**Ime in priimek:** **Datum:**

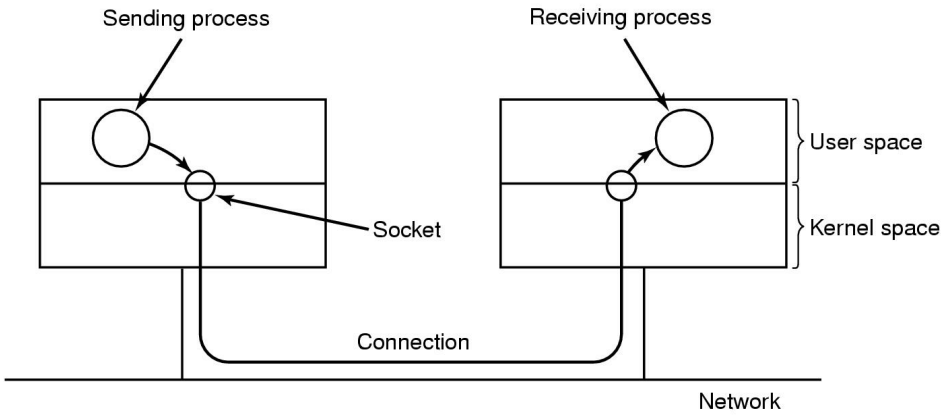
## Vtičnica (Socket)

**Namen vaje** je:

- Spoznati se z medprocesno komunikacijo
- Vzpostavljanje komunikacije preko internetne povezave

GRADIVA: https://github.com/matejkrenLTFE/POKS2022
Pri medprocesni komunikaciji se pogosto uporablja model odjemalec/strežnik (client/server). Proces odjemalec (client) se poveže na strežnik (server) pri čemer običajno zahteva podatke, ki mu jih strežnik ob uspešni zahtevi vrne. Odjemalec potrebuje naslov strežnika. Potem, ko je povezava vzpostavljena, lahko obe strani pošiljata in sprejemata podatke.

Za vzpostavitev povezave preko internetnega omrežja se na obeh straneh uporablja socket (vtičnica).




Za naslavljanje v internetni domeni se uporabljajo IP naslovi in porti.
Največ se uporabljata:

- stream socket za TCP in
- datagram socket za UDP.

Koraki za vzpostavitevi TCP povezave preko socket-a na strani odjemalca:

1. Kreiranje s sistemskim ukazom: socket()
2. Povezava socket-a z naslovom strežnika: connect()
3. Sprejem in oddaja podatkov, npr. read(), write()

Koraki za vzpostavitevi TCP povezave preko socket-a na strani strežnika:

1. Kreiranje s sistemskim ukazom: socket()
2. Vezava socket-a z naslovom strežnika; za internet je to port strežnika: bind()
3. Poslušanje za nove povezave (odjemalcev): listen()
4. Sprejem povezave: accept()
5. Sprejem in oddaja podatkov, npr. read(), write()

Strežnik za datagram (UDP) ne potrebuje korakov 3 (listen) in 4 (accept), za sprejem novih TCP povezav. UDP socket za sprejem sporočil uporablja API funkcijo recvfrom() in za oddajo sendto().

Več na:
http://www.linuxhowtos.org/C_C++/socket.htm
http://www.cs.cf.ac.uk/Dave/C/node28.html

Protokol SIP uporablja UDP ali TCP povezave, ki so realizirane s pomočjo vtičnic (*socket*).

Dopolnite pripravljena programa za strežnik in odjemalec za datagram (UDT) povezavo z uporabo vtičnice (*socket*). Nastavite pravilne IP naslove in porte od 5065 do 5069 za povezavo makete samo nase in med maketami.

Prevajanje:

**gcc server_udp.c –o server**

**gcc client_udp.c –o client**

## 1.VAJA

Dopolnite spodaj pripravljena programa *server_udp.c* in *client_udp.c*, da se bo vzpostavila UDP povezava preko vtičnic. Strežnik najprej čaka na sprejem, odjemalec pa začne z oddajo sporočila (vpisanega preko konzole). Nato si izmenjujeta sporočila do končnega sporočila. Nato zapremo UDP povezavo.

Stanje UDP povezav spremljajte s: netstat –lnup

Potek povezovanja spremljajte s: sudo tcpdump –i eth0 udp

**Opišite rešitev, nastavljene naslove, diagram poteka in obnašanje programa**

## 2.VAJA

**Povezava odjemalca in strežnika na različnih maketah**

a. Povežite se s sosednjo skupino, ena kupina starta *server*, druga *client*.
b. Zamenjajte vlogi *server* – *client*.
c. Potek povezovanja spremljajte z **tcpdump** programom.

**Opišite rešitev, nastavljene naslove, diagram poteka in obnašanje programa**

**server_udp.c**

```
/* Creates a datagram UDP server.  The port number is passed as an
argument. */
/* This server runs forever */

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

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

int main(int argc, char *argv[])
{
   int sock, length, n;
   struct sockaddr_in server, from;
   socklen_t fromlen;
   char buf[1024];

   if(argc < 2) {
      fprintf(stderr, "ERROR, no port provided\n");
      exit(0);
   }

   sock=socket(AF_INET, SOCK_DGRAM, 0);  /* create new socket */
   if(sock < 0) error("Opening socket");
   length = sizeof(server);
   bzero(&server, length);
   server.sin_family=AF_INET;  /* Internet */
   server.sin_addr.s_addr=INADDR_ANY;
   server.sin_port=htons(atoi(argv[1])); /* port */
   if (bind(sock,(struct sockaddr *)&server, length)<0) /* bind socket with
port */
       error("binding");
   fromlen = sizeof(struct sockaddr_in);
   while (1)
   {
       n = recvfrom(sock,buf,1024,0,(struct sockaddr *)&from, &fromlen);
       if(n < 0) error("recvfrom");
       printf("Received a datagram:%s\n ",buf);
       printf("Please enter the message: ");
       bzero(buf,1024);
       fgets(buf,1024,stdin);
       printf("Sent %s \n", buf);
       n = sendto(sock,buf,1024,0,(struct sockaddr *)&from, fromlen);
       if(n < 0) error("sendto");
   }
   close(sock);
   return 0;
 }
```

**client_udp.c**

```
/* Creates datagram UDP client in the internet domain. */
/* IP address and port are passed as argument */

int main(int argc, char *argv[])
{
   struct hostent *hp;

   if(argc != 3) { printf("Usage: client IP port\n");
                   exit(1);
   }
   sock= socket(AF_INET, SOCK_DGRAM, 0);
   if(sock < 0) error("socket");

   server.sin_family = AF_INET;
   hp = gethostbyname(argv[1]);
   if(hp==0) error("Unknown host");
   bcopy((char *)hp->h_addr, (char *)&server.sin_addr, hp->h_length);
   server.sin_port = htons(atoi(argv[2]));
   length=sizeof(struct sockaddr_in);

   printf("Please enter the message: ");
   bzero(buf,1024);
   fgets(buf,1024,stdin);
   printf("Send %s \n", buf);
   n=sendto(sock,buf,1024,0,(const struct sockaddr *)&server,length);
   if(n < 0) error("Sendto");

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