1 Klijentski deo koda

client.c

```
/*
 * Primer client koda koji koristi
 * funkcije sendstring i receivestring
 * deklarisane u protocol.h da bi
 * razmenjivao poruke sa klijentom.
 *
 * Autor: Ikac, ikac.ikax@gmail.com
 */

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

#include "protocol.h"


void error(const char *msg)
{
  /*
   * funkcija koja ispisuje predatu
   * poruku, i izlazi iz programa
   */
 perror(msg);
 exit(1);
}

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

 if (argc < 3) {
  fprintf(stderr,
   "usage %s hostname port \n",
   argv[0]);
  exit(0);
 }

 portno = atoi(argv[2]);
```

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);


if (sockfd < 0)
 error("ERROR opening socket");

 /*
  * Prihvatanje podataka o serveru na
  * osnovu njegove adrese predate pri
  * pokretanju programa
  *
  */
server = gethostbyname(argv[1]);


 perror("Connecting to host");

bzero((char *)&serv_addr, sizeof(serv_addr)); // nuliranje cele
    strukture

serv_addr.sin_family = AF_INET; //odabir Internet adresa
bcopy((char *)server->h_addr, (char *)&serv_addr.sin_addr.
   s_addr,
      server->h_length); //kopiranje adrese servera
serv_addr.sin_port = htons(portno); //navodjenje porta na
   serveru


if (connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(
   serv_addr)) <
    0)
 error("ERROR connecting"); //konektovanje na zeljeni server
    preko sockfd socket deskriptora

char *buffer;

 const short len = 256;

buffer = malloc(sizeof(char)*len);
 memset(buffer, ' ', len); //nuliranje buffera

strcpy(buffer, "THIS IS A MESSAGE");//smestanje stringa u
   buffer
n = sendstring(sockfd, buffer); //slanje stringa serveru

n = receivestring(sockfd, &buffer); //prihvatanje odziva

if(buffer) printf("%s\n", buffer);
```

```
  if(buffer) free(buffer);

 close(sockfd);//zatvaranje socketa i zavrsetak rada klijenta
 return 0;
}
```

2 Serverski deo koda

server.c

```
/*
 * Primer server koda koji koristi
 * funkcije sendstring i receivestring
 * deklarisane u protocol.h da bi
 * razmenjivao poruke sa klijentom.
 *
 * Autor: Ikac, ikac.ikax@gmail.com
 */

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

#include "protocol.h"

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

int main(int argc, char **argv)
{
 int sockfd, newsockfd, portno;
 socklen_t clilen;

 struct sockaddr_in serv_addr, cli_addr;
 int n;

 if (argc < 2) {
  fprintf(stderr, "ERROR, no port provided\n");
  exit(1);
 }

 sockfd = socket(AF_INET, SOCK_STREAM, 0);

 bzero((char *)&serv_addr, sizeof(serv_addr));

 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("ERROR on binding");

 listen(sockfd, 5);

 clilen = sizeof(cli_addr);

  char * buffer;

 while (1) {
  newsockfd =
       accept(sockfd, (struct sockaddr *)&cli_addr, &clilen);

  if (newsockfd < 0)
   error("ERROR on accept");

  n = receivestring(newsockfd, &buffer);

  if (n < 0)
   error("ERROR reading from socket");

  printf("Here is the message: %s\n", buffer);

  n = sendstring(newsockfd, buffer);

  if (n < 0)
   error("ERROR writing to socket");

  close(newsockfd);
 }

 close(sockfd);
 return 0;
}
```

3 Protocol.h

protocol.h

```
/*
 * Header fajl potreban procotol.c kodu
 * kao i klijentskom i serverskom kodu da
 * se kompajliraju.
 *
 * Ovo je vrlo prosta verzija protokola
 * gde se ispred stringa ubaci zaglavlje
 * velicine HEADERLEN koje sadrzi broj
 * karaktera koje sadrzi poruka.
 *
 * Nakon njega ide poruka, koja je tipa
 * string, pa se niz karaktera zavrsava sa '\0'
 *
 * Izgled zaglavlja:
 *
 *  ____________________
 *  |0010|0123456789\0|
 *  --------------------
 *     ||           ||
 *   zaglavlje    poruka
 *
 * MESSAGELEN treba da ogranici broj karaktera
 * u poruci, kako bi duzina poruke
 * mogla da odgovara broju u zaglavlju
 *
 * Implementacije funkcija sendstring i
 * receivestring su date u protocol.c fajlu
 *
 * Nacin kompajliranja je dat u make fajlu
 *
 * Autor: Ikac, ikac.ikax@gmail.com
 */

#define HEADERLEN 4
#define MESSAGELEN (10^HEADERLEN-1)

int sendstring(int, char * );
int receivestring(int, char ** );
```

4 Implementacija protokola

protocol.c

```
/*
 * Ovaj fajl sadrzi implementacije funkcija
 * koje su deklarisane u protoco.h fajlu
 *
 * Imajuci u vidu zamisljen protokol ove
 * funkcije ga implementiraju na jedan od
 * mogucih nacina
 *
 * Generalno posmatrajuci, nacin rada je sledeci:
 * 1. Kada saljemo odredjeni niz karaktera,
 * broj karaktera se belezi u zaglavlju, zatim
 * se poruka 'zalepi' za zaglavlje i doda joj se
 * \0 karakter. On ne ulazi u broj karaktera u
 * zaglavlju. Zato se salje bafer koji ima duzinu
 * HEADERLEN + messagelen + 1. Ovako se celokupan
 * 'paket' posmatra kao string.
 *
 * 2. Prihvatanje 'paketa' se obavlja jednostavnije.
 * Procita se prvo HEADERLEN bajtova iz sistemskog bafera
 * i tako se odredi koliko jos bajtova treba da se
 * ucita, kako bi se formirao validan 'paket'. Pored
 * broja karaktera, vodi se racuna i o \0 koji se nalazi
 * iza njih, tako da se sa sistemskog soketa povlaci
 * messagelen+1 bajt.
 *
 * Autor: Ikac, ikac.ikax@gmail.com
 */

#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include "protocol.h"

int sendstring(int sockfd, char * buffer)
{
  int n, len;
  char format[4];
  char headerbuffer[HEADERLEN];

  len = strlen(buffer); //odredjujemo duzinu bafera

  sprintf(format, "%%0%dd", HEADERLEN);//pravimo format %04d

  sprintf(headerbuffer, format, len);//formiranje zaglavlja
```

```
  /*
   * kopira se sadrzaj hedera a zatim i sadrzaj bafera
   * u bafer koji ce nam sluziti za slanje
   */
  char buffer_out[HEADERLEN+len+1];
  strcpy(buffer_out, headerbuffer);
  strcpy(buffer_out+ HEADERLEN, buffer);

  /*
   * stavljamo nul karakter da bi
   * se sekvenca tretirala kao string
   */
  buffer_out[HEADERLEN+len] = '\0';

  n = write(sockfd, buffer_out, HEADERLEN+len+1);

  #ifdef _DEBUG_
    perror("Sending message");
  #endif

  return n;
}

int receivestring(int sockfd, char ** buffer)
{
  int n;
  char headerbuffer[HEADERLEN];

  n = read(sockfd, headerbuffer, HEADERLEN);//citanje 4 bajta za
      heder
  int messagelen = atoi(headerbuffer);//pretvaranje hedera u
     broj koji oznacava duzinu poruke u bajtovima

  *buffer = malloc(sizeof(char)*(messagelen+1));//+1 bajt za nul
      karakter
  memset(*buffer, ' ', messagelen+1);//nulovanje bafera
  n = read(sockfd, *buffer, messagelen+1);//uzimanje iz
     sistemskog bafera ukljucujuci i nevidljivi nul karakter
     (+1)

  #ifdef _DEBUG_
    perror("Receiving message");
  #endif

  if(n < 0) return n;
  else return messagelen;
}
```

5 Make file za kompajliranje

makefile

```
#
# Ovo je make file za kompajliranje postojecih c fajlova.
# Upozorenje: razmaci na pocetku su tabulatori
#

all: client.c server.c protocol.o
 gcc client.c protocol.o -o client
 gcc server.c protocol.o -o server

protocol.o: protocol.h protocol.c
 gcc protocol.c -c

clean:
 rm client server protocol.o
```

Literatura

[1] Understanding big and little endian byte order, *http://betterexplained.com/articles/understanding-big-and-little-endian-byte-order/*

[2] Beej's guide to socket programming in C, *http://beej.us/guide/bgnet/output/html/singlepage/bgnet.html*

[3] Errors: errno in UNIX programs, Chris Herborth, *http://www.ibm.com/developerworks/aix/library/au-errnovariable/*