

Uso dei socket

Programmazione in Ambienti
Distribuiti

marco.torchiano @polito.it

© 2006, Marco Torchiano

Materiale ed Esercizi

- Web Page
 - ♦ <http://softeng.polito.it/05EKE>
 - Contenuti
 - ♦ Testo degli esercizi
 - ♦ Soluzioni
 - ♦ Esempi
-

Socket

- Asimmetrici nella connessione
 - ◆ Server
 - ◆ Client
 - Simmetrici nella comunicazione
 - ◆ `recv()`
 - ◆ `send()`
-

Lato server

- Creazione socket
 - ◆ `socket()`
 - Bind ad un indirizzo locale
 - ◆ `bind(.. struct sockaddr* ..)`
 - Ascolto
 - ◆ `listen()`
 - Accettazione connessioni
 - ◆ `accept(.. struct sockaddr * ..)`
-

Lato client

- Creazione socket
 - ◆ `socket()`
 - Connessione
 - ◆ `connect(..struct sockaddr *..)`
-

Prove in locale

- Conviene fare le prove in locale, lanciando sia client che server sulla stessa macchina
 - Usate porte > 1024
 - La macchina locale si chiama
 - ◆ `127.0.0.1`
 - ◆ `localhost`
-

Errori tipici

- Mancata conversione da *Host order* a *Net order* o viceversa
 - ♦ Prima si ricava l'indirizzo in H.O.
 - `addr_host = 0x82C00351; //130.192.3.81`
 - ♦ Poi si converte in N.O.
 - `addr_net = htonl (a_host);`
 - ♦ Alcuni metodi sono già in N.O.
 - `addr_n = inet_addr ("130.192.3.81");`
 - ♦ Stesso discorso per le porte
 - Si usano `htons` e `ntohs` (short int)
-

Errori Tipici

- Lettura di stringhe senza terminatore
 - ♦ Soluzione 1
 - `memset (buffer, 0, sizeof (buffer));`
 - `r=recv(s, buffer, sizeof (buffer)-1, 0);`
 - ♦ Soluzione 2
 - `r=recv(s, buffer, sizeof (buffer)-1, 0);`
 - `rc[r]='\0';`
 - ♦ Funziona con buffer `char[]`, non con `char*`
-

Errori Tipici

- Il server non invia niente al client
 - ♦ Verificate di utilizzare il socket restituito dalla funzione **accept()** e non il socket server creato inizialmente con la funzione **socket()**
 - Il campo **sin_addr.s_addr** della struttura passata a **bind()** è una maschera e non un indirizzo
 - ♦ Di solito si usa **INADDR_ANY** che equivale a "*" cioè accetta connessioni da tutti
-

Attenzione

- I simboli **SOCKET** e **INVALID_SOCKET** non sono definiti.
 - Potete aggiungere le seguenti definizioni
 - ♦ **#define SOCKET int**
 - ♦ **#define INVALID_SOCKET -1**
-

File header

```
#include <unistd.h>
    /*funz socket, es. close()*/
#include <stdlib.h>
    /* atoi(), exit() */
#include <string.h>
    /* memset() */
#include <sys/socket.h>
    /* socket() */
#include <netinet/in.h>
    /* rete internet */
#include <arpa/inet.h>
    /* funzioni di utilità es. inet_addr() */
```

Tempo di sistema

- Per ottenere il tempo di sistema in formato stringa:

```
time_t t;
```

```
char* ts;
```

```
t = time(NULL);
```

```
ts = ctime(&t);
```

Select

▪ int select(

- ♦ int *n*,
- ♦ fd_set **readfds*,
- ♦ fd_set **writefds*,
- ♦ fd_set **exceptfds*,
- ♦ struct timeval **timeout*);

= numero del descrittore piu' alto + 1

NULL → blocking
N.B. Dopo la chiamata la struttura dati viene modificata!!

Esempio Select

```
char buf[1024];
fd_set rfd;
struct timeval tv;
int retval;
FD_ZERO(&rfd);
FD_SET(0, &rfd);
tv.tv_sec = 1; tv.tv_usec = 0;
retval = select(1, &rfd, NULL, NULL, &tv);
if (FD_ISSET(0, &rfd)){
    memset(buf, 0, sizeof(buf));
    read(0, buf, sizeof(buf));
    printf("[%s]\n", buf);
}
```

Aggiunge lo standard input (0)

Aspetta per un secondo