```
/*
*  Plik " tcp_udp.h "
*
*  Definicje dla protokolow TCP i UDP
*/

#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <signal.h>

#define PORT_SERWERA_UDP  7000
#define PORT_SERWERA_TCP  7000
#define ADRES_SERWERA     "127.0.0.1"

#define STR_BLEDOW     2

#define BLAD_SERWERA1  "Serwer: Brak mozliwosci otwarcia gniazda\n"
#define BLAD_SERWERA2  "Serwer: Blad okreslenia adresu serwera\n"
#define BLAD_SERWERA3  "Serwer: Blad akceptacji zlecenia klienta\n"
#define BLAD_SERWERA4  "Serwer: Brak mozliwosci utworzenia potomka\n"

#define BLAD_KLIENTA1  "Klient: Brak mozliwosci otwarcia gniazda\n"
#define BLAD_KLIENTA2  "Klient: Brak mozliwosci polaczenia z serwerem\n"
#define BLAD_KLIENTA3  "Klient: Blad okreslenia adresu klienta\n"

#define BLAD_CZYTANIA_KL  "Klient: Blad czytania z gniazda\n"
#define BLAD_PISANIA_KL   "Klient: Blad pisania do gniazda\n"
#define BLAD_CZYTANIA_SR  "Serwer: Blad czytania z gniazda\n"
#define BLAD_PISANIA_SR   "Serwer: Blad pisania do gniazda\n"

#define MAX_DL_WIERSZA    512
#define MAX_DL_KOMUNIKATU 2048
```

```
/* * Plik " tcp_sr.c "
** Program serwera wspolbieznego dla protokolu TCP */
#include "tcp_udp.h"
#include "sock_fun.c"
void przetworzenie_zgloszenia_klienta( int gniazdo);
/*************************************************************/
main( int argc, char *argv[])
{
  int  gniazdo, nowe_gniazdo, dl_kom_klienta, id_potomka;
  struct sockaddr_in  adres_klienta, adres_serwera;

/* Otwarcie gniazda TCP (internetowe gniazdo strumieniowe */
  if ((gniazdo = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
       write(STR_BLEDOW,BLAD_SERWERA1,strlen(BLAD_SERWERA1));
       _exit(1); }
/* Dowiazanie adresu lokalnego serwera */
  memset((char *) &adres_serwera, 0, sizeof(adres_serwera)) ;
  adres_serwera.sin_family = AF_INET;
  adres_serwera.sin_addr.s_addr = htonl(INADDR_ANY);
  adres_serwera.sin_port = htons(PORT_SERWERA_TCP);

  if (bind(gniazdo , &adres_serwera , sizeof(adres_serwera)) < 0) {
      write(STR_BLEDOW,BLAD_SERWERA2,strlen(BLAD_SERWERA2));
      _exit(1); }

  listen( gniazdo,5);
  signal(SIGCLD, SIG_IGN); // ignorowanie smierci potomkow
  while (1) {
        /*  Czekanie na polaczenie z procesem klienta */
      dl_kom_klienta = sizeof(adres_klienta);
      nowe_gniazdo = accept( gniazdo,  &adres_klienta,
                             &dl_kom_klienta);   /* BLOKOWANIE !!! */
      if (nowe_gniazdo < 0) {
          write(STR_BLEDOW,BLAD_SERWERA3,strlen(BLAD_SERWERA3));
          _exit(1); }
      if (( id_potomka = fork()) < 0) {
          write(STR_BLEDOW,BLAD_SERWERA4,strlen(BLAD_SERWERA4));
          _exit(1); }
      else if (id_potomka == 0) {          /* proces potomny */
               close(gniazdo);
               przetworzenie_zgloszenia_klienta(nowe_gniazdo);
               _exit(0); }
       close(nowe_gniazdo);                /* proces macierzysty */
       } /* end while */
}
```

```
/*************************************************************/
/*  plik tcp_sr.c ciąg dalszy */


void przetworzenie_zgloszenia_klienta( int gniazdo)
{
/*      funkcja pobiera kolejne wiersze z gniazda strumieniowego
        i odsyla kazdy wiersz z powrotem do nadawcy
*/
  int n;
  char linia[MAX_DL_WIERSZA];

  while(1) {
    n = czytaj_linie(gniazdo,linia,MAX_DL_WIERSZA);
    if (n == 0) return;      /* koniec polaczenia */
    else if (n < 0) {
       write(STR_BLEDOW,BLAD_CZYTANIA_SR,strlen(BLAD_CZYTANIA_SR));
       _exit(1); }

    write(STR_BLEDOW, linia, strlen(linia));

    if (pisz_linie(gniazdo, linia, n) != n) {
       write(STR_BLEDOW,BLAD_PISANIA_SR,strlen(BLAD_PISANIA_SR));
       _exit(1); }
  }
}
```

```
/* *     Plik " tcp_kl.c "
*   *    Program klienta dla protokolu TCP */

#include "tcp_udp.h"
#include "sock_fun.c"
void klient_TCP(FILE *plik, int gniazdo);
/*****************************************************************/

main( int argc, char *argv[])
{
  int  gniazdo;
  struct sockaddr_in  adres_serwera;

/*  Umieszczenie w strukturze adresowej adresu serwera */

  memset((char *) &adres_serwera, 0, sizeof(adres_serwera)) ;
  adres_serwera.sin_family = AF_INET;
  adres_serwera.sin_addr.s_addr = inet_addr(ADRES_SERWERA);
  adres_serwera.sin_port = htons(PORT_SERWERA_TCP);

/*  Otwarcie gniazda TCP (internetowe gniazdo strumieniowe) */

  if ((gniazdo = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
       write(STR_BLEDOW,BLAD_KLIENTA1,strlen(BLAD_KLIENTA1));
       _exit(1); }

/*  Proba polaczenia z serwerem    */

      if ( connect( gniazdo, &adres_serwera,
                    sizeof(adres_serwera)) < 0 ) {
          write(STR_BLEDOW,BLAD_KLIENTA2,strlen(BLAD_KLIENTA2));
          _exit(1); }

       klient_TCP(stdin,gniazdo);
       close(gniazdo);
       _exit(0);
}
```

```
/* *     Plik " tcp_kl.c " - ciąg dalszy */

void klient_TCP(FILE *plik, int gniazdo)
{
/*      funkcja pobiera kolejne wiersze z pliku i odsyla do
        gniazda strumieniowego. Nastepnie odbiera z gniazda
        echo wyslanych wierszy i odsyla do wyjscia standardowego.
*/
  int n;
  char nadawana[MAX_DL_WIERSZA], odebrana[MAX_DL_WIERSZA +1];

  while(fgets(nadawana,MAX_DL_WIERSZA,plik) != NULL) {
     n=strlen(nadawana);
     if ( pisz_linie(gniazdo,nadawana,n) < n ) {
         write(STR_BLEDOW,BLAD_PISANIA_KL,strlen(BLAD_PISANIA_KL));
         _exit(1); }

/*  Pobranie wiersza z gniazda i odeslanie do standardowego wyjscia */

    n = czytaj_linie(gniazdo, odebrana, MAX_DL_WIERSZA);
    if (n < 0) {
       write(STR_BLEDOW,BLAD_CZYTANIA_KL,strlen(BLAD_CZYTANIA_KL));
       _exit(1); }
    fputs ( odebrana, stdout);
  }
}
```

```
/*  Plik " udp_sr.c "
    Program serwera dla protokolu UDP */

#include "tcp_udp.h"

main( int argc, char *argv[])
{
  int  gniazdo;
  int n, dl_klienta;
  char komunikat[MAX_DL_KOMUNIKATU];
  struct sockaddr_in  adres_klienta, adres_serwera;

/*  Otwarcie gniazda UDP (internetowe gniazdo datagramowe) */

  if ((gniazdo = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
       write(STR_BLEDOW,BLAD_SERWERA1,strlen(BLAD_SERWERA1));
       _exit(1); }

/*  Dowiazanie adresu lokalnego serwera */

  memset((char *) &adres_serwera, 0, sizeof(adres_serwera)) ;
  adres_serwera.sin_family = AF_INET;
  adres_serwera.sin_addr.s_addr = htonl(INADDR_ANY);
  adres_serwera.sin_port = htons(PORT_SERWERA_UDP);

  if (bind(gniazdo, &adres_serwera , sizeof(adres_serwera)) < 0) {
      write(STR_BLEDOW,BLAD_SERWERA2,strlen(BLAD_SERWERA2));
      _exit(1); }
  while(1) {
    dl_klienta = sizeof(adres_klienta);
    n = recvfrom(gniazdo,komunikat,MAX_DL_KOMUNIKATU, 0,
                 &adres_klienta, &dl_klienta );
    if (n < 0) {
       write(STR_BLEDOW,BLAD_CZYTANIA_SR,strlen(BLAD_CZYTANIA_SR));
       _exit(1); }

    write(STR_BLEDOW, komunikat, strlen(komunikat));

    if(sendto(gniazdo,komunikat, n, 0, &adres_klienta, dl_klienta) != n)
       { write(STR_BLEDOW,BLAD_PISANIA_SR,strlen(BLAD_PISANIA_SR));
       _exit(1); }
  }
}
```

```
/* *     Plik " udp_kl.c "
*   Program klienta dla protokolu UDP */

#include "tcp_udp.h"

void klient_UDP(FILE *plik, int gniazdo,
              struct sockaddr_in *adres_serwera, int dl_serw);

/*************************************************************************/
main( int argc, char *argv[])
{
  int  gniazdo;
  struct sockaddr_in  adres_serwera, adres_klienta;

/*  Umieszczenie w strukturze adresowej adresu serwera */

  memset((char *) &adres_serwera, 0, sizeof(adres_serwera)) ;
  adres_serwera.sin_family = AF_INET;
  adres_serwera.sin_addr.s_addr = inet_addr(ADRES_SERWERA);
  adres_serwera.sin_port = htons(PORT_SERWERA_UDP);

/*  Otwarcie gniazda UDP (internetowe gniazdo datagramowe) */

  if ((gniazdo = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
       write(STR_BLEDOW,BLAD_KLIENTA1,strlen(BLAD_KLIENTA1));
       _exit(1); }

/*  Dowiazanie adresu lokalnego    */

      memset((char *) &adres_klienta, 0, sizeof(adres_klienta)) ;
      adres_klienta.sin_family = AF_INET;
      adres_klienta.sin_addr.s_addr = htonl(INADDR_ANY);
      adres_klienta.sin_port = htons(0);

      if ( bind( gniazdo,  &adres_klienta, sizeof(adres_klienta)) < 0 )
         { write(STR_BLEDOW,BLAD_KLIENTA3,strlen(BLAD_KLIENTA3));
          _exit(1); }

       klient_UDP(stdin, gniazdo, &adres_serwera, sizeof(adres_serwera));
       close(gniazdo);
       _exit(0);
}
```

```
/* *     Plik " udp_kl.c "   ciąg dalszy */


void klient_UDP(FILE *plik, int gniazdo,
              struct sockaddr_in *adres_serwera, int dl_serw)
{
/*      funkcja pobiera kolejne wiersze z pliku i odsyla do
        gniazda datagramowego. Nastepnie odbiera z gniazda
        echo wyslanych wierszy i odsyla do wyjscia standardowego.
*/
  int n;
  char nadawana[MAX_DL_WIERSZA], odebrana[MAX_DL_WIERSZA +1];

  while(fgets(nadawana,MAX_DL_WIERSZA,plik) != NULL) {
     n=strlen(nadawana);
     if ( sendto(gniazdo, nadawana, n, 0, adres_serwera, dl_serw) < n )
       { write(STR_BLEDOW,BLAD_PISANIA_KL,strlen(BLAD_PISANIA_KL));
         _exit(1); }

/*  Pobranie wiersza z gniazda i odeslanie do
        standardowego wyjscia    */

    n = recvfrom(gniazdo, odebrana, MAX_DL_WIERSZA, 0,
                 (struct sockaddr *) 0, (int *) 0);
    if (n < 0) {
       write(STR_BLEDOW,BLAD_CZYTANIA_KL,strlen(BLAD_CZYTANIA_KL));
       _exit(1); }

    odebrana[n] = 0;
    fputs ( odebrana, stdout);
  }
}
```