INTRODUZIONE ALL'INTERFACCIA SOCKET DI BERKELEY

E.Mumolo

IL MODELLO CLIENT/SERVER

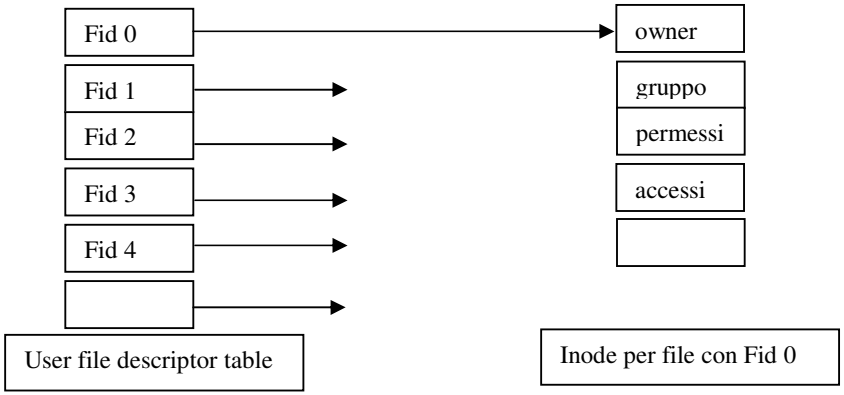
- NB : TCP/IP = famiglia di protocolli
- Il modello client/server e' alla base della maggior parte delle applicazioni distribute
- Semplice soluzione al problema del rendez vous
- Tipi di interazione : NON orientato alla connessione (UDP)/Orientato alla connessionte (TCP)
- TCP garantisce affidabilita':
    1. verifica che i dati arrivino,
    2. ritrasmette automaticamente segmenti
    3. calcola checksum
    4. ricostruisce l'ordine dei pacchetti
    5. elimina duplicazioni
    6. realizza controllo di flusso
    7. informa se la rete non e' piu' utilizzabile
- UDP fornisce un servizio di consegna 'best effort': funziona bene se la rete sottostante funziona bene (es.LAN)
- Tipi di server : Privo di stato/con stato (esempio : file server)

L'ASTRAZIONE SOCKET

- Astrazione file :

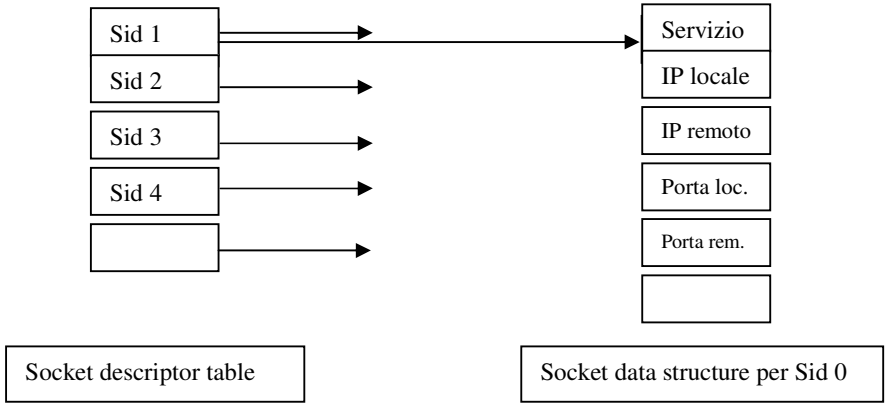
File description table, file identifier 'fid' → struttura dati interna per il file "fid", diversa per ogni processo

Creazione di fid: fid=open("nome",FLAG). Rappresentazione schematica:



- Astrazione per la comunicazione di rete :

Socket description table, socket identifier'sid' → struttura dati interna per la comunicazione "sid", diversa per ogni processo



Osservazioni:

- Cose da specificare:
  1. famiglia di protocolli,
  2. tipo di protocollo della famiglia,
  3. indirizzo locale/remoto,
  4. porta locale/remoto,
  5. servizio di trasporto (connesso/non connesso)
- Un cliente ha bisogno solo dell'indirizzo remoto e porta remota
- Un cliente deve essere attivo, nel senso che deve specificare l'indirizzo e la porta con la quale vuole parlare
- Un server ha bisogno di sapere qual'e' il suo indirizzo IP e il nr. di porta; viene poi messo in modo passivo nel quale aspetta a ricevere richieste
- Chiamate di sistema principali:
  - **socket** **crea un punto di comunicazione; specifica la famiglia di protocolli, il protocollo delle famiglia, il tipo di comunicazione**
  - **connect** **stabilisce una connessione attiva con il server remoto; specifica l'indirizzo IP e porta remoti, sceglie IP e porta locali**
  - **bind** **specifica indirizzo e porta locale da associare ad un socket**
  - **listen** **pone il socket in modo passivo**
  - **accept** **estrae una nuova richiesta dall coda delle richieste**
  - **read, write, close** **leggono e scrivono dati, chiusura di un socket**
  - **sendto, recfrom** **scrivono e leggono dati specificando l'indirizzo IP**
- Struttura client/server in comunicazione orientata alla connessione:

  Cliente: socket → connect → write → read → close

  Server: socket → bind → listen → accept → read → write → close
- Struttura client/server in comunicazione NON orientata alla connessione:

  Cliente: socket → bind → sendto → read → close

  Server: socket → bind → recfrom → sendto → close

- Su una macchina ci sono moltissimi server attivi. Il comando netstat [-a] visualizza connessioni e servizi.
- Esempio:

```
ingsun1/home/mumolo/net/cli-ser $ netstat
TCP
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
localhost.32794      localhost.32772      32768      0 32768      0 CLOSE_WAIT
ingsun1.54345        presing01.univ.trieste.it.6000 17184      0  8760      0 ESTABLISHED
ingsun1.54347        presing01.univ.trieste.it.6000 17340      0  8760      0 ESTABLISHED
ingsun1.1023         ingsun2.login         8760      0  8760      0 ESTABLISHED
localhost.61388      localhost.61386      32768      0 32768      0 ESTABLISHED
localhost.32893      localhost.32892      32768      0 32768      0 ESTABLISHED
localhost.32892      localhost.32893      32768      0 32768      0 ESTABLISHED
ingsun1.telnet       enepc32.univ.trieste.it.1445  8585      0 10192      0 ESTABLISHED
ingsun1.1698         uts.univ.trieste.it.2267 33580      0  8760      0 ESTABLISHED
ingsun1.nfsd         ingsun21.586          8760      0  8760      0 ESTABLISHED
ingsun1.42081        ingsun1.32772        32768      0 32768      0 CLOSE_WAIT
ingsun1.42109        ingsun1.32772        32768      0 32768      0 CLOSE_WAIT
ingsun1.nfsd         ingsun15.ufsd         8760      0  8760      0 ESTABLISHED
ingsun1.telnet       caesar.univ.trieste.it.1178  8743      0  8760      0 ESTABLISHED
ingsun1.telnet       caesar.univ.trieste.it.1687  8698      0  8760      0 ESTABLISHED
```

- Alcuni servizi TCP/IP:
  - echo (porta 7)
  - users (porta 11) utenti attivi
  - daytime (porta 13) ora del giorno
  - chargen (porta 19) generatore di caratteri
  - ftp (porta 21)
  - telnet (porta 23)
  - smtp (25) simple mail transport protocol
  - name (porta 42) server di nomi dell'host
  - nameserver (porta 53) server di nomi di dominio
  - finger (porta 79) nntp (porta 119) server di news

SPECIFICA DEGLI INDIRIZZI NELLE COMUNICAZIONI DI RETE

- Gli indirizzi dei due endpoint devono essere definiti (endpoint=punto di comunicazione)
- Le famiglie di protocolli definiscono gli indirizzi in modi diversi
- Per garantire generalita' vengono definite famiglie di indirizzi per ogni tipo di protocollo:

famiglie di protocolli → famiglie (tipo) di indirizzi

Es. TCP/IP, dominio internet : PF_INET (=2) → solo 1 tipo di indirizzi, chiamato AF_INET (=2)

Es. dominio Unix: PF_UNIX(=1) → AF_UNIX (=1)

- Struttura generale degli endpoint : (famiglia di indirizzi, indirizzo endpoint). In TCP/IP: (famiglia, IP, nr.porta protocollo)

Struttura dati predefinite :

```
struct sockaddr{
     u_short   sa_family;     //famiglia di indirizzi, determina il formato degli ottetti
                              //restanti per es. AF_INET(internet), AF_NS(Xerox), AF_UNIX
     char      sa_data[14];   //valore dell'indirizzo
}
```

bit 0 bit 31

Famiglia di indirizzi	Ottetti 0-1
Ottetti 2-3	Ottetti 4-5
Ottetti 6-7	Ottetti 8-9
Ottetti 10-11	Ottetti 12-13

- Solo per gli indirizzi TCP/IP, gli indirizzi si specificano con la seguente :

```
struct sockaddr_in{
     u_short            sin_family;   //famiglia di indirizzi: DEVE essere AF_INET ! !
     u_short            sin_port;     //numero di protocollo
     struct  in_addr    sin_addr;     //indirizzo IP remoto, generalmente u_long
     char               sin_zero[8];  //non usato
}
```

dove `struct in_addr{u_long s_addr;}`

bit 0 bit 31

Famiglia di indirizzi	Porta di protocollo
Indirizzo IP	
Ottetti 6-7	Ottetti 8-9
Ottetti 10-11	Ottetti 12-13

Tipicamente sockaddr_in viene sovrapposto a sockaddr tramite casting. Esempio:

```
struct sockaddr_in skaddr;
connect(sk, (struct sockaddr *) &skaddr,sizeof(skaddr))
```

- Invio dei dati tramite un socket:
  - socket connessi → chiamate di sistema write, writev, send
  - socket non connessi → sendto(specifica l'indirizzo di dest.tramite sockaddr_in), sendmsg(usa una struttura)
- Ricezione dei dati tramite socket:
  - Socket connesso → read, readv, recv
  - Socket non connesso → recvfrom, recvmsg (specificano l'indirizzo di provenienza)

- Alcune procedure utili

  htons() ritorna l'argomento intero nell'ordine richiesto dalla rete
  inet_addr() converte la notazione decimale di IP in numero

- Un esempio di definizione

```
struct sockaddr_in sin;
sin.sin_family = AF_INET;
sin.sin_port = htons(9999);
sin.sin_addr.s_addr = inet_addr("140.105.50.38");
```

- **ALCUNE CHIAMATE DI SISTEMA**

```
#include <sys/types.h>
#include <sys/socket.h>
```
int socket(int famiglia, int tipo, int ptotocollo)

ritorna un descrittore numerico o –1. Nella 5-tupla che definisce una connessione, socket specifica il protocollo
Famiglia di protoc.:

PF_UNIX	protocolli interni a Unix
PF_INET	protocolli di Internet
PF_NS	protocolli di Xerox
PF_IMPLINK	collegamento ad un IMP (Interface Message Processor)

Tipo, protocollo

SOCK_STREAM, IPPROTO_TCP	→ usa TCP
SOCK_DGRAM, IPPROTO_UDP	→ usa UDP
SOCK_RAW, IPPROTO_RAW	→ usa un protocollo Rudimentale

Normalmente, si puo' mettere protocollo=0
#include <sys/types.h>
#include <sys/socket.h>
int bind(int sockfd, struct sockaddr *mioindirizzo, int lunghdir);

Definisce gli elementi indirizzo-locale, processo locale nella 5-tupla: assegna un nome ad un socket senza nome.
Ci sono tre possibili usi di bind:
1) un server registra il suo indirizzo nel sistema → qualsiasi indirizzo a questo sistema deve essere consegnato a me
2) un cliente registra un indirizzo per se' stesso
3) un cliente senza connessione deve accertarsi che il sistema assegni ad esso un indirizzo unico, affinche' il server sappia dove mandare la risposta

#include <sys/types.h>
#include <sys/socket.h>
int connect(int sockfd, struct sockaddr *indirserver, int lunghindir);

Attiva la connessione tra il sistema locale ed il sistema estero: vengono scambiati dei messaggi tra i due sistemi.

Connect non ritorna finche' non viene stabilita la connessione (o non ritorna un errore)

int listen(int sockfd, int backlog);

Usata da un server per comunicare che e' disposto ad accettare connessioni. Backlog indica il umero di richieste di connessione che possono essere accodate.

#include <sys/types.h>
#include <sys/socket.h>
int accept(int sockfd, struct sockaddr *pari, int *lunghindir);

- prende la prima richiesta in coda
- crea un altro socket
- se non ci sono richieste in sospeso, si blocca in attesa
- restituisce un codice di ritorno intero che puo' essere:
    - descrittore del nuovo socket
    - indirizzo del processo alla pari
    - dimensione di questo indirizzo
    - eventualmente, indicazione d'errore

- Server di tipo concorrente/non concorrente:

1) concorrente

```
newsockfd=accept(sockfd,…);
if(fork() == 0){ close(sockfd); elabora(newsockfd); exit(0); }
```

2) iterativo

```
for(;;){
        newsockfd=accept(); elabora(newsokfd);
        close(newsockfd);
    }
```

- send, sendto, recv, recfrom

Inviano o ritornano un datagramma e la sua lunghezza

Esempio:

```
#include <sys/types.h>
#include <sys/socket.h>

int sendto(int sockfd, char *buf, int nbytes, int flags,
           struct sockaddr *to, int lunghdindir);
```

- close

```
int close(int fd);
```

- routine di ordinamento dei byte

```
unsigned long htonl(unsigned long hostlungo); /* converte da host a rete */
unsigned short htons(unsigned short hostcorto); /*  da host a rete */
unsigned long ntohl(unsigned long retelungo); /* converte da rete ad host*/
unsigned short ntohs(unsigned short retecorto); /*  da rete ad host */
```

- operazioni di byte

```
bcopy(char prov, char dest, int nbytes);        /* trasferisce il numero specificato di byte da prov a dest */
bzero(char dest, int nbytes);      /* scrive il numero specificato di byte nulli */
int bcmp(char ptr1, char ptr2, int nbytes);      /* confronta due stringhe */
```

- conversione di indirizzi da notazione decimale a numero e viceversa

```
unsigned long inet_addr(char *ptr);
char *inet_ntoa(struct in_addr inaddr);
```

- Porte:
  - riservate da 0 a 1023
  - assegnate automaticamente dal sistema specificando porta =0 da 1024 a 5000
  - porte richieste dall'utente > 5000

  - Possibili problemi con la definizione degli indirizzi. Es. : sa_family=AF_UNIX ; sa_data= "nome del file maggiore di 14 caratteri" ; //AF_UNIX e' una pipe con nome

  - Nota sulla compilazione: le librerie di socket (da includere nella compilazione) sono: **-lsocket –lnsl**
  - Nel caso di ambienti Windows (es: GygWin) la libreria e' winsock: **-l wsock32**
  - Alcuni include files:

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
```

- In Windows i socket vengono usati mediante la libreria WINSOCK:

```
// client.cpp
#include <winsock.h>
SOCKET      ClientSocket;

ClientSocket = socket (PF_INET,SOCK_STREAM,0);
if (ClientSocket==INVALID_SOCKET) ErrorStop("socket()");
IndirizzoServerSocket.sin_family      = AF_INET;
IndirizzoServerSocket.sin_addr.S_un.S_addr=inet_addr("127.0.0.1");
IndirizzoServerSocket.sin_port=htons(IPPORT_TELNET);

rv = connect(ClientSocket,(structsockaddr*)&IndirizzoServerSocket,
            sizeof(IndirizzoServerSocket));
if (rv!=0) ErrorStop("connect()");
...
```

Esempio: codice **server AF_UNIX**

```
#include <sys/types.h>
#include <sys/socket.h>
main()
{
     int sd, ns;
     char buf[256];
     struct sockaddr addr, addr1;
     int fromlen=256, i=0;
     sd = socket(AF_UNIX, SOCK_STREAM, 0);
     if(sd==-1) syserr("socket");
     else printf("socket id=%d\n",sd);
     addr.sa_family=AF_UNIX;
     memcpy(addr.sa_data,"nomesocket\0",11);
     if(bind(sd, &addr, sizeof(addr) )==-1)
        sysserr("bind");
     else
        printf("realizzato il binding con
                ìl'indirizzo locale\n");
     if(listen(sd,1)==-1) syserr("listen");
     else printf("dopo la listen\n");

     for(i=0;i<4;i++)
     {
          ns = accept(sd, &addr1, &fromlen);
          if(ns == -1) syserr("errore in
                               accept");
          read(ns, buf, sizeof(buf));
          printf("Sono il server. Ho
                  ricevuto:%s\n",buf);
          close(ns);
     }
     unlink("nomesocket"); exit(0);
}
```

Esempio: codice **client AF_UNIX**

```
#include <sys/types.h>
#include <sys/socket.h>

main()
{
     int sd, ns;
     char buf[256];
     struct sockaddr addr;
     int fromlen;

     sd=socket(AF_UNIX, SOCK_STREAM, 0);
     if(sd == -1) syserr("socket");
     addr.sa_family=AF_UNIX;
     memcpy(addr.sa_data,"nomesocket\0",11);

     if(connect(sd, &addr, sizeof(addr)) == -1)
           syserr("connect");

     write(sd,"ciao ciao ciao", 14);
}
```

In pratica:

```
ingsun1/home/mumolo $ cc -o s server.c syserr.c
server.c:
syserr.c:
Linking:
ingsun1/home/mumolo $ cc -o c client.c syserr.c
client.c:
syserr.c:
Linking:
ingsun1/home/mumolo $ s&
[1]     4321
ingsun1/home/mumolo $ socket id=3
realizzato il binding con l'indirizzo locale
dopo la listen

ingsun1/home/mumolo $ ps
   PID TTY      TIME CMD
  3139 pts/1    0:01 ksh
  4321 pts/1    0:00 s

ingsun1/home/mumolo $ ls -l no*
srwxrwxrwx   1 mumolo   calcolat       0 Jan 23 14:35 nomesocket

ingsun1/home/mumolo $ c
Sono il server. Ho ricevuto: ciao ciao ciao
ingsun1/home/mumolo $ c
ingsun1/home/mumolo $ Sono il server. Ho ricevuto: ciao ciao ciao

ingsun1/home/mumolo $ c
Sono il server. Ho ricevuto: ciao ciao ciao
ingsun1/home/mumolo $ c
ingsun1/home/mumolo $ Sono il server. Ho ricevuto: ciao ciao ciao

[1] +  Done                    s&
ingsun1/home/mumolo $
ingsun1/home/mumolo $ ls -l no*
no*: No such file or directory
```

- Come indicare l'indirizzo del server ?
  - come costante (numero,nome) nel programma → usato con gli alias, utili per lo spostamento del server: se sposto il server su un'altra macchina, devo solo cambiare alias
  - come parametro → piu' semplice e versatile
  - da disco → il file deve essere disponibile (i clienti non sono facilmente portabili)
  - con protocollo di ricerca multicast o broadcast → va bene solo per ambienti locali

- Come determinare l'indirizzo del server? Tramite le DNS e la funzione gethostbyname():

```
struct hostent *gethostbyname( char * nome)
```

guarda nella DNS e ritorna il puntatore, ad es. *hptr, alla struttura

```
struct hostent{
     char *    h_name;        //nome ufficiale della macchina
     char **   h_aliases;     //vettore degli alias
     short     h_addrtype;    //tipo, famiglia di indirizzi
     short     h_length;      //lunghezza indirizzo
     char **   h_addr_list;   //vettore con tutti gli indirizzi della
                              //macchina
};
```

- Gli indirizzi IP sono in binario.
  - Es. int IP; IP=hptr->h_addr_list;

- Esempio di utilizzo della chiamata di sistema gethostbyname: programma hostent.c:

```
/* hostent.c */
#include <stdio.h>
#include <sys/types.h>
#include <netdb.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

main(argc, argv)
int     argc;
char    **argv;
{
        register char   *ptr;
        char            *host_err_str();
        register struct hostent *hostptr;

        if(argc==1) {printf("uso: hostent <lista di host>\n"); exit(1);}
        while (--argc > 0) {
                ptr = *++argv;
                if((hostptr=gethostbyname(ptr))==NULL) {
                        printf("errore di gethostbyname per host:%s\n",
                                ptr);
                        continue;
                }
                printf("nome ufficiale dell'host: %s\n", hostptr->h_name);
                while ((ptr=*(hostptr->h_aliases)) != NULL) {
                        printf("        alias: %s\n", ptr);
                        hostptr->h_aliases++;
                }
                printf("        tipo indir. = %d, lunghezza indir. = %d\n",
                        hostptr->h_addrtype, hostptr->h_length);
                switch(hostptr->h_addrtype) {
                case AF_INET: pr_inet(hostptr->h_addr_list, hostptr->h_length);
                        break;
                default: printf("tipo di indirizzo sconosciuto");
                        break;
                }
        }
}
```

```
/* scandisce una lista di indirizzi di Internet;
   visualizza ciascun indirizzo con punto decimale
*/
pr_inet(listptr, length)
char    **listptr;
int     length;
{
        struct in_addr *ptr;
        while((ptr=(struct in_addr *) *listptr++) != NULL)
        printf("        indirizzo di Internet: %s\n", inet_ntoa(*ptr));
}
```

- Esempio di utilizzo di **hostent**:

```
ingsun1/home/mumolo/net/cli-ser $ hostent falso uts ingsun1 smartultra Caesar mail
errore di gethostbyname per host:falso
nome ufficiale dell'host: uts.univ.trieste.it
        tipo indir. = 2, lunghezza indir. = 4
        indirizzo di Internet: 140.105.48.1
nome ufficiale dell'host: ingsun1
        alias: ingsun1.univ.trieste.it
        alias: loghost
        alias: mailhost
        tipo indir. = 2, lunghezza indir. = 4
        indirizzo di Internet: 140.105.161.101
nome ufficiale dell'host: smartultra.univ.trieste.it
        tipo indir. = 2, lunghezza indir. = 4
        indirizzo di Internet: 140.105.50.206
nome ufficiale dell'host: Caesar.univ.trieste.it
        tipo indir. = 2, lunghezza indir. = 4
        indirizzo di Internet: 140.105.50.115
nome ufficiale dell'host: utsmail1.univ.trieste.it
        alias: mail.univ.trieste.it
        tipo indir. = 2, lunghezza indir. = 4
        indirizzo di Internet: 140.105.48.120
ingsun1/home/mumolo/net/cli-ser $
```

- Determinare il numero di porta di un servizio: routine getservbyname()

  struct servent *getservbyname(char *servizio, char *protocollo)

  Ritorna un puntatore alla struttura

```
struct servent{
   Char *  s_name;        //nome del servizio
   Char **      s_aliases;          //altri alias
   Short   s_port;        //porta per questo servizio
   Char *  s_proto;       //protocollo da usare
};
```

  Il numero di porta restituito e' un intero che inizia con il byte piu' significativo → compatibile con la struttura sockaddr_in

Programma di prova: servent.c. Esempio d'uscita di servent.c:

```
ingsun1/home/mumolo/net/cli-ser $ servent echo daytime chargen ftp time finger
nome ufficiale del servizio: echo
        porta per questo servizio = 7, protocollo = tcp
nome ufficiale del servizio: daytime
        porta per questo servizio = 13, protocollo = tcp
nome ufficiale del servizio: chargen
        alias: ttytst
        alias: source
        porta per questo servizio = 19, protocollo = tcp
nome ufficiale del servizio: ftp
        porta per questo servizio = 21, protocollo = tcp
nome ufficiale del servizio: time
        alias: timserver
        porta per questo servizio = 37, protocollo = tcp
nome ufficiale del servizio: finger
        porta per questo servizio = 79, protocollo = tcp
ingsun1/home/mumolo/net/cli-ser $
```

```
/* servent.c */
#include <stdio.h>
#include <sys/types.h>
#include <netdb.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

main(argc, argv)
int     argc;
char    **argv;
{
        register char   *ptr;
        char            *serv_err_str();
        register struct servent *servptr;

        if(argc==1) {printf("uso: servent <lista di servizi>\n"); exit(1);}
        while (--argc > 0) {
                ptr = *++argv;
                if((servptr=getservbyname(ptr, (char *)0))==NULL) {
                        printf("errore di getservbyname per il servizio:%s\n",
                                ptr);
                        continue;
                }
                printf("nome ufficiale del servizio: %s\n", servptr->s_name);
                while ((ptr=*(servptr->s_aliases)) != NULL) {
                        printf("        alias: %s\n", ptr);
                        servptr->s_aliases++;
                }
                printf("        porta per questo servizio = %d, protocollo = %s\n",
                        servptr->s_port, servptr->s_proto);
        }
}
```

- Funzioni per la compatibilita' dei formati numeri in rete con i formati numerici della macchina (ordine di byte della rete):
  - shortrete=htons(shortlocale) : host short → network
  - shortlocale=ntohs(shortrete) : network → host short
  - longrete=htonl(longlocale) : host long → network
  - longlocale=ntohl(longrete) : network → host long
- Routine per la manipolazione degli indirizzi:
  - ulong inet_addr(char *stringa) → da stringa, es "140.105.50.60", a indirizzo IP (intero a 32 bit)
  - ulong inet_network(char *stringa) → da stringa in notazione decimale a indirizzo di rete a 32 bit con zeri per la parte di host
  - struct in_addr inet_makeaddr(int net, int localhost)→ combina indirizzo rete/host per formare un indirizzo IP
  - int inet_lnaof(struct in_addr in) → da IP a indirizzo della rete locale
  - int inet_netof(struct in_addr in) → da IP a numero di rete
  - char * inet_ntoa(struct in_addr in) → da indirizzo IP a indirizzo decimale come stringa

Routine di manipolazione di indirizzi

```
#include <stdio.h>
#include <sys/types.h>
#include <netdb.h>
#include <sys/socket.h>
main(int argc, char **argv)
{
        ulong ip,irete,ireteloc,ir;

        if(argc==1) {printf("uso: inet <indirizzo notazione decimale>\n"); exit(1);}
        printf("indirizzo fornito=%s\n",argv[1]);
        ip=(ulong)inet_addr(argv[1]);
        printf("IP da notazione decimale=%x\n",ip);
        irete=inet_network(argv[1]);
        printf("indirizzo di rete a 32 bit con zeri per la parte di host=%x\n",irete);
}
```

- **Esempio di uscita del programma:**

```
ingsun1/home/mumolo/net/cli-ser $ inet 140.105.50.115
indirizzo fornito=140.105.50.115
IP da notazione decimale=8c693273
indirizzo di rete a 32 bit con zeri per la parte di host=8c693273
ingsun1/home/mumolo/net/cli-ser $
```

- **Determinare il numero di protocollo: funzione getprotobyname()**

```
struct protoent * getprotobyname(char *nome_del_protocollo)
```

Ritorna un puntatore alla struttura

```
struct protoent{
     char *    p_name;   //nome del protocollo
     char **   p_aliases;     //lista degli alias
     short     p_proto;  //numero ufficiale del protocollo
};
```

- Esempio d'uscita del programma protoent.c che usa getprotobyname per leggere i protocolli

```
ingsun1/home/mumolo/net/cli-ser $ protoent ip udp tcp
nome ufficiale del protocollo: ip
        alias: ip
        numero ufficiale del protocollo = 0
nome ufficiale del protocollo: udp
        alias: udp
        numero ufficiale del protocollo = 17
nome ufficiale del protocollo: tcp
        alias: tcp
        numero ufficiale del protocollo = 6
```

```
   /* protoent.c */
#include <stdio.h>
#include <sys/types.h>
#include <netdb.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

main(argc, argv)
int     argc;
char    **argv;
{
        register char   *ptr;
        char            *serv_err_str();
        register struct protoent *servptr;

        if(argc==1) {printf("uso: protoent <lista di protocolli>\n"); exit(1);}

        while (--argc > 0) {
                ptr = *++argv;
                if((servptr=getprotobyname(ptr))==NULL) {
                        printf("errore di getprotobyname per il protocollo:%s\n", ptr);
                        continue;
                }
                printf("nome ufficiale del protocollo: %s\n", servptr->p_name);
                while ((ptr=*(servptr->p_aliases)) != NULL) {
                        printf("        alias: %s\n", ptr);
                        servptr->p_aliases++;
                }
                printf("        numero ufficiale pdel protocollo = %d\n",
                        servptr->p_proto);

        }
}
```

Allocazione di un socket TCP/IP

`int socket(int famiglia, int servizio, int protocollo)` → ritorna il socket identifier

- in TCP: famiglia di protocolli=PF_INET, servizio=SOCK_STREAM, protocollo=TCP
  Ovvero: s=socket(PF_INET, SOCK_DGRAM, getprotobyname("tcp")→proto);

NB. se il servizio e' SOCK_STREAM, il protocollo deve essere TCP, quindi il terzo argomento e' irrilevante e viene messo a zero
  Ovvero: s=socket(PF_INET, SOCK_STREAM, 0)

- In UDP: famiglia di protocolli=PF_INET, servizio=SOCK_DGRAM, protocollo=UDP
  Ovvero: s=socket(PF_INET, SOCK_DGRAM, getprotobyname("UDP")→proto);

- Scegliere un numero di porta delle applicazioni remota e locale
  - La porta del server deve essere nota a tutti i clienti
  - La porta del cliente deve essere tale che:
    - → Non sia usata da altre applicazioni
    - → Non sia gia' assegnata a servizi
  - Soluzione: determinazione automatica della porta mediante la connect

- Scelta dell'indirizzo IP locale
  - Se la macchina e' collegata ad una rete, la scelta e' semplice
  - Se la macchina ha molti indirizzi, la scelta puo' essere complessa per motivi di routine
  - Soluzione: si lascia vuoto il campo IP → TCP/IP sceglie un indirizzo locale automaticamente mediante la chiamata di sistema connect

In conclusione:

- Connessione di un socket ‘s’ del cliente ad un server mediante chiamata di sistema connect

```
int connect(s, indirizzo remoto, lunghezza indirizzo)
```

Indirizzo remoto: puntatore a sockaddr_in che specifica l’endpoint remoto

- Connect non ritorna finche’ non si stabilisce una connessione TCP oppure timeout
- Connect ritorna 0 se si e’ realizzata una connessione oppure SOCKET_ERROR
- Connect realizza 4 funzioni:
  - verifica la validita’ del socket
  - carica indirizzo remoto dal secondo argomento al socket
  - sceglie un indirizzo IP locale e la porta locale
  - inizia la connessione TCP
- Comunicazione in TCP:

```
        /                 →
       |     invio richiesta con write()
cliente |                 ←
       |     aspetta risposta con read()
        \
```

→ TCP/IP invia affidabilmente sequenze di dati ma puo’ dividere i dati in blocchi diversi

Frammento di un programma d'esempio per leggere un messaggio in TCP:

```
char buf[LEN]; char * bufptr;
bufptr=buf;
int buflen=LEN;
...
write(s,"richiesta",lunghezza richiesta);
while( (n=read (s, bufptr, buflen) > 0){
       bufptr += n;
       buflen-=n;
}
```

- Chiusura della connessione TCP:

  `int shutdown(s,direzione);` //chiude la connessione nella direzione specificata

  direzione: 0 = chiude l'input (il server non riceve piu' richieste)
  1 = chiude l'output (il client non manda piu' le richieste)
  2 = entrambi
- shutdown riporta un messaggio di chiusura al server

- Comunicazione con UDP (servizio SOCK_DGRAM):
  - UDP in Modo connesso: chiamate di sistema connect, write, read. Stesse di TCP ma: connect non inizia scambi di pacchetti, non testa la validita' dell'indirizzo remoto! Salva solo l'informazione nella struttura dati → maggiore facilita'
  - UDP in modo non connesso: specifica la destinazione remota ogni volta che manda un messaggio → maggiore flessibilita'
  - Ogni volta che il cliente chiama write() o read() viene inviato o ricevuto tutto il pacchetto
    - Non c'e' bisogno di chiamate ripetute come con TCP
  - Chiusura di un socket in UDP
    - Close: non informa della chiusura il modo remoto
    - Close e shutdown non informano della chiusura il nodo remoto

Esempio di codifica per la connessione a socket

```
int connectsocket(const char *host, const char *servizio, const char *trasporto)
{
        struct sockaddr_in      sin;
        struct hostent          *phe;
        struct servent          *pse;
        struct protoent         *ppe;
        int                     s, type;

        sin.sin_family=AF_INET;
        if(pse=getservbyname(servizio, trasporto)) {
                        sin.sin_port=pse->s_port;
                        printf("porta per il servizio %s = %d\n", servizio, pse->s_port);
        }
        else
        if((sin.sin_port=htons((u_short)atoi(servizio))) == 0) {
                printf("errore\n"); exit(1);
        }

        if(phe=gethostbyname(host)) memcpy(&sin.sin_addr, phe->h_addr, phe->h_length);
        else
        if((sin.sin_addr.s_addr=inet_addr(host))==-1) {
                printf("errore\n"); exit(1);
        }
        printf("indirizzo IP di %s = %x\n", host, sin.sin_addr.s_addr);

        ppe=getprotobyname(trasporto);
        if(strcmp(trasporto,"udp")==0) type=SOCK_DGRAM; else type=SOCK_STREAM;
        printf("tipo servizio = %d, numero protocollo = %d\n", type, ppe->p_proto);

        s=socket(PF_INET, type, ppe->p_proto);

        connect(s, (struct sockaddr*)&sin, sizeof(sin));
        return s;
}
```

Esempio di cliente: programma gc.c per il servizio daytime

```
main( int argc, char **argv )
{
  int sk;
  char buf[200];
  char *host;
  char *servizio="daytime";

  switch(argc){
  case 1:       host="ingsun1";
                break;
  case 2:       host = argv[1];
                break;
  case 3:       servizio = argv[2];
                host = argv[1];
                break;
  default:      printf("Errore di utilizzo!\n");
                exit(1);
  }
  printf("STREAM=%d, DGRAM=%d\n",SOCK_STREAM,SOCK_DGRAM);
  sk=connectsocket(host, servizio, "tcp");

  printf("numero del socket=%d\n",sk);

  read(sk, buf, 200);

  close(sk);
  printf("%s\n", buf);

}
```

- **Esempio di utilizzo di gc.c:**

```
ingsun1/home/mumolo/net/cli-ser $ gc
STREAM=2, DGRAM=1
porta per il servizio daytime = 13
indirizzo IP di ingsun1 = 8c69a165
tipo servizio = 2, numero protocollo = 6
numero del socket=9
Fri Jan 26 09:53:24 2001

ingsun1/home/mumolo/net/cli-ser $ gc smartultra
STREAM=2, DGRAM=1
porta per il servizio daytime = 13
indirizzo IP di smartultra = 8c6932ce
tipo servizio = 2, numero protocollo = 6
numero del socket=9
Fri Jan 26 09:53:08 2001

ingsun1/home/mumolo/net/cli-ser $ gc smartultra chargen
STREAM=2, DGRAM=1
porta per il servizio chargen = 19
indirizzo IP di smartultra = 8c6932ce
tipo servizio = 2, numero protocollo = 6
numero del socket=9
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefg

ingsun1/home/mumolo/net/cli-ser $ gc smartultra ftp
STREAM=2, DGRAM=1
porta per il servizio ftp = 21
indirizzo IP di smartultra = 8c6932ce
tipo servizio = 2, numero protocollo = 6
numero del socket=9
220 smartultra FTP server (SunOS 5.8) ready.

ingsun1/home/mumolo/net/cli-ser $
```

/* Cliente per il servizio di echo */

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define MAXLEN 200
main( int argc, char **argv )
{
  int sk;
  char buf[MAXLEN];
  char *host;
  char *servizio="echo";
  char *msg="Comando di prova\n";
  int n;

  switch(argc){
  case 1:       host="ingsun1";
                break;
  case 2:       host = argv[1];
                break;
  case 3:       servizio = argv[2];
                host = argv[1];
                break;
  case 4:       msg=argv[3];
                servizio = argv[2];
                host = argv[1];
                break;
  default:      printf("Errore di utilizzo!\n");
                exit(1);
  }
  printf("STREAM=%d, DGRAM=%d\n",SOCK_STREAM,SOCK_DGRAM);
  sk=connectsocket(host, servizio, "tcp");  printf("numero del socket=%d\n",sk);

  write(sk, msg, strlen(msg)); /* invio del messaggio al socket */
  printf("Stringa inviata al server: %s\n", msg);
  n=read(sk, buf, MAXLEN); /* ricevo in risposta dal socket */
  printf("Risposta dal server:\n"); write(1,buf,n);  printf("\n");
  close(sk);
}
```

Esempio di utilizzo del servizio di echo

```
ingsun1/home/mumolo/net/cli-ser $ r cc
cc -o ech ech.c
ingsun1/home/mumolo/net/cli-ser $ ech
STREAM=2, DGRAM=1
porta per il servizio echo = 7
indirizzo IP di ingsun1 = 8c69a165
tipo servizio = 2, numero protocollo = 6
numero del socket=9
Stringa inviata al server: Comando di prova

Risposta dal server:
Comando di prova

ingsun1/home/mumolo/net/cli-ser $ ech smartultra echo Provaprovaprova
STREAM=2, DGRAM=1
porta per il servizio echo = 7
indirizzo IP di smartultra = 8c6932ce
tipo servizio = 2, numero protocollo = 6
numero del socket=9
Stringa inviata al server: Provaprovaprova
Risposta dal server:
Provaprovaprova
ingsun1/home/mumolo/net/cli-ser $ ech smartultra 7 blablabla
STREAM=2, DGRAM=1
indirizzo IP di smartultra = 8c6932ce
tipo servizio = 2, numero protocollo = 6
numero del socket=9
Stringa inviata al server: blablabla
Risposta dal server:
blablabla
ingsun1/home/mumolo/net/cli-ser $
```

Programma getch.c Esempio di cliente che realizza le seg. Funzioni: si connette all'host e alla porta specificati, spedisce un messaggio e legge le risposta piu' volte.

```
main( int argc, char **argv )
{
  int sk;
  char buf[MAXLEN];
  char *host;
  char *servizio="daytime";
  char *cmd="help";

  printf("Uso: gech [host [servizio [messaggio]]]\n");
  switch(argc){
  case 1:       host="ingsun1";
                break;
  case 2:       host = argv[1];
                break;
  case 3:       servizio = argv[2];
                host = argv[1];
                break;
  case 4:       cmd=argv[3];
                servizio = argv[2];
                host = argv[1];
                break;
  default:      printf("Errore di utilizzo!\n");
                printf("Uso: gech [host [servizio [messaggio]]]\n");
                exit(1);
  }
  printf("SOCK_STREAM=%d, SOCK_DGRAM=%d\n",SOCK_STREAM,SOCK_DGRAM);
  sk=connectsocket(host, servizio, "tcp");
  printf("numero del socket=%d\n",sk);
  printf("commando da inviare=%s\n",cmd);
  read(sk, buf, MAXLEN); /* legge il socket innanzitutto */
  printf("%s\n", buf);
  write(sk, cmd, strlen(cmd)); /* scrive il commando sul socket */
  write(sk, "\n", 1);
  read(sk, buf, MAXLEN);      /* legge il socket una 1a volta */
  printf("%s\n", buf);
  read(sk, buf, MAXLEN);      /* legge il socket una 2a volta */
  printf("%s\n", buf);
  close(sk);
}
```

Esempio di utilizzo di getch.c

```
ingsun1/home/mumolo/net/cli-ser $ gech
Uso: gech [host [servizio [messaggio]]]
SOCK_STREAM=2, SOCK_DGRAM=1
porta per il servizio daytime = 13
indirizzo IP di ingsun1 = 8c69a165
tipo servizio = 2, numero protocollo = 6
numero del socket=9
comando=help
Fri Jan 26 13:12:13 2001
ingsun1/home/mumolo/net/cli-ser $ gech smartultra daytime
Uso: gech [host [servizio [messaggio]]]
SOCK_STREAM=2, SOCK_DGRAM=1
porta per il servizio daytime = 13
indirizzo IP di smartultra = 8c6932ce
tipo servizio = 2, numero protocollo = 6
numero del socket=9
comando=help
Fri Jan 26 13:12:05 2001

Fri Jan 26 13:12:05 2001

Fri Jan 26 13:12:05 2001

ingsun1/home/mumolo/net/cli-ser $ gech smartultra ftp help
Uso: gech [host [servizio [messaggio]]]
SOCK_STREAM=2, SOCK_DGRAM=1
porta per il servizio ftp = 21
indirizzo IP di smartultra = 8c6932ce
tipo servizio = 2, numero protocollo = 6
numero del socket=9
comando=help
220 smartultra FTP server (SunOS 5.8) ready.

214-The following commands are recognized:


   USER    EPRT    STRU    MAIL*   ALLO    CWD     STAT*   XRMD
   PASS    LPRT    MODE    MSND*   REST*   XCWD    HELP    PWD
   ACCT*   EPSV    RETR    MSOM*   RNFR    LIST    NOOP    XPWD
   REIN*   LPSV    STOR    MSAM*   RNTO    NLST    MKD     CDUP
   QUIT    PASV    APPE    MRSQ*   ABOR    SITE*   XMKD    XCUP
   PORT    TYPE    MLFL*   MRCP*   DELE    SYST    RMD     STOU
214 (*'s => unimplemented)
ingsun1/home/mumolo/net/cli-ser $ gech news.uni-stuttgart.de 119 help
Uso: gech [host [servizio [messaggio]]]
SOCK_STREAM=2, SOCK_DGRAM=1
```

```
porta per il servizio 119 = 119
indirizzo IP di news.uni-stuttgart.de = 8145013b
tipo servizio = 2, numero protocollo = 6
numero del socket=9
comando=help
201 news.uni-stuttgart.de InterNetNews NNRP server INN 1.7.2 08-Dec-1997 (News In Dosen v0.98 (11-Apr-1998)) ready (no
posting).

100 Legal commands
  authinfo user Name|pass Password|generic <prog> <args>
  article [MessageID|Number]
  body [MessageID|Number]
  date
  group newsgroup
  head [MessageID|Number]
  help
  ihave
  last
  list [active|active.times|newsgroups|distributions|distrib.pats|overview.fmt|subscriptions]
  listgroup newsgroup
  mode reader
  newgroups yymmdd hhmmss ["GMT"] [<distributions>]
  newnews newsgroups yymmdd hhmmss ["GMT"] [<distributions>]
  next
  post
  slave
  stat [MessageID|Number]

  xgtitle [group_pattern]
  xhdr header [range|MessageID]
  xover [range]
  xpat header range|MessageID pat [morepat...]
  xpath MessageID
Report problems to <news@news.uni-stuttgart.de>
.
 ihave
  last
  list [active|active.times|newsgroups|distributions|distrib.pats|overview.fmt|subscriptions]
  listgroup newsgroup
  mode reader
  newgroups yymmdd hhmmss ["GMT"] [<distributions>]
  newnews newsgroups yymmdd hhmmss ["GMT"] [<distributions>
  next
  post
  slave
  stat [MessageID|Number]
ingsun1/home/mumolo/net/cli-ser $
ingsun1/home/mumolo/net/cli-ser $ gech news.uni-stuttgart.de 119 date
```

```
Uso: gech [host [servizio [messaggio]]]
SOCK_STREAM=2, SOCK_DGRAM=1
porta per il servizio 119 = 119
indirizzo IP di news.uni-stuttgart.de = 8145013b
tipo servizio = 2, numero protocollo = 6
numero del socket=9
comando=date
201 news.uni-stuttgart.de InterNetNews NNRP server INN 1.7.2 08-Dec-1997 (News In Dosen v0.98 (11-Apr-1998)) ready (no
posting).

111 20010126121940
rt.de InterNetNews NNRP server INN 1.7.2 08-Dec-1997 (News In Dosen v0.98 (11-Apr-1998)) ready (no posting).
ingsun1/home/mumolo/net/cli-ser $ gech news.uni-stuttgart.de 119 body
Uso: gech [host [servizio [messaggio]]]
SOCK_STREAM=2, SOCK_DGRAM=1
porta per il servizio 119 = 119
indirizzo IP di news.uni-stuttgart.de = 8145013b
tipo servizio = 2, numero protocollo = 6
numero del socket=9
comando=body
201 news.uni-stuttgart.de InterNetNews NNRP server INN 1.7.2 08-Dec-1997 (News In Dosen v0.98 (11-Apr-1998)) ready (no
posting).

412 Not in a newsgroup
e InterNetNews NNRP server INN 1.7.2 08-Dec-1997 (News In Dosen v0.98 (11-Apr-1998)) ready (no posting).
ingsun1/home/mumolo/net/cli-ser $ gech news.uni-stuttgart.de 119 list
Uso: gech [host [servizio [messaggio]]]
SOCK_STREAM=2, SOCK_DGRAM=1
porta per il servizio 119 = 119
indirizzo IP di news.uni-stuttgart.de = 8145013b
tipo servizio = 2, numero protocollo = 6
numero del socket=9
comando=list
201 news.uni-stuttgart.de InterNetNews NNRP server INN 1.7.2 08-Dec-1997 (News In Dosen v0.98 (11-Apr-1998)) ready (no
posting).

215 Newsgroups in form "group high low flags".
3dfx.glide.linux 0000014703 0000014629 y
alt.0d 0000010489 0000010450 y
alt.1d 0000070883 0000069563 y
alt.3d 0000073679 0000073577 y
alt.3d.misc 0000013205 0000013172 y
alt.3d.studio 0000078883 0000078203 y
alt.abortion.inequity 0000273631 0000273281 y
alt.abuse.recovery 0000206206 0000204663 y
alt.acting 0000062091 0000060955 y
alt.activism 0000510047 0000509193 y
alt.activism.d 0000050619 0000050473 y
```

```
alt.activism.death-penalty 0000205858 0000204
901 y
alt.adoption 0000278155 0000276170 y
alt.adoption.agency 0000017505 0000017486 y
alt.adoption.searching 0000031450 0000031140 y
alt.aeffle.und.pferdle 0000032935 0000032657 y
alt.agriculture.fruit 0000017398 0000017368 y
alt.agriculture.misc 0000026745 0000026700 y
alt.ahbqs.com.sucks 0000001809 0000001806 y
alt.alcohol 0000045110 0000044931 y
alt.aldus.freehand 0000013425 0000013405 y
alt.aldus.misc 0000008356 0000008343 y
alt.aldus.pagemaker 0000049135 0000049015 y
alt.algebra.help 00000
ingsun1/home/mumolo/net/cli-ser $
ingsun1/home/mumolo/net/cli-ser $ gech news.uni-stuttgart.de 119 'listgroup alt.3d'
Uso: gech [host [servizio [messaggio]]]
SOCK_STREAM=2, SOCK_DGRAM=1
porta per il servizio 119 = 119
indirizzo IP di news.uni-stuttgart.de = 8145013b
tipo servizio = 2, numero protocollo = 6
numero del socket=9
comando=listgroup alt.3d
201 news.uni-stuttgart.de InterNetNews NNRP server INN 1.7.2 08-Dec-1997 (News In Dosen v0.98 (11-Apr-1998)) ready (no
posting).

211 Article list follows
73583
73609
73613
73616
73622
73634
73635
73636
73639
```

Progetto di server

- Funzioni di un server:
  - crea un socket
  - collega il socket alla porta locale dalla quale desidera ricevere
  - realizza un loop infinito durante il quale accetta la prossima richiesta
  - elabora la richiesta e risponde

- Server concorrenti/ server iterativi
  - Gestisce multiple richieste/gestisce una richiesta alla volta
  - Server iterativi possono avere un tempo di risposta alto (ritardo nella coda)

- Server orientati alla connessione/ non orientati alla connessione
  - TCP fornisce collegamento affidabile
  - TCP richiede un socket per ogni connessione
  - UDP permette comunicazione con host multipli da un singolo socket
  - TCP offre comunicazione punto-a-punto: non puo' fornire comunicazione broadcast o multicast

- Collegare il server all'indirizzo locale
  - Creazione socket e collegamento alla porta del servizio locale
  - Funzione bind: int bind(int s, struct sockaddr *indirizzo, int lunghezzaindir)
  - Bind definisce anche l'indirizzo locale: se ci sono diversi indirizzi locali, si usa INADDR_ANY (wildcard) al posto di un indirizzo IP

- Modalita' passiva
  - Il socket viene messo in modalita' passiva con listen
  - Chiamata listen:

`int listen(int s, int nrich);` → s=socket id, nrich=numero di richieste in coda

- Accettare le connessioni: funzione accept

```
int accept(int s, struct sockaddr *p, int *lunghezzaindir);
```

  - Lunghezzaindir viene messo al valore massimo di un buffer; la chiamata di sistema la modifica uguagliandola al valore corretto
  - Accept estrae la connessione dalla coda delle richieste

- I quattro tipi fondamentali di server:
  - Iterativi e non orientati alla connessione
  - Iterativi e orientati alla connessione
  - Concorrenti e non orientati alla connessione
  - Concorrenti e orientati alla connessione

```
/* server iterativo per servizio echo */
#include <stdio.h>      /* standard C i/o facilities */
#include <sys/types.h>  /* system data type definitions */
#include <sys/socket.h> /* socket specific definitions */
#include <netinet/in.h>
#include <netdb.h>

#include "readline.h"

/* gestisce la connessione con il cliente. Fa' l'eco di ogni linea che manda il cliente */
void echo( int sd )
{
  int len;
  char bufin[200],bufout[250];

   while (readline(sd,bufin,100)!=0)
     {
       printf("Received: %s\n",bufin);
       sprintf(bufout,"ECHOING: %s",bufin);
       writeline(sd,bufout);
     }
}
/* Server
   1. crea un socket
```

```
   2. assegna un indirizzo al socket e stampa il numero di porta
   3. metti il socket in modo passivo (listen)
   4. ciclo infinito
        prendi la prossima connessione
        gestisci la connessione (fai l'echo)
*/
main()
{
  int ld,sd;
  struct sockaddr_in skaddr;
  struct sockaddr from;
  int addrlen,length;

/* crea il socket con: IP protocol family (PF_INET), TCP protocol (SOCK_STREAM) */
  if ((ld = socket( PF_INET, SOCK_STREAM, 0 )) < 0){
      printf("Problem creating socket\n");
      exit(1);
  }

/* stabilisce l'indirizzo del socket:
   address family  AF_INET
   IP address INADDR_ANY (any of our IP addresses)
   il numero di porta e' assegnato dal kernel
*/

  skaddr.sin_family = AF_INET;
  skaddr.sin_addr.s_addr = INADDR_ANY;
  skaddr.sin_port = 0;

  if (bind(ld, (struct sockaddr *) &skaddr, sizeof(skaddr))<0) {
        printf("Problem connecting binding\n");
        exit(0);
  }
```

```
 /* ricava il numero di porta e lo stampa */
   length = sizeof( skaddr );
   if (getsockname(ld, (struct sockaddr *) &skaddr, &length)<0){
         printf("Error getsockname\n");
         exit(1);
   }

   printf("The port number is %d\n",ntohs(skaddr.sin_port));
 /* mette il socket in modo passivo */
   if (listen(ld,5) < 0 ){
         printf("Error calling listen\n");
         exit(1);
   }
while (1){ /* versione server iterativo! */
         addrlen=18;
         if ( (sd = accept( ld, &from, &addrlen)) < 0)
             {
               printf("Problem with accept call\n");
               exit(1);
             }

         printf("Got a connection - processing...\n");
         echo(sd);
         printf("Done with connection\n");
         close(sd);
   }
 }
```

```
while (1) { /* versione server concorrente! */
   addrlen=18;
   if ( (sd = accept( ld, &from, &addrlen)) < 0) {
           printf("Problem with accept call\n");
           exit(1);
    }
    if(fork()==0){
           close(ld);
           printf("Got a connection\n");
           echo(sd);
           printf("Done with connection\n");
           exit(0);
     }
     close(sd);
   }
}
```

```
/* readline.c (da linkare con server.c). Legge una linea dal descrittore, un byte alla volta, controllando il
fineriga. Mette la linea nel buffer, termina con null. Ritorna il nr. di char fino al terminatore (escluso). */
#include <string.h>
int readline(int fd, char *ptr, int maxlen)
{
        int     n, rc;
        char    c;

        for (n = 1; n <= maxlen; n++) {
                if ( (rc = read(fd, &c, 1)) == 1) {
                        *ptr++ = c;
                        if (c == '\n')
                                break;
                } else if (rc == 0) {
                        if (n == 1)
                                return(0);      /* EOF, non sono stati letti dati */
                        else
                                break;          /* EOF, e' stato letto qualche dato */
                } else
                        return(-1);     /* error */
        }
/* elimina  newline */
        *(ptr-1) = 0;
        return(n-1);
}
/* Scrive "n" byte sul descrittore.*/
int writen(int fd, char *ptr, int nbytes)
{
        int     nleft, nwritten;
        nleft = nbytes;
        while (nleft > 0) {
                nwritten = write(fd, ptr, nleft);
                if (nwritten <= 0)
                        return(nwritten);               /* errore */
                nleft -= nwritten;
                ptr   += nwritten;
        }
        return(nbytes - nleft);
}
int writeline( int fd, char *s )
{
  char c = '\n';
  writen(fd,s,strlen(s));
  writen(fd,&c,1);
}
```

```
/* cliente per il server echo. Programma client.c */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#include "readline.h"


/* defisci la stringa da inviare */

char *message = "PROVA DI COLLEGAMENTO!!!\n";


/* client <nomeserver> <numero di porta> */

main( int argc, char **argv )
{
  int sk;
  struct sockaddr_in skaddr;
  struct hostent *hp;
  char buf[200];

  if (argc!=3)
  {
    printf("Usage: client <server name> <server port>\n");
    exit(0);
  }

/* crea un socket con: IP protocol family (PF_INET), TCP protocol (SOCK_STREAM) */

  if ((sk = socket( PF_INET, SOCK_STREAM, 0 )) < 0)
  {
    printf("Problem creating socket\n");
    exit(1);
  }

/*
   riempie la struttura degli indirizzi. L'address family e' IP  (AF_INET). L'indirizzo IP del server e' trovato
   chiamando gethostbyname con il nome del server
*/

  skaddr.sin_family = AF_INET;
  if ((hp = gethostbyname(argv[1]))==0)
    {
```

```
      printf("Invalid or unknown host\n");
      exit(1);
    }
  memcpy( &skaddr.sin_addr.s_addr, hp->h_addr, hp->h_length);
  skaddr.sin_port = htons(atoi(argv[2]));
/* connessione con il server */
  if (connect(sk,(struct sockaddr *) &skaddr,sizeof(skaddr)) < 0 )
    {
      printf("Problem connecting socket\n");
      exit(1);
    }
/* manda una stringa e aspetta la risposta */
  writeline(sk,message);
  readline(sk,buf,200);
  printf("%s\n",buf);
}
```

- Esempio di utilizzo:

```
ingsun1/home/mumolo/net/cli-ser $ server&
[2]     8402
ingsun1/home/mumolo/net/cli-ser $ The port number is 38451
ingsun1/home/mumolo/net/cli-ser $ ps
   PID TTY      TIME CMD
  8273 pts/1    0:01 ksh
  8402 pts/1    0:00 server
ingsun1/home/mumolo/net/cli-ser $ client
Usage: client <server name> <server port>
ingsun1/home/mumolo/net/cli-ser $ client ingsun1 38451
Got a connection
Received: MESSAGGIO DI PROVA!!!
Done with connection
ECHOING: MESSAGGIO DI PROVA!!!
ingsun1/home/mumolo/net/cli-ser $
```