UNIVERSITÀ DEGLI STUDI DI PARMA
Facoltà di Ingegneria
Corso di Laurea in Ingegneria Elettronica-Informatica-Telecomunicazioni
a.a. 2001-2002

CORSO DI SISTEMI OPERATIVI A - ESERCITAZIONE 6

1 Socket

Una socket fornisce una interfaccia di comunicazione tra processi che si trovano su macchine distinte. In generale ogni socket è identificata da un indirizzo e da un numero di porta. Le principali funzioni che consentono la creazione e la gestione delle socket sono le seguenti:

PRIMITIVE	Descrizione
socket	Crea una socket di un dato dominio,tipo e protocollo
bind	Assegna un nome alla socket
listen	Specifica il massimo numero di connessioni che una socket server accetta
accept	Una socket server accetta una richiesta di connessione da una socket client
connect	Una socket client invia una richiesta di connessione ad una socket server

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int domain,int type,int protocol)
```

L'argomento *domain* specifica la convenzione da utilizzare per la denominazione della socket (tipicamente AF_INET), *type* specifica il tipo della socket (SOCK_STREAM per comunicazione affidabile). L'argomento *protocol* viene settato normalmente a zero. Il valore di ritorno della funzione è un intero che identifica la socket creata.

```
int bind(int sid,struct sockaddr* addr_p,int len)
```

L'intero *sid* è il descrittore ritornato dalla funzione socket. L'argomento *addr_p* punta ad una struttura che contiene il nome da assegnare alla socket. L'argomento *len* specifica la dimensione di tale struttura.

```
int listen(int sid,int size)
```

L'argomento *size* specifica il massimo numero di connessioni che possono essere accettate in coda da una socket. Nella maggior parte dei sistemi UNIX il valore massimo consentito è 5.

```
int accept(int sid,struct sockaddr* addr_p,int* len_p)
```

L'argomento *sid* specifica il descrittore della socket del processo server, *addr_p* punta ad una struttura che contiene il nome della socket client che effettua la connessione.

```
int connect(int sid,struct sockaddr* addr_p,int len)
```

L'argomento *sid* specifica il descrittore della socket del processo client, *addr_p* punta ad una struttura che contiene il nome della socket server alla quale il processo client intende connettersi.

2 Esercizi

Tutti i file con il codice sorgente degli esercizi proposti (es∗.c) si trovano nel direttorio:

/home/soa/eserc6/

Esercizio 1:

Esempio di scambio di messaggi tra client e server mediante socket. Il client invia un messaggio al server. Il server, dopo aver ricevuto il messaggio comunica al client l'avvenuta ricezione.
Utilizzo (invocare i due programmi in due shell separati):
./server1 <numero porta>
./client1 darkstar <numero porta>
Utilizzare come <numero porta> il numero di matricola.

Server1.c

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

void error(char *msg)
{
    perror(msg);
    exit(1);
}

int main(int argc, char *argv[])
{
     int sockfd, newsockfd, portno, clilen;
     char buffer[256]="";
     struct sockaddr_in serv_addr, cli_addr;
     int n;
     if (argc < 2) {
         fprintf(stderr,"ERRORE, nessuna porta specificata\n");
         exit(1);
     }
     sockfd = socket(AF_INET, SOCK_STREAM, 0);
     if (sockfd < 0)
        error("ERRORE DI APERTURA DELLA SOCKET");

     portno = atoi(argv[1]);
     serv_addr.sin_family = AF_INET;
     serv_addr.sin_addr.s_addr = INADDR_ANY;
     serv_addr.sin_port = htons(portno);
     if (bind(sockfd, (struct sockaddr *) &serv_addr,
```

```
              sizeof(serv_addr)) < 0)
              error("ERRORE DI BINDING");
     listen(sockfd,5);
     clilen = sizeof(cli_addr);
     newsockfd = accept(sockfd,
                 (struct sockaddr *) &cli_addr,
                 &clilen);
     if (newsockfd < 0)
          error("ERRORE DI ACCEPT");

     n = read(newsockfd,buffer,255);
     if (n < 0) error("ERRORE in lettura dalla socket");
     printf("(SERVER) Ecco il messaggio ricevuto dal client: %s\n",buffer);
     n = write(newsockfd,"MESSAGGIO RICEVUTO, FINE COMUNICAZIONE", 38);
     if (n < 0) error("ERRORE in scrittura sulla socket");
     return 0;
}
```

Client1.c

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

void error(char *msg)
{
    perror(msg);
    exit(0);
}

int main(int argc, char *argv[])
{
    int sockfd, portno, n;
    struct sockaddr_in serv_addr;
    struct hostent *server;

    char send_buffer[256];
    char rec_buffer[256]="";

    if (argc < 3) {
       fprintf(stderr,"uso %s nomehost porta\n", argv[0]);
       exit(0);
    }
    portno = atoi(argv[2]);
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
```

```
        error("ERRORE in apertura");
    server = gethostbyname(argv[1]);
    if (server == NULL) {
        fprintf(stderr,"ERRORE, l'host non esiste\n");
        exit(0);
    }

    serv_addr.sin_family = AF_INET;
    memcpy((char *)&serv_addr.sin_addr,(char *)server->h_addr,server->h_length);

    serv_addr.sin_port = htons(portno);
    if (connect(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr)) < 0)
        error("ERRORE di connessione");
    printf("(CLIENT) Scrivere un messaggio: ");

    fgets(send_buffer,255,stdin);
    n = write(sockfd,send_buffer,strlen(send_buffer));
    if (n < 0)
         error("ERRORE in scrittura sulla socket");

    n = read(sockfd,rec_buffer,255);
    if (n < 0)
         error("ERRORE in lettura sulla socket");
    printf("(CLIENT) Ecco il messaggio ricevuto dal server: %s\n",rec_buffer);
    return 0;
}
```

Esercizio 2:

Il client invia un file al server. Il server salva il file ricevuto in un file locale. Utilizzo (invocare i due programmi in due shell separati):
./server2 <numero porta> <nome nuovo file>
./client2 <nome file da trasferire> darkstar <nome porta>
Utilizzare come <nome porta> il numero di matricola.

Server2.c

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


#define MAXBUF     8192

int main(int argc, char * argv[])
{
```

```
int server_socket,connect_socket,portno;
unsigned int client_addr_len;
int retcode,fd;
struct sockaddr_in client_addr, server_addr;
char line[MAXBUF];
struct hostent *server;

if(argc==1)
{
  printf("Usage:\n%s port nomefilelocale\n",argv[0]);
  exit(0);
}
printf("Server: fase di inizializzazione\n");
server_socket = socket(AF_INET,SOCK_STREAM,0);
if(server_socket == -1)
{
  perror("aprendo il socket del server");
  exit(-1);
}

portno = atoi(argv[1]);
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = INADDR_ANY;
server_addr.sin_port   = htons(portno);

retcode = bind(server_socket,
               (struct sockaddr *)&server_addr,
               sizeof(server_addr));
if(retcode == -1)
{
  perror("bind error");
  exit(-1);
}
listen(server_socket,1);
printf("Server: attendo connessione\n");
client_addr_len = sizeof(client_addr);
connect_socket = accept(server_socket,
                        (struct sockaddr *) & client_addr,
                        &client_addr_len);
printf("Server: accettata nuova connessione\n Apro file locale %s",argv[2]);
fd = open(argv[2],O_WRONLY|O_TRUNC|O_CREAT,0644);
if(fd == -1)
{
  perror("aprendo il file locale");
  exit(-2);
}
do {
  retcode = read(connect_socket,line,MAXBUF);
```

```
    if(retcode != -1)
      write(fd,line,retcode);
  } while(retcode > 0);
  close(fd);
  printf("\nFine del messaggio, chiusura della connessione\n");
  close(connect_socket);

  printf("Chiusura dei lavori ... \n");
  close(server_socket);
  exit(0);
}
```

Client2.c

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

#define MAXBUF     8192

int main(int argc, char * argv[])
{
  int client_socket,fd,portno;
  int retcode,letti;
  struct sockaddr_in  server_addr;
  char message[MAXBUF];
  char *nomehost,*filename;

  if(argc == 1)
  {
    printf("Usage:\n%s nomefile nomehost portno\n",argv[0]);
    return(0);
  }
  filename = argv[1];
  nomehost = argv[2];
  printf("Client (%d): fase di inizializzazione\n",getpid());
  client_socket = socket(AF_INET,SOCK_STREAM,0);
  if(client_socket == -1)
  {
    perror("aprendo il socket del cliente");
    return(-1);
  }

  portno = atoi(argv[3]);
  server_addr.sin_family = AF_INET;
  server_addr.sin_port   = htons(portno);
  memcpy(&server_addr.sin_addr,(gethostbyname(nomehost)->h_addr),
 sizeof(server_addr.sin_addr));
```

```
 retcode = connect(client_socket,
(struct sockaddr *)&server_addr,
sizeof(server_addr));
 if(retcode == -1)
 {
   perror("connettendo il socket");
   return(-1);
 }
 fd = open(filename,O_RDONLY);
 if(fd == -1)
 {
   perror("aprendo il file");
   return(-3);
 }
 do {
   letti = read(fd,message,MAXBUF);
   if(letti > 0) { /* solo se la lettura ha avuto buon fine */
     retcode = write(client_socket,message,letti);
     if(retcode == -1)
     {
perror("scrivendo il messaggio");
return(-3);
     }
   }
 } while (letti > 0);
 close(fd);
 close(client_socket);
 return(0);
}
```

Esercizio 3:

Si progetti in ambiente Unix/C la seguente interazione tra il processo server Ps e i suoi client Pi:

1. il processo Ps viene eseguito sul server darkstar alla porta specificata come argomento di invocazione del programma;
2. i processi clienti Pi (anch'essi vengono eseguiti su darkstar) inviano richieste di servizio arbitrarie al server Ps attraverso socket di tipo STREAM;
3. Ps attiva un processo figlio per ogni nuova richiesta di servizio accettata;

Utilizzo ./server3 <numero porta>
./client3 darkstar <nome porta>
Utilizzare come <nome porta> il numero di matricola.

Server3.c

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

```
#include <netinet/in.h>
#include <stdio.h>


typedef struct _RICHIESTA_MSG
{
  int req;
} RICHIESTA_MSG;

typedef struct _RISPOSTA_MSG
{
  int answ;
} RISPOSTA_MSG;


main(int argc, char* argv[])
{
  int sock,length,portno;
  struct  sockaddr_in  server,client;
  int pid,s,msgsock,rval,rval2,i;
  struct hostent *hp,*gethostbyname();
  RICHIESTA_MSG request;
  RISPOSTA_MSG  answer;


  if (argc !=2) {
    printf("Usage: %s <port_number>\n", argv[0]);
    exit(-1);
  }

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

  portno = atoi(argv[1]);

  /* Utilizzo della wildcard per accettare le connessioni da ogni client */
  server.sin_family =  AF_INET;
  server.sin_addr.s_addr= INADDR_ANY;
  server.sin_port = htons(portno);
  if (bind(sock,(struct sockaddr *)&server,sizeof server)<0)
  {
    perror("binding stream socket");
    exit(1);
```

```
  }

  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,(socklen_t *)&length);

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

      /* Il client e' autorizzato */
      if((pid = fork())== 0)
      {
        close(sock);
        read(msgsock,&request,sizeof(request));

        /* Esecuzione del servizio */
        answer.answ = request.req + (rand()%100);
        write(msgsock,&answer,sizeof(answer));
        close(msgsock);
        exit(0);
      }
      else
      {
        if(pid == -1) /* Errore nella fork */
        {
          perror("Error on fork: ");
          exit(-1);
        }
        else
{
```

```
          /* OK, il padre torna in accept */
          close(msgsock);
        }
      }
    }
  }
  while(1);
  exit(0);
}
```

Client3.c

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


typedef struct _RICHIESTA_MSG
{
  int req;
} RICHIESTA_MSG;

typedef struct _RISPOSTA_MSG
{
  int answ;
} RISPOSTA_MSG;

main(int argc,char* argv[])
{
  int i,s,sock,rval,rval2,portno;
  struct  sockaddr_in  server;
  struct hostent *hp,*gethostbyname();
  int tsum;
  float avg;
  time_t nsec;
  unsigned short nmil;
  RICHIESTA_MSG request;
  RISPOSTA_MSG  answer;


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

  srand(getpid());
```

```
  /* 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);
  }

  /* 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 */
  portno = atoi(argv[2]);
  server.sin_port= htons(portno);

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

  printf("Connected.\n");

  request.req = rand() %100;
  write(sock,&request,sizeof(request));

  read(sock,&answer,sizeof(answer));

  printf("Sent %d - server answer is : %d\n",request.req,answer.answ);

  close(sock);

  exit(0);
}
```

Esercizio 4:

Si realizzi un server TCP sulla porta <nome porta> che ritorni ai clienti la data e l'ora in ASCII.
Utilizzo:
./server4 <nome porta>
Da un altro shell invocare il comando *telnet darkstar <nome porta>*.
Utilizzare come <nome porta> il numero di matricola.

Server4.c

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


main(int argc, char *argv[])
{
  char buf[26];
  time_t ticks;
  int sock,length,msgsock,rval,portno;
  struct sockaddr_in server;
  struct hostent *hp;


  if (argc !=2) {
    printf("Usage: %s <port_number>\n", argv[0]);
    exit(-1);
  }

  printf("\n*-----------------------------------------------*\n");
  printf("*                                               *\n");
  printf("*         SERVER data e ora correnti            *\n");
  printf("*                                               *\n");
  printf("* ^c per terminare!                             *\n");
  printf("*                                               *\n");
  printf("*-----------------------------------------------*\n\n");

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

```
  portno = atoi(argv[1]);

  server.sin_family = AF_INET;         /* Utilizzo della wildcard per*/
  server.sin_addr.s_addr= INADDR_ANY;  /* accettare le connessioni da ogni client */
  server.sin_port = htons(portno);
  if (bind(sock,(struct sockaddr *)&server,sizeof server)<0)
  {
    perror("binding stream socket");
    exit(1);
  }

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

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

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

  do {
    /* Attesa di una connessione */
    msgsock= accept(sock,(struct sockaddr *)0,(int *)0);

    if(msgsock ==-1)
      perror("accept");
    else
    {
      ticks = time(NULL);        /*calcola data e ora corrente*/
      strcpy(buf,ctime(&ticks)); /*la converte in stringa e la memorizza in buf[26]*/

      /* Invio dell'informazione */
      if((rval = write(msgsock,buf,sizeof buf))<0)
        perror("writing on stream socket");
      printf("%d byte scritti\n",rval);
    }
    close (msgsock);
  }
  while(1);

  exit(0);
}
```

EsercizioProposto:

Si realizzi un client che si collega al server dell'esercizio precedente e legge data e ora corrente.

3 Socket Datagram

Esercizio 5:

Esempio di ping pong su socket DATAGRAM. Utilizzo ./servdgram <numero porta>
./clientdgram localhost <numero porta>
Nel caso in cui la porta specificata risulti in uso da altri utenti modificare il codice (suggerimento: creare il numero della porta in modo casuale)

servdgram.c

```
#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 64
#define MSG_NR 512


char buf[BYTES_NR];


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


  if(argc !=2)
    {
      fprintf(stderr,"Usage: %s port\n",argv[0]);
      exit(-1);
    }

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

  /* Name socket using wildcards */
```

```
  server.sin_family =  AF_INET;
  server.sin_addr.s_addr= INADDR_ANY;
  server.sin_port = htons(atoi(argv[1]));
  if (bind(sock,(struct sockaddr *)&server,sizeof server)<0)
    {
      perror("binding stream socket");
      exit(1);
    }

  /* Find out assigned port and print out */
  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));

  while(1)
    {
      do
        {
          bzero(buf,sizeof buf);
          if((rval = recvfrom(sock,buf,sizeof buf, 0,
                              (struct sockaddr *)&client,
                              (socklen_t *)&length ))<0)
            perror("reading stream message");
          i=0;
          if(rval==0)
            printf("Ending connection\n");
          else
            {
              printf("Message received: sending back\n");
              strcat(buf,"*");
              if(sendto(sock,buf,sizeof buf, 0,
                 (struct sockaddr *)&client, sizeof client)<0)
              /* (write(msgsock,buf,sizeof buf)<0) */
              perror("writing on stream socket");
            }
         } while(rval !=0);
     }
  exit(0);
}
```

clientdgram.c

```
#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 64
#define MSG_NR 512


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(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,"Usage: %s servername serverport\n",argv[0]);
      exit(-1);
    }

    for(i=0;i<MSG_NR;i++)
      {
        sprintf(&msg[i][0],"%d",i);
      }

  /* Create socket */
  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("sending datagram message");
    exit(1);
  }

length= sizeof client;

if(getsockname(sock,(struct sockaddr *)&server,&length)<0)
  {
    perror("getting socket name");
    exit(1);
  }
printf("Socket port #%d\n",ntohs(client.sin_port));

hp = gethostbyname(argv[1]);
if (hp == 0)
  {
    fprintf(stderr,"%s :unknow host",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]);
    gettimeofday(&xstime[i],NULL);
    if(sendto(sock,buf,sizeof buf,0,(struct sockaddr *)&server,sizeof server)<0)
      perror("sendto problem");

    if((rval = read(sock,buf2,sizeof buf2))<0)
      perror("reading stream message");

    strcpy(answ[i],buf2);

    gettimeofday(&xftime[i],NULL);
  }
close(sock);

for(i=0;i<MSG_NR;i++)
  {
```

```
      delay = (xftime[i].tv_sec-xstime[i].tv_sec)*1000000
               +(xftime[i].tv_usec-xstime[i].tv_usec);
      printf("msg %d [%s]: %0.3f ms\n",i,answ[i],delay/1000);
    }

  exit(0);
}
```