Programmazione in ambienti distribuiti I

(01FQT, 01FEL, 06EKE, 02HDF)

AA 2009-2010, Esercitazione di laboratorio n. 1

Esercizio 1.1 (connessione)

Scrivere un client che si colleghi ad un server TCP all'indirizzo e porta specificati come primo e secondo parametro sulla riga di comando e quindi termini chiudendo correttamente il canale e segnalando se il collegamento è riuscito o fallito.

Esercizio 1.2 (dati ASCII)

Sviluppare un client che si colleghi ad un server TCP all'indirizzo e porta specificati come primo e secondo parametro sulla riga di comando e quindi invii due numeri interi senza segno (rappresentabili in binario puro su 16 bit) letti da standard input e riceva la risposta (somma, o errore) dal server.

La somma è un numero intero senza segno a 16 bit, ma il server gestisce anche il caso di overflow, che è segnalato con uno specifico errore (si veda l'esempio).

Il client invia al server i due numeri in notazione decimale espressi mediante caratteri numerici ASCII. I numeri devono essere separati da un solo spazio e la trasmissione deve essere terminata con CR-LF. Il server restituisce il risultato (un solo numero) espresso mediante caratteri ASCII, privo di spazi e terminato con CR-LF oppure un messaggio di errore (caratteri ASCII) terminato anch'esso da CR-LF.

Esempi (ogni riquadro rappresenta un singolo byte trasmesso in codice ASCII):

(client → server) `1` `2` `3` `4` `5` ` ` `3` `CR` `LF`

(server → client) `1` `2` `3` `4` `8` `CR` `LF`

o in caso di errore:

(server → client) `o` `v` `e` `r` `f` `l` `o` `w` `CR` `LF`

(server → client) `i` `n` `c` `o` `r` `r` `e` `c` `t` ` ` `o` `p` `e` `r` `a` `n` `d` `s` `CR` `LF`

Esercizio 1.3 (dati binari)

Sviluppare un client che si colleghi ad un server TCP all'indirizzo e porta specificati come primo e secondo parametro sulla riga di comando e quindi invii due numeri interi senza segno (rappresentabili in binario puro su 16 bit) letti da standard input e riceva la risposta (somma o errore) dal server.

Il client invia al server i due numeri in formato binario (network order): prima i due byte del primo operando e poi i due byte del secondo operando.

Il server restituisce tre byte, di cui il primo contiene lo stato (esito dell'operazione) mentre il secondo e il terzo byte rappresentano il valore del risultato (network order). Il byte di stato può assumere i seguenti valori binari:

- 0 significa che l'operazione è stata svolta correttamente;
- 1 significa overflow (in questo caso il secondo ed il terzo byte sono entrambi uguali a zero)

Esercizio 1.4 (dati binari tagged)

Sviluppare un client che si colleghi ad un server TCP all'indirizzo e porta specificati come primo e secondo parametro sulla riga di comando e quindi invii due numeri interi senza segno

(rappresentabili in binario puro su 8 o 16 bit, letti da standard input) e riceva la risposta (somma o errore) dal server.

Lo scambio dati avviene con numeri in binario puro tagged (un byte per esprimere in binario puro la dimensione dei dati, poi uno o due byte per dato a seconda che siano a 8 o 16 bit, sempre in network order).

Il client deve inviare al server tre o cinque byte di cui il primo contiene il valore binario della lunghezza di ciascuno degli operandi che seguono espressa in numero di byte (1 o 2). I byte successivi (rispettivamente due o quattro) contengono i valori interi senza segno su 8 o 16 bit dei due operandi.

Il server restituisce due o tre byte, di cui il primo contiene lo stato (esito dell'operazione, come nell'esercizio 1.3) ed i seguenti il valore del risultato (stessa lunghezza e codifica degli operandi inviati dal client).

Server di prova

Per provare i client viene fornito un server di prova (sum_tcps.c) che risponde alla porta 1500/tcp.

Per compilarlo:

```
gcc –o sum_tcps –DTRACE sum_tcps.c errlib.c sockwrap.c
```

Per avviarlo:

```
./sum_tcps protocollo
```

ove *protocollo* è una stringa che specifica il protocollo usato nel dialogo col client):

- **`-a`** per il protocollo ASCII
- **`-b`** per il protocollo binario fisso
- **`-t`** per il protocollo binario tagged