Univerisita' degli Studi di Trento

Sistemi Operativi

AA 2014

CODEC

Autori:
Andrea Ghizzoni
Federica Lago

1 Analisi del Progetto

Il progetto scelto, sviluppato per sistemi operativi Linux, 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 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.
  - /flags contiene alcuni flag utilizzati nei programmi (es: flag modalità debug)
  - /res contiene i sorgenti del programma generatore degli res
- /res contiene degli esempi di input file generati automaticamente dal comando make res
- /bin contiene i binari del server e del client dopo il comando make bin

Per gestire i parametri della shell abbiamo scelto di utilizzare la funzione getopt long, con la possibilità di utilizzare sia nomi lunghi che brevi, riportati nelle relative sezioni client e server. Di seguito verrà elencata la nostra soluzione distinguendo tra makefile, client, server e comunicazione.

Makefile

Sono presenti, oltre ad una breve descrizione iniziale, tutti i target richiesti nel testo del progetto:

- make: senza altri parametri genera una breve descrizione del progetto e degli altri target disponibili.
- bin: bin, deve generare i binari compilati eseguibili. Questo target 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. Inoltre richiama il target res.
- res: compila ed esegue il programma che genera automaticamente dei possibili messaggi di input. Le parole generate non appartengono ad una lingua, ma servono solo per testare il funzionamento delle funzioni encode e decode.
- test: richiama i target bin ed res ed avvia l'esecuzione del programma.
- clear: elimina ogni file automaticamente generato o temporaneo.

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

Client

Il client accentta i seguenti parametri

OPZIONE	SHORTCUT	NOTE
- -name	-n	Indica il nome del server a cui collegarsi. Campo obbligatorio.
- -key	-k	Deve essere composto da sole lettere. Obbligatorio in caso di encode o decode.
- -file - -message	-f -m	Indica se l'input sarà da console o da file. Uno dei due è obbligatorio in caso di encode o decode.
- -output	-o	Indica che l'output è su file. Se non è seguito da un argomento verrà creato un file di default. Se manca l'output avverà a video.
- -encode - -decode - -list	-e -d -l	Indica il tipo di servizio richiesto al server. Una di queste opzioni obbligatoria.

Il client svolge le seguenti operazioni

- accetta i parametri sopra elencati dall'utente;
- invia i parametri al server per poterli elaborare;
- si predispone per la ricezione dei risultati, attraverso un loop;
- nel caso in cui l'output sia a video, stampa quanto ritornato dal server.

Al client non è richiesto di rimanere in esecuzione una volta che ha ricevuto i dati elaborati e quindi viene terminato una volta svolto il suo compito.

Server

Il server accetta i seguenti parametri:

OPZIONE	SHORTCUT	NOTE
- -name	-n	Indica il nome, che deve essere univoco, del server da lanciare. Campo obbligatorio.
- -msgmax	-M	Contiene la dimensione massima dei messaggi che possono essere codificati/decodificati su quel server.
- -keymin	-k	Contiene la dimensione minima delle chiavi che possono essere usate su quel server.
- -keymax	-K	Contiene la dimensione minima delle chiavi che possono essere usate su quel server.

Il server, una volta lanciato con i parametri sopra citati, svolge le seguenti operazioni:

- rimane in attesa della connessione da parte di un client;
- riceve i dati dal client per poter soddisfare la sua richiesta;
- passa i dati del client ad un thread che procede con l'elaborazione dei dati e manda i risultati al client. In caso di un errore inatteso (dati in ingresso malformati, errore nel canale di comunicazione, ecc..) produce un errore opportuno e lo comunica al client.

L'unico caso di terminazione del server è attraverso i segnali di terminazione dei processi del sistema.

Comunicazione

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