```c
/*
  ******************************************************************

  Ejemplo de programación sobre sockets PF_INET de tipo Datagrama (UDP)

  Extremo Cliente ------>

Compilar con:
    gcc -o udpclt1 udpclt1.c -lnsl -lsocket  -lposix4

Uso del programa (2):
    udpclt1     ip_servidor     puerto_servidor
    udpclt1     ip_servidor     puerto_servidor     "mensaje"

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

#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <errno.h>
#include <unistd.h>
#include <string.h>

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


#define DIM 1024        // tamaño buffer de datos de mensaje

// Mensaje a enviar si no se indica como parámetro
char *mensaje_def= "Este es el mensaje enviado desde el cliente SOCK_DGRAM";

int main(int argc, char *argv[])
{
    int socketID; // identificador de dispositivo para el socket
    struct sockaddr_in datosSocketSrv;  // dirección IP, tipo y puerto del servidor
    struct sockaddr_in datosSocketClt;  // dirección IP, tipo y puerto del cliente
    struct hostent *hp;             // dirección IP a partir de nombre simbólico
    ssize_t res;                    // resultado de un sento()
    size_t longdir;                 // tamaño estructura socaddr_in

    char mensaje[DIM];              // buffer memoria para los mensajes


    // Comprueba los parámetros
    //*****************************************************************************
    if(argc<3)
    {
        printf("Uso del programa: udpclt1   servidor  puerto  <mensaje>\n");
        exit(0);
    }
    if(argc==4)     strcpy(mensaje,argv[3]);        // mensaje pasado como parámetro
    else            strcpy(mensaje,mensaje_def);    // mensaje por defecto

    /*
      1ª etapa: Creamos un socket PF_INET de tipo SOCK_DGRAM (obtenemos un manejador)
    ******************************************************************************
    */
    socketID = socket( PF_INET, SOCK_DGRAM, 0);
    if(socketID<0)
    {
        perror("Creando socket");
        exit(-1);
    }


    /*
     2ª etapa: Obtenemos la dirección IP y puerto del servidor
               a partir de los  parámetros
    ********************************************************************
    */

    // convierte el nombre simbólico a una direeción IP
```



```c
    hp=gethostbyname(argv[1]);
    if(hp == NULL)
    {
        fprintf(stderr, "\n %s: host desconocido\n", argv[1]);
        close(socketID);
        exit(-2);
    }
    // copiamos la dirección IP en la estructura datosSocket
    memcpy( &(datosSocketSrv.sin_addr), hp->h_addr, hp->h_length);
    datosSocketSrv.sin_family=AF_INET;                  // tipo de dirección
    datosSocketSrv.sin_port = htons(atoi(argv[2])); // convierte el puerto a un número de re

    /*
     3ª etapa:   Enviamos inforamción al servidor:
        ssize_t sendto(int socket, const void *message, size_t length, int flags,
                  const struct sockaddr *dest_addr, size_t dest_len);
    ********************************************************************************
    */
    res = sendto(socketID, mensaje, strlen(mensaje)+1, 0,
            (struct sockaddr *) &datosSocketSrv, sizeof datosSocketSrv);
    if(res>0)
    {   // muestra lo que ha enviado
        printf("%d bytes Enviados a [%s:%u]: %s\n", res,
                inet_ntoa(datosSocketSrv.sin_addr), ntohs(datosSocketSrv.sin_port), mensaje)

        // Datos del puerto del cliente (asignados automáticamente tras la primera transmisi
        longdir=sizeof datosSocketClt;
        res = getsockname(socketID,  (struct sockaddr *) &datosSocketClt, &longdir);
        if(res==0)
            printf("---Cliente -> IP: %s,  Puerto: %u---\n\n",
                inet_ntoa(datosSocketClt.sin_addr),     ntohs(datosSocketClt.sin_port));

    }

    /*
     4ª etapa:
     Cerrar el socket
     ********************************************************************************
    */
    close(socketID);

    exit(0);

}
```

```c
/*
  ********************************************************************

  Ejemplo de programación sobre sockets PF_INET de tipo Datagrama (UDP)

   ------> Extremo SERVIDOR

Compilar con:
    gcc -o udpsrv1 udpsrv1.c -lnsl -lsocket   -lposix4

Uso del programa (2):
        udpsrv1             -> usa puerto por defecto (se muestra en la consola)
        udpsrv1 puerto

  ********************************************************************
*/
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <errno.h>
#include <unistd.h>
#include <string.h>

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


#define DIM         1024    // tamaño buffer mensaje
#define CONEXIONES  2       // número de trasnmisiones esperadas
#define PUERTO      0       // asigna automáticamente un puerto libre

int main(int argc, char *argv[])
{
    int socketID;                       // identificador de dispositivo para el socket
    in_port_t puerto;                       // puerto publico
    struct sockaddr_in datosSocketSrv;      // dirección IP, tipo y puerto swl Servidor
    struct sockaddr_in datosSocketClt;      // dirección IP, tipo y puerto del cliente
    ssize_t res;                            // resultado de un recvfrom()
    size_t longdir;                         // tamaño estructura  sockaddr_in

    char mensaje[DIM];                      // buffer memoria para los mensajes
    int cont;                               // contador de conexiones


    // Comprueba los parámetros
    //*************************************************************************
    if(argc>2)
    {
        printf("Uso del programa: udpsrv1  <puerto>\n");
        exit(0);
    }

    if(argc==2)     puerto=atoi(argv[1]);   // puerto pasado como parámetro
    else            puerto=PUERTO;          // puerto por defecto

    /*
      1ª etapa: Creamos un socket PF_INET de tipo SOCK_DGRAM (obtenemos un manejador)
    ********************************************************************
    */
    socketID = socket( PF_INET, SOCK_DGRAM, 0);
    if(socketID<0)
    {
        perror("Creando socket");
        exit(-1);
    }


    /*
     2ª etapa:  Configuramos la estructura dirección:
           puerto 0 -> primero libre
     la dirección IP se toma la suya propia INADDR_ANY ya que es el servidor
    ********************************************************************
    */
```



```c
    datosSocketSrv.sin_family=AF_INET;          // tipo de dirección
    datosSocketSrv.sin_port = puerto;           // convierte el puerto a un número de red (h
    datosSocketSrv.sin_addr.s_addr = htonl(INADDR_ANY); // convierte la dirección IP a un nú


    /*
     3ª etapa: Nombramos el socket asignándole el puerto y dirección
    ********************************************************************
    */
    res = bind(socketID, (struct sockaddr *) &datosSocketSrv, sizeof datosSocketSrv);
    if(res<0)
    {
        perror("Nombrando el socket");
        close(socketID);
        exit(-2);
    }

    // Pedimos los datos del puerto asignado
    longdir=sizeof datosSocketSrv;
    res = getsockname(socketID, (struct sockaddr *) &datosSocketSrv, &longdir);
    if(res<0)
    {
        perror("Leyendo información  del socket");
        close(socketID);
        exit(-2);
    }

    // muestra los datos del servidor
    printf("---Servidor IP: %s,  Puerto: %u---\n",
        inet_ntoa(datosSocketSrv.sin_addr),ntohs(datosSocketSrv.sin_port));

    /*
     4ª etapa:   Recibimos inforamción del cliente. Opciones:
        ssize_t recvfrom(int socket, void *message, size_t length, int flags,
                  const struct sockaddr *dest_addr, size_t *dest_len);
    ********************************************************************
    */
    for(cont=0; cont<CONEXIONES; cont++)
    {
        printf("\nEsperando datos....\n");
        longdir=sizeof datosSocketClt;
        res = recvfrom(socketID, mensaje, DIM, 0,
                        (struct sockaddr *) &datosSocketClt, &longdir);
        if(res>0)
        {   // muestra los datos del cliente y los datos recibidos
            printf("%d bytes Recibidos desde [%s:%u]: %s\n", res,
                    inet_ntoa(datosSocketClt.sin_addr), ntohs(datosSocketClt.sin_port), mens
        }
    }

    /*
     5ª etapa:   Cerrar el socket
     ********************************************************************
    */
    close(socketID);

    exit(0);
}
```