Socket

Una socket è un punto estremo di un canale di comunicazione accessibile mediante un file descriptor

Le socket costituiscono un fondamentale strumento di comunicazione, basato sullo scambio di messaggi, tra processi locali e/o remoti (sia UNIX che di altri sistemi operativi) :

⇒ vengono superate le limitazioni delle pipe e delle FIFO (comunicazione locale, con le pipe ristretta ai processi di uno stesso utente, discendenti di uno stesso avo)

Una socket va creata all'interno di un dominio di comunicazione che determina i protocolli utilizzati :

⇒ le socket sono l'elemento di base per la programmazione di applicativi e servizi di rete (ad es. utilizzando i protocolli TCP/IP - Internet)

Socket

Una socket è un oggetto con un tipo, determinato dal sottoinsieme delle seguenti proprietà che quel tipo di socket garantisce:

1) **consegna ordinata dei messaggi** (l' ordine di ricezione dei messaggi è uguale all'ordine di trasmissione)
2) **consegna non duplicata** (lo stesso messaggio non può essere consegnato due volte)
3) **consegna affidabile** (i messaggi inviati non possono andare persi)
4) **preservamento dei confini dei messaggi** (i messaggi inviati non vengono frazionati nella comunicazione)
5) **supporto per i messaggi** out-of-band (messaggi prioritari che superano quelli ordinari nella coda di ricezione)
6) **comunicazione orientata alla connessione** (più avanti)

Le pipe (che non sono socket) garantiscono le proprietà 1, 2 e 3

Socket

Alcuni tipi predefiniti di socket

- **SOCK_STREAM**
  orientata alla connessione, trasferisce byte stream con proprietà 1, 2, 3, 5, 6 ma non 4
- **SOCK_DGRAM**
  trasferisce datagram con proprietà 4 ma non 1, 2, 3, 5, 6
- **SOCK_SEQPACKET**
  trasferisce datagram con proprietà 1, 2, 3, 4, 5, 6 (non è implementata nel dominio di comunicazione Internet)
- **SOCK_RAW**
  permette l'accesso diretto ai protocolli di rete sottostanti (ad es. Ethernet)

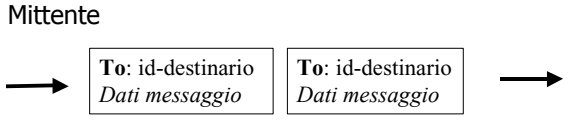
Socket

Le modalità di comunicazione byte stream e datagram

- **byte stream (SOCK_STREAM / protocollo TCP)**

Nella connessione il flusso di dati trasmesso è uguale a quello ricevuto (ma il protocollo può suddividere i messaggi i cui confini non sono quindi preservati)

- **datagram (SOCK_DGRAM / protocollo UDP)**



Ogni messaggio reca l'indicazione del destinario ed è singolarmente inviato, trasferito e ricevuto

Socket

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

int socket(int domain, int type, int protocol);
```

Una socket viene creata nel dominio di comunicazione `domain` (detto anche protocol family o address family)

Domini principali:
- PF_UNIX dominio per una comunicazione locale
- PF_INET dominio per una comunicazione su TCP/IP (IPv4)
- PF_INET6 dominio per una comunicazione su TCP/IP (IPv6)

`type` indica il tipo di socket che si vuole creare (ad. es. `SOCK_STREAM` oppure `SOCK_DGRAM`) ; `protocol` indica lo specifico protocollo utilizzato tra quelli disponibili nel dominio (se ne esiste uno solo vale zero)

Viene restituito un descrittore da utilizzare sia per la lettura (ricezione) che la scrittura (invio) di messaggi

Socket

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

int bind(int sockfd, struct sockaddr *my_addr, socklen_t
addrlen);
```

Una socket può venire legata ad un indirizzo (nome)

`bind` assegna un nome ad una socket per renderla designabile (indirizzabile) da parte di un processo intenzionato a comunicare con il processo che ha creato la socket

L'interfaccia è generica (`my_addr, addrlen`) in quanto i diversi domini di comunicazione prevedono indirizzi di forma diversa:

- PF_UNIX indirizzo= un percorso nel file system (ad es. /tmp/.X11-unix/X0)
- PF_INET indirizzo= (indirizzo IP del nodo, numero porta)

Socket

Nel dominio PF_INET

indirizzo socket = (indirizzo IP del nodo, numero porta)

Ad esempio
indirizzo socket = (172.28.14.253, 22)

Indirizzo IP del nodo	Porta assegnata
darkstar.cedi.unipr.it	al servizio sshd

È l'indirizzo della socket utilizzata dal servizio sshd su darkstar

Se l'indirizzo specificato in una `bind` è già assegnato ad un'altra socket (la porta non è libera) la `bind` fallisce

Se il numero di porta contenuto nell'indirizzo specificato nella bind è zero, viene restituita una porta a caso tra quelle libere
- numeri di porta < 1024 riservati ai servizi di rete di sistema (root)
- numeri di porta >= 1024 utilizzabili liberamente dagli utenti

Socket

Porte riservate ai servizi di rete di sistema (/etc/services)

```
tcpmux          1/tcp                           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
msp             18/tcp                          # message send protocol
msp             18/udp                          # message send protocol
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
fsp             21/udp          fspd
ssh             22/tcp                          # SSH Remote Login Protocol
ssh             22/udp                          # SSH Remote Login Protocol
telnet          23/tcp
# 24 - private
smtp            25/tcp          mail
# 26 - unassigned
time            37/tcp          timserver
time            37/udp          timserver
rlp             39/udp          resource        # resource location
nameserver      42/tcp          name            # IEN 116
whois           43/tcp          nicname
re-mail-ck      50/tcp                          # Remote Mail Checking Protocol
re-mail-ck      50/udp                          # Remote Mail Checking Protocol
domain          53/tcp          nameserver      # name-domain server
…
```

Nel dominio AF_INET le strutture dati utilizzate da `bind` e dalle altre primitive sono:

```
struct sockaddr_in {
   sa_family_t     sin_family; /* address family: AF_INET */
   u_int16_t       sin_port;   /* port in network byte order */
   struct in_addr  sin_addr;  /* internet address */
   };
```

Network byte order corrisponde alla convenzione big-endian : i byte più significativi sono all'indirizzo più basso

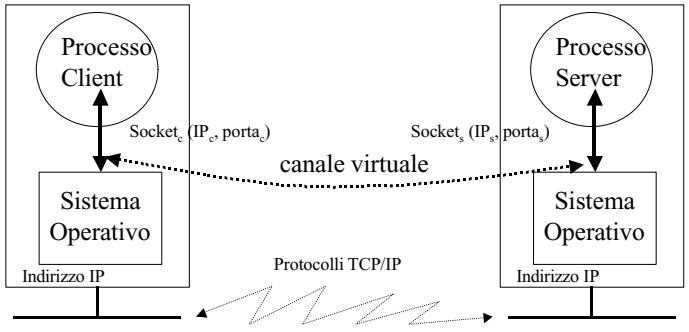
⇒ utilizzare sempre la macro `htons(numeroporta)` che effettua (se necessario) la conversione

```
/* Internet address. */
struct in_addr {
  u_int32_t  s_addr;  /* address in network byte order */
};
```

L'indirizzo Internet può essere ricavato dal nome simbolico del nodo (ad es. darkstar.cedi.unipr.it) utilizzando la funzione `gethostbyname`

SOCK_STREAM nel dominio AF_INET

Prima di effettuare il trasferimento dati deve essere creata una connessione (protocollo TCP di TCP/IP)



Completata con successo la fase di creazione della connessione (socket "connessa") è sufficiente inoltrare i messaggi lungo la connessione perché raggiungano la destinazione

Creazione della connessione

Un **cliente** inizia una connessione sulla propria socket specificando l'indirizzo della socket del server:

```
connect(int cli_sockfd, ...);
/* bloccante */
```

sincronizzazione

Il **server** dichiara al S.O. la sua disponibilità a ricevere connessioni sulla propria socket:

```
listen( int serv_sockfd,...) ;
/* non bloccante */
```

Il **server** attende richieste di connessioni sulla propria socket e riceve un nuovo descrittore (conn_sockfd) per ogni nuova connessione:

```
conn_sockfd=accept(int serv_sockfd,...);
/* bloccante */
```

Su socket connesse

```
write(cli_sockfd, …) ;
read(cli_sockfd, …) ;
```

```
read(conn_sockfd, …) ;
write(conn_sockfd, …) ;
```



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

int connect(int sockfd, const struct sockaddr *serv_addr,
socklen_t addrlen);
```

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

int listen(int s, int backlog);
```

`backlog` specifica la dimensione massima della code delle richieste di connessione pendenti (non ancora accettate)

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

int accept(int s, struct sockaddr *addr, socklen_t *addrlen);
```

Socket

Server concorrente

Normalmente un server su SOCK_STREAM (cfr. i servizi TCP, ad es. ftp) crea un nuovo server figlio per gestire una nuova connessione da un cliente mentre il server padre continua ad attendere nuove connessioni

```
do
 { /* Attesa di una connessione */
 if((msgsock= accept(sock,(struct sockaddr *)&client,(int *)&len))<0) {
       perror("accept"); exit(-1); }
 else  {
    if(fork()==0) {
       /* Server figlio  */
       printf("Serving connection from %s, port %d\n",
            inet_ntoa(client.sin_addr), ntohs(client.sin_port));

       close(sock); /* Non interessa la socket di contr. per listen */
       ftpserv(msgsock); /* Servizio specifico del server */
       close(msgsock); /* La socket connessa può essere rimossa */
       exit(0);
       }
    else /* Server padre */
       close(msgsock); /* Non interessa la socket connessa */
       }
    }
while(1);
```


Socket

Datagram

Non vi alcun stato di connessione (protocollo UDP di TCP/IP)

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

int  sendto(int  s,  const  void *msg, size_t len, int flags,
const struct sockaddr *to, socklen_t tolen);
```

Invio di un messaggio con designazione esplicita del destinatario (indirizzo specificato in `to`)

```
int recvfrom(int s, void *buf, size_t len, int flags, struct
sockaddr *from, socklen_t *fromlen);
```

Ricezione di un messaggio

L'indirizzo del mittente del messaggio viene posto in `from` (se diverso da `NULL`)

Socket

Esempio n. 1

```
/*********************************************************************

             Semplice interazione su socket di tipo STREAM

                    S    E   R   V  E   R

Utilizzo:             server [numeroporta]

*********************************************************************/

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>

#define DEFAULTPORT    1520
#define BYTES_NR        8192
#define MSG_NR          64
```


Socket

Esempio n. 1 (server cont.)

```
main(int argc, char  *argv[])
{
  int           sock,length;
  struct        sockaddr_in    server;
  int           s,msgsock,rval;
  struct        hostent *hp,*gethostbyname();
  char          buf[BYTES_NR];
  int           myport;

  if(argc == 2)
    myport = atoi(argv[1]);
  else
    myport = DEFAULTPORT;

  if((myport < 1024  && myport != 0)|| myport > 65535)
    {
      fprintf(stderr, "Il numero di porta puo essere 0 (per ottenerne
una libera)\noppure deve essere compresa tra 1024 e 65535\n");
      exit(-1);
    }
```

Socket

Esempio n. 1 (server cont.)

```
/* Crea la  socket STREAM */
sock= socket(AF_INET,SOCK_STREAM,0);
if(sock<0)
  {   perror("creazione stream socket");
  exit(1);
  }

server.sin_family =  AF_INET;
/* La socket viene legata a tutti gli indirizzi IP del server
(un indirizzo per ciascuna interfaccia di rete):
il server otterrà i pacchetti ricevuti su qualunque interfaccia.
Con INADDR_ANY non serve conoscere l'indirizzo IP del nodo su cui
esegue il server */
server.sin_addr.s_addr= INADDR_ANY;
server.sin_port = htons(myport);

if (bind(sock,(struct sockaddr *)&server,sizeof server)<0)
  {
    perror("bind su stream socket");
    exit(1);
  }
```


Socket

Esempio n. 1 (server cont.)

```
/* Chiede conferma dell'indirizzo assegnato alla socket  */
length= sizeof server;
if(getsockname(sock,(struct sockaddr *)&server,&length)<0)
  { perror("getsockname"); exit(-1); }

printf("Porta (della socket) del server =
                    #%d\n",ntohs(server.sin_port));

/* Il server e' pronto ad accettare connessioni */
if(listen(sock,2) <0)
 { perror("listen"); exit(-1); }

do
   {
   /* Attesa di una richiesta di connessione: l'indirizzo della socket
      del mittente viene in questo caso ignorato (NULL) */

   if((msgsock= accept(sock,(struct sockaddr *)NULL,(int *)NULL)) < 0)
      {
    perror("accept");
    exit(-2);
  }
     else
```


Socket

Esempio n. 1 (server cont.)

```
do
  {
    s = 0;
    /* Ricezione del messaggio  */
    do {
/* Il messaggio può essere stato frammentato dal protocollo TCP */

      if((rval = read(msgsock,&buf[s],sizeof buf))<0)
            {
            perror("read su  stream message");
            exit(-3);
            }
      s+= rval;
      printf("r=%d(%d) byte letti\n",rval,s);
    }
    while((s!=BYTES_NR) && rval !=0);


    if(rval == 0)
      printf("Termine della connessione\n");
    else
      {
```


Socket

Esempio n. 1 (server cont.)

```
      /* Invio della risposta */

      if((rval = write(msgsock,buf,sizeof buf))<0)
        {
        perror("writing on stream socket");
        exit(-4);
        }

      printf("w=%d byte scritti\n",rval);
        }

    } while(rval !=0);
     close (msgsock);
  }
 while(1); /* ^C o un altro segnale per terminare il server */

 exit(0);
}
```

Socket

Esempio n. 1

```
/*********************************************************************

             Semplice interazione su socket di tipo STREAM


                       C  L  I  E  N  T


Utilizzo:            client nomeserver portaserver


*********************************************************************/

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>

#define BYTES_NR        8192
#define MSG_NR          64
```


Socket

Esempio n. 1 (client cont.)

```
main(int argc, char *argv[])
  {
  int          i,s,sock,rval;
  struct       sockaddr_in server;
  struct       hostent *hp,*gethostbyname();
  char  buf[BYTES_NR];


  if(argc != 3)
    {
      fprintf(stderr,"Uso: %s nomeserver portaserver\n\n",argv[0]);
      exit(-1);
    }

  /* Crea una  socket di tipo STREAM per il dominio TCP/IP */

  if((sock= socket(AF_INET,SOCK_STREAM,0)) <0)
    {
      perror("creazione stream socket");
      exit(1);
    }
```


Socket

Esempio n. 1 (client cont.)

```
/* Ottiene l'indirizzo IP del server */
  server.sin_family=  AF_INET;
  hp=  gethostbyname(argv[1]);

  if(hp==0)
    {
      fprintf(stderr,"%s: server sconosciuto",argv[1]);
      exit(2);
    }

  memcpy( (char *)&server.sin_addr, (char *)hp->h_addr ,hp->h_length);

  /* La porta è sulla linea di comando */
  server.sin_port= htons(atoi(argv[2]));

  /* Tenta di realizzare la connessione */
  printf("Connessione in corso...\n");
  if(connect(sock,(struct sockaddr *)&server,sizeof server) <0)
    {
      perror("connect su stream socket");
      exit(1);
    }
```


Socket

Esempio n. 1 (client cont.)

```
  printf("…connesso.\n");

  rval=BYTES_NR;

  /* Invio/Ricezione di MSG_NR messaggi */
  for(i=0;i<MSG_NR;i++)
    {
      if((rval = write(sock,buf,sizeof buf))<0)
        perror("write su stream socket");

      printf("w=%d byte scritti\n",rval);

      s=0 ;
      do
{ /* Il messaggio puo` essere stato frammentato dal protocollo TCP */
          if((rval = read(sock,&buf[s],sizeof buf))<0)
            perror("reading stream message");

          s+= rval;
          printf("r=%d(%d) byte letti\n",rval,s);
        }
      while((s!=BYTES_NR) && rval !=0);
    }

  close(sock);
  exit(0);
}
```

Esempio n. 2 (header file)

```
/*********************************************************************
              FTP-like CLIENT-SERVER su socket di tipo STREAM

                      ftpdefines.h
*********************************************************************/
#define BYTES_NR       8192
#define MSG_NR         64

#define SOL_TCP          pp->p_proto

#define RICHMSG_MAXPATHNAME 256

typedef struct _RICHIESTA_MSG {
  char filename[RICHMSG_MAXPATHNAME];
} RICHIESTA_MSG;

typedef struct _RISPOSTA_MSG {
  int result;
  char errmsg[512];
  int filesize;
} RISPOSTA_MSG;
```


Esempio n. 2 (server)

```
/*********************************************************************
              FTP-like CLIENT-SERVER su socket di tipo STREAM

                    S    E   R   V  E   R
*********************************************************************/
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <sys/timeb.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <unistd.h>
#include <errno.h>
#include "ftpdefines.h"

char    buf[BYTES_NR];

/***********************************************
Utilizzo:    ftpserver numeroporta
***********************************************/
```


Esempio n. 2 (server - cont.)

```
main()
{
  int          sock,length;
  struct       sockaddr_in    server,client;
  char         buff[512];
  int          s,msgsock,rval,rval2,i;
  struct       hostent *hp,*gethostbyname();

  /* Crea la  socket STREAM */
  sock= socket(AF_INET,SOCK_STREAM,0);
  if(sock<0)
    {   perror("opening stream socket");
    exit(1);
    }
  server.sin_family =  AF_INET;
  server.sin_addr.s_addr= INADDR_ANY;
  server.sin_port = htons(1520);
  if (bind(sock,(struct sockaddr *)&server,sizeof server)<0)
    {
      perror("binding stream socket");
      exit(1);
    }
```


Esempio n. 2 (server - cont.)

```
  length= sizeof server;
  if(getsockname(sock,(struct sockaddr *)&server,&length)<0)
    {
      perror("getting socket name");
      exit(1);
    }

  printf("Socket port #%d\n",ntohs(server.sin_port));

  /* Pronto ad accettare connessioni */
  listen(sock,2);

  do  {
      /* Attesa di una connessione */
      msgsock= accept(sock,(struct sockaddr *)&client,(int *)&length);

      if(msgsock ==-1)
               { perror("accept"); exit(-1);
      else
          {
            if(fork()==0)    {
               printf("Serving connection from %s, port %d\n",
                   inet_ntoa(client.sin_addr), ntohs(client.sin_port));
```

Socket

Esempio n. 2 (server - cont.)

```
                close(sock);
                ftpserv(msgsock);
                close(msgsock);
                exit(0);
               }
             else
               close(msgsock);
           }
    } while(1);
}

ftpserv(int sock)
{
int             s,rval,nread, fd;
struct stat     fbuf;
RICHIESTA_MSG   rich_mesg;
RISPOSTA_MSG    risp_mesg;

s = 0;
 /* Ricezione del comando GET  */

 if((rval=read(sock,&rich_mesg,sizeof(RICHIESTA_MSG)))<0)
   {
     perror("reading client request");
```


Socket

Esempio n. 2 (server - cont.)

```
        exit(-1); }

 if((fd= open(rich_mesg.filename,O_RDONLY))<0)
   { fprintf(stderr,"Non riesco ad aprire il file %s (%s)...uscita
               !\n",rich_mesg.filename,strerror(errno));
     risp_mesg.result = -1;
     strcpy(risp_mesg.errmsg,strerror(errno));
   }
 else  {
   risp_mesg.result= 0;
   fstat(fd,&fbuf); /* Ottiene la dimensione del file */
   risp_mesg.filesize= fbuf.st_size;
   }
 /* Invio della risposta */
if((rval = write(sock,&risp_mesg,sizeof(RISPOSTA_MSG)))<0)
  perror("writing on stream socket");

if(risp_mesg.result  != 0)  return -1;

 do {
   if((nread = read(fd,buf,sizeof buf))<0)
          perror("reading from file");
   if (nread >0)
       if((rval = write(sock,buf,nread))<0)
                perror("writing on stream socket");
 }
 while(nread > 0);
}
```


Socket

Esempio n. 2 (client)

```
/*************************************************************************
             FTP-like CLIENT-SERVER su socket di tipo STREAM

                          C   L   I   E   N   T
*************************************************************************/

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <sys/timeb.h>
#include <fcntl.h>

#include "ftpdefines.h"

struct protoent *pp;

char    buf[BYTES_NR];

/************
Utilizzo:  ftpclient nomeserver portaserver nomefile
************/
```


Socket

Esempio n. 2 (client - cont.)

```
main(argc,argv)
int     argc;char *argv[];
{
int                     i,s,fd,sock,rval,rval2;
struct  sockaddr_in     server;
struct  hostent         *hp,*gethostbyname();
char                    copiafilename[RICHMSG_MAXPATHNAME+16];
RICHIESTA_MSG           rich_mesg;
RISPOSTA_MSG            risp_mesg;

if(argc != 4) {
    fprintf(stderr,"Uso: %s servername porta nomefile\n\n",argv[0]);
    exit(-1);
  }

/* Crea una  socket di tipo STREAM per il dominio TCP/IP */
sock=   socket(AF_INET,SOCK_STREAM,0);

if(sock<0)
        {
        perror("opening stream socket");
        exit(1);
        }
```

Socket

Esempio n. 2 (client - cont.)

```
/* Ottiene l'indirizzo del server */
server.sin_family=     AF_INET;

hp=     gethostbyname(argv[1]);
if(hp==0){
        fprintf(stderr,"%s: unknown host",argv[1]);
        exit(2);
        }
memcpy( (char *)&server.sin_addr, (char *)hp->h_addr ,hp->h_length);

/* La porta e' sulla linea di comando */
server.sin_port= htons(atoi(argv[2]));

/* Tenta di realizzare la connessione */
printf("Connecting to the server %s...\n",argv[1]);
if(connect(sock,(struct sockaddr *)&server,sizeof server) <0)
        {
        perror("connecting stream socket");
        exit(1);
        }

printf("Connected to the server.\n");
```


Socket

Esempio n. 2 (client - cont.)

```
strncpy(rich_mesg.filename,argv[3],RICHMSG_MAXPATHNAME);

/* Invio comando RICHIESTA (GET) */
write(sock,&rich_mesg,sizeof(RICHIESTA_MSG));

/* Riceve la  RISPOSTA dal server */
 if((rval = read(sock,&risp_mesg,sizeof(RISPOSTA_MSG)))<0)
   perror("reading server answer");

if(risp_mesg.result !=0) {
    fprintf(stderr,"OOPS il server risponde %d (%s)...uscita
!\n",risp_mesg.result,risp_mesg.errmsg);
    close(sock);
    exit(0);
  }

strcpy(copiafilename,"copia.");
strcat(copiafilename, rich_mesg.filename);

if((fd= open(copiafilename,O_WRONLY|O_CREAT|O_TRUNC,0644))<0) {
        fprintf(stderr,"Non posso aprire il file copia %s ...uscita
!\n",copiafilename);
        close(sock);
```


Socket

Esempio n. 2 (client - cont.)

```
        exit(0);
        }
s=0;
do
  {
    if((rval = read(sock,buf,sizeof buf))<0)
      perror("reading stream message");

    if(rval >0)
      {
       write(fd,buf,rval);
       putchar('.');
       s += rval;
      }
  }
 while(rval !=0);

printf("\nTrasferimento completato - %s - letti %d
        byte\n",rich_mesg.filename,s);

close(sock);
close(fd);
exit(0);
}
```


Socket

Esempio n. 3

```
/*************************************************************************

              Semplice ping-pong su socket di tipo DATAGRAM

                     S    E    R    V    E    R

Utilizzo:              server [numeroporta]

*************************************************************************/
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <sys/timeb.h>
#include <string.h>

#define BYTES_NR       512
#define MSG_NR         32

#define DEFAULTPORT 3001
 char   buf[BYTES_NR];
```

Socket

Esempio n. 3 (server - cont)

```
main(argc,argv)
     int argc;char *argv[];
{
 int           sock,length;
 struct        sockaddr_in    server,client;
 int           msgsock,rval,i;
 struct hostent *hp,*gethostbyname();
 int myport;

if(argc >2)
  {
    fprintf(stderr,"Uso: %s [portaserver]\n",argv[0]);
    exit(-1);
  }

  if(argc == 2)
    myport = atoi(argv[1]);
  else
    myport = DEFAULTPORT;

  if((myport < 1024  && myport != 0)|| myport > 65535)
    {
      fprintf(stderr, "Il numero di porta puo essere 0 (per ottenerne
una libera)\noppure deve essere compresa tra 1024 e 65535\n");
      exit(-1);
    }
```


Socket

Esempio n. 3 (server - cont)

```
/* Crea la socket DGRAM */
sock=   socket(AF_INET,SOCK_DGRAM,0);
if(sock<0)
        {
        perror("open sulla socket dgram");
        exit(1);
        }

/* Name socket using wildcards */
server.sin_family =    AF_INET;
server.sin_addr.s_addr= INADDR_ANY;
server.sin_port = htons(myport);

if (bind(sock,(struct sockaddr *)&server,sizeof server)<0)
        {
        perror("bind sulla socket dgram");
        exit(1);
        }

/* Find out assigned port and print out */
length= sizeof server;
if(getsockname(sock,(struct sockaddr *)&server,&length)<0)
        {
        perror("getsockname");
        exit(1);
        }
```


Socket

Esempio n. 3 (server - cont)

```
printf("Server attivo sulla porta #%d\n",ntohs(server.sin_port));

while(1)
  {
    bzero(buf,sizeof buf);
    if((rval = recvfrom(sock,buf,sizeof buf, 0, (struct sockaddr* )
               &client, (socklen_t *)&length ))<0)
      {
       perror("recvfrom sulla socket dgram");
       exit(-3);
      }

    printf("Messaggio ricevuto dal client IP=%s porta=%d: rispedisco il
messaggio\n",inet_ntoa(client.sin_addr), client.sin_port);
    strcat(buf,"*");
    if(sendto(sock,buf,sizeof buf,0,(struct sockaddr *)&client,sizeof
client)<0)
      {
       perror("recvfrom sulla socket dgram");
       exit(-4);
      }
  } /* ^C o un altro segnale per farlo terminare */

}
```


Socket

Esempio n. 3 (client)

```
/*************************************************************************

              Semplice ping-pong su socket di tipo DATAGRAM

                    C   L   I   E   N   T

Utilizzo:             client nomeserver portaserver

*************************************************************************/

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>

#include <sys/time.h>
#include <unistd.h>

#define BYTES_NR       512
#define MSG_NR         32
```

Socket

Esempio n. 3 (client - cont)

```
char    buf[BYTES_NR];
char    buf2[BYTES_NR];

char    msg[MSG_NR][BYTES_NR];
char    answ[MSG_NR][BYTES_NR];

struct timeval xstime[MSG_NR];
struct timeval xftime[MSG_NR];

main(argc,argv)
int     argc;char *argv[];
{
int                     i,sock,rval,length;
unsigned long           delay;
struct  sockaddr_in     server,client;
struct  hostent         *hp,*gethostbyname();

if(argc !=3)
  {
    fprintf(stderr,"Uso: %s nomeserver portaserver\n",argv[0]);
    exit(-1);
  }
```


Socket

Esempio n. 3 (client - cont)

```
/* Prepara i messaggi da inviare al server  */
for(i=0;i<MSG_NR;i++)
        {
        sprintf(&msg[i][0],"%d",i);
        }

/* Crea la socket DGRAM */
sock=   socket(AF_INET,SOCK_DGRAM,0);
if(sock<0)
        {
        perror("opening stream socket");
        exit(1);
        }

client.sin_family=     AF_INET;
client.sin_addr.s_addr = INADDR_ANY;
client.sin_port = htons(0);

if (bind(sock,(struct sockaddr *)&client,sizeof client) <0)
        {
        perror("bind su socket dgram");
        exit(1);
        }
```


Socket

Esempio n. 3 (client - cont)

```
length= sizeof client;
if(getsockname(sock,(struct sockaddr *)&server,&length)<0)
        {
        perror("getsockname");
        exit(1);
        }

/* Ottiene l'indirizzo IP del server */
hp = gethostbyname(argv[1]);
if (hp == 0)
        {
        fprintf(stderr,"%s : nodo sconosciuto",argv[1]);
        exit(2);
        }
bcopy( (char *)hp ->h_addr, (char *)&server.sin_addr,hp ->h_length);
server.sin_family = AF_INET;
server.sin_port = htons(atoi(argv[2]));

for(i=0;i<MSG_NR;i++)
        {
        strcpy(buf,msg[i]);
        /* Ottiene il tempo corrent (prima della send)  */
        gettimeofday(&xstime[i],NULL);
```


Socket

Esempio n. 3 (client - cont)

```
        if(sendto(sock,buf,sizeof buf,0,(struct sockaddr
                        *)&server,sizeof server)<0)
                perror("sendto sulla socket dgram");
        if((rval = recvfrom(sock,buf2,sizeof buf2, 0, (struct sockaddr
                *)NULL, (socklen_t *)NULL))<0)
          {
            perror("recvfrom sulla socket dgram");
            exit(-3);
      }
        strcpy(answ[i],buf2);
        /* Ottiene il tempo corrente (dopo la recvfrom)  */
        gettimeofday(&xftime[i],NULL);
        }
 close(sock);

 printf("Ping-pong con %s con datagrammi da %d
                bytes\n",inet_ntoa(server.sin_addr),BYTES_NR);
 for(i=0;i<MSG_NR;i++)
        {
        /* Calcolo del ritardo  */
        delay= (xftime[i].tv_sec-xstime[i].tv_sec)
                *1000000.+(xftime[i].tv_usec-xstime[i].tv_usec);
        printf("msg n.%d [%s]: %0.3f ms\n",i,answ[i],delay/1000.);
        }
exit(0);
}
```

Select

```
#include <sys/time.h>
#include <sys/types.h>
#include <unistd.h>

int select(int n, fd_set *readfds, fd_set *writefds,
           fd_set *exceptfds, struct timeval *timeout);
```

La primitiva `select` permette di attendere una variazione di stato per i file descriptor (riferiti a file, pipe, socket, ...) all'interno di tre distinti insiemi di file descriptor (tipo fd_set):

- si attende la disponibilità di dati in lettura per i fd contenuti in `readfds`
- si attende la possibilità di scrittura immediata sui fd contenuti in `writefds`
- si attende la presenza di eccezioni per i fd contenuti in `exceptfds`

`n` è il valore del descrittore più alto nei tre insiemi, aumentato di 1

L'attesa può essere limitata superiormente da un `timeout`

Select

Il valore di uscita è il numero di descrittori che sono variati di stato (zero significa che è scaduto l'intervallo di timeout)

Gli `fd_set` sono modificati in uscita dalla `select` in modo che contengano i soli fd che hanno variato di stato

Macro utili per la manipolazione di variabili fd_set

`FD_ZERO(fd_set *set)`	azzera un fd_set
`FD_CLR(int fd, fd_set *set)`	rimuove un fd da un fd_set
`FD_SET(int fd, fd_set *set)`	inserisce un fd in un fd_set
`FD_ISSET(int fd, fd_set *set)`	predicato che verifica se un certo fd è membro di un fd_set

Select

Esempio: processo lettore su due pipe

```
int                 max_fd;
fd_set              rpipe_fds;
struct timeval      tv;
…
if(pipe(pipea) < 0) …
if(pipe(pipeb) < 0) …

if(fork() == 0) {
/* I descrittori di lettura delle pipe sono inseriti nel
fd_set di lettura) */

FD_ZERO(&rpipe_fds);
FD_SET(pipea[0], &rpipe_fds);
FD_SET(pipeb[0], &rpipe_fds);

/* Attesa al massimo di 5 secondi */
tv.tv_sec= 5 ; tv.tv_nsec= 0 ;

max_fd = pipeb[0]+1;
```


Select

Esempio

```
/* Il processo si blocca in attesa della lettura su pipea e/o
su pipeb o del timeout */
retval= select(max_fd, &rpipe_fds, NULL, NULL, &tv) ;

if(retval)
       {
       /* Occorre verificare quale descrittore sia pronto */
       if(FD_ISSET(pipea[0], &rpipe_fds)
              read(pipea[0], buffer, sizeof message;

       if(FD_ISSET(pipeb[0], &rpipef_ds)
              read(pipeb[0], buffer, sizeof message;
       }
else
       {/* retval vale 0 */
       printf("Timeout !\n");
       }

}
```