codificare o decodificare un testo scelto dall'utente. La codifica e la decodifica avvengono tramite traslitterazione sull'alfabeto inglese, attraverso una determinata chiave fornita dall'utente. Come indicato nel testo il client e il server devono poter accettare determinati parametri dalla shell: il nome del server a cui il client deve collegarsi, il nome (univoco) del server da lanciare, nome del file o messaggio da inviare al server per essere elaborato, ecc..

È possibile utilizzare tutte le funzioni che il kernel Linux mette a disposizione, tranne ovviamente la chiamata di sistema system(), la quale darebbe pieno accesso ad una serie di comandi già presenti. In aggiunta non è possibile utilizzare il meccanismo di comunicazione dei socket in favore di PIPE e FIFO.

La compilazione di tutto il progetto deve avvenire tramite makefile e produrrà due binari all'interno alla cartella bin rispettivamente per il client e per il server.

Da testo il server deve poter rispondere alle richieste di più client, quindi sarà necessario che non sia lui direttamente a soddisfare tutte le richieste (intese come codifica/decodifica/lista dei messaggi codificati), ma che una terza entità prenda in carico la richiesta del client e lo serva.

2 Analisi della Soluzione

Prima di presentare la soluzione riportiamo in seguito la gerarchia delle cartelle del progetto:

- /src contiene tutti i sorgenti necessari
  - /server contiene tutti i file per compilare il server
  - /client contiene tutti i file per compilare il client
  - /util contiene tutte le utility necessarie sia al client che al server.
  - /assets contiene i sorgenti del programma generatore degli assets
- /assets contiene degli esempi di input file generati automaticamente dal comando make assets
- /bin contiene i binari del server e del client dopo il comando make bin

Di seguito verrà elencata la nostra soluzione distinguendo tra makefile, client, server e comunicazione.

Makefile

- sono presenti tutti i target richiesti nel testo del progetto (bin, res, test ecc..)
- il compilatore gcc e' stato messo in condizioni di evidenziare più errori possibili e in modo da seguire lo standard ANSI C:
  - Werror: tutti i warning vengono segnati come errori
  - Wall: visualizza tutti i messaggi di warning
  - Wunreachable-code: visualizza un warning quando trova una variabile non usata
  - ansi: utilizza lo standard ANSI C
  - pedantic: il compilatore visualizza più warning del dovuto
  - O2: flag di ottimizzazione del compilatore
- il target bin non eseguirà direttamente la compilazione del client e del server, ma richiamerà i makefile dei rispettivi programmi; i quali compileranno tutte le dipendenze e poi il programma stesso. Ad esempio: nella cartella src/server è presente un makefile con tre target: description, dep e server: la prima semplicemente stampa una descrizione del makefile, la seconda compila tutte le dipendenze del server contenute nella cartella include e, infine, il target server compila appunto il server con tutte le dipendenze.

Client
Il client deve poter svolgere le seguenti operazioni

- accettare i parametri dell'utente (nome del server, operazione richiesta, eventuale messaggio o file di input, chiave, eventuale file di output)
- inviarle i suddetti parametri al server per poterli elaborare e predisporsi per la ricezione dei risultati.

Al client non è richiesto di rimanere in esecuzione una volta che ha ricevuto i dati elaborati, ma eventualmente solo di stamparli a console.

Server
Il Server deve rispettare determinate caratteristiche:

- deve essere etichettato con un nome.
- deve rimanere in attesa della connessione di un client
- deve poter accettare la connessione attraverso un messaggio specifico
- deve poter ricevere i dati dal client per poter procedere all'elaborazione

- in caso di un errore inatteso (dati in ingresso malformati, errore nel canale di comunicazione, ecc..) deve produrre un errore opportuno e comunicarlo al client
- l'unico caso di terminazione del server è attraverso i segnali di terminazione dei processi del sistema

Comunicazione

- Per gestire i parametri della shell abbiamo scelto di utilizzare la funzione getop long, con la possibilità di utilizzare sia nomi lunghi che brevi, ad esempio per definire il nome del server si può usare –name oppure -n.
- Per gestire la comunicazione abbiamo scelto di usare delle FIFO, per avere la possibilità di accedere ai processi senza che client e server debbano per forza essere in una relazione di parentela.
- Per la comunicazione vera e propria abbiamo stabilito un nostro protocollo, che prevede che entrambi gli estremi della FIFO attendano un primo messaggio nel quale sono indicati i numero di byte del messaggio vero e proprio. Nel dettaglio:
  - Il client invia al server le dimensione della richiesta che vuole inviare.
  - Il client invia al server il messaggio strutturato nel seguente modo (da notare che non tutti i campi sono obbligatori):

    client's pid | mode;key | input type ; input | output type; output

    In seguito si mette in attesa della risposta del client.
  - il server, dopo aver eseguito l'operazione richiesta dal client, gli invia un messaggio contenente la dimensione del messaggio di risposta. E poi procede all'invio del messaggio vero e proprio. Nel caso in cui il server debba inviare più messaggio di risposta, come nel caso dell'operazione list, ognuno di essi dovrà essere preceduto dalla sua dimensione.